UNIVERSITAT AUTÒNOMA DE BARCELONA
ESCOLA TÈCNICA SUPERIOR D'ENGINYERIA

TESI DOCTORAL

Ensemble Case Based Learning for Multi-Agent Systems

Santi Ontañón Villar

Director: Enric Plaza i Cervera

Bellaterra, Abril 2005

CONSELL SUPERIOR D'INVESTIGACIONS CIENTÍFIQUES
INSTITUT D'INVESTIGACIÓ EN INTEL·LIGÈNCIA ARTIFICIAL

UNIVERSITAT AUTÒNOMA DE BARCELONA
ESCOLA TÈCNICA SUPERIOR D'ENGINYERIA

TESI DOCTORAL

Ensemble Case Based Learning for Multi-Agent Systems

Bellaterra, Abril 2005

Memòria presentada per Santi Ontañón Villar per optar al títol de Doctor en Informàtica per la Universitat Autònoma de Barcelona sota la direcció del Dr. Enric Plaza i Cervera. El treball presentat en aquesta memoria ha estat realitzat a l'Institut d'Investigació en Intel·ligència Artificial (IIIA), del Consell Superior d'Investigacions Científiques (CSIC)

Contents

Agradecimientos

Por mucho que en la portada de una tesis aparezca el nombre de un único autor, está bien claro que una tesis no la realiza una única persona, sino que es el resultado del trabajo de un grupo de personas. Por ello me gustaría agradecer la ayuda y apoyo de toda esa gente gracias a la cual esta tesis ha podido llevarse a cabo. En particular a Enric Plaza, el director de este trabajo, por su aportación de experiencia e ideas, y por su casi infinita paciencia al revisar y mejorar una por una las innumerables versiones de los capítulos que forman esta tesis. También me gustaría dedicar un agradecimiento especial a Eva Armengol y Josep Lluís Arcos, ya que sin su ayuda y trabajo esta tesis tampoco hubiera sido posible. Pero no solo a ellos, también quiero dedicar un agradecimiento al resto de miembros del IIIA, por crear un ambiente tan agradable, social y bueno para la investigación.

También me gustaría dar las gracias a mis amigos y a mi familia, y especialmente a mis padres y mi hermana, que han tenido que aguantar con paciencia mi comportamiento "ermitaño" durante los últimos meses del desarrollo de este trabajo.

Éste trabajo ha estado financiado por los proyectos IST-1999-19005 "IBROW", TIC2000-1414 "eInstitutor" y TIC2002-04146-C05-01 "SAMAP", y por una beca de la Generalitat de Catalunya CIRIT FI/FAP 2001.

Resumen

Esta monografía presenta un marco de trabajo para el aprendizaje en un escenario de datos distribuidos y con control descentralizado. Hemos basado nuestro marco de trabajo en Sistemas Multi-Agente (MAS) para poder tener control descentralizado, y en Razonamiento Basado en Casos (CBR), dado que su naturaleza de aprendizaje perezoso lo hacen adecuado para sistemas multi-agentes dinámicos. Además, estamos interesados en agentes autónomos que funcionen como *ensembles*. Un ensemble de agentes soluciona problemas de la siguiente manera: cada agente individual soluciona el problema actual individualmente y hace su predicción, entonces todas esas predicciones se agregan para formar una predicción global. Así pues, en este trabajo estamos interesados en desarrollar estrategias de aprendizaje basadas en casos y en ensembles para sistemas multi-agente.

Concretamente, presentaremos un marco de trabajo llamado *Razonamiento Basado en Casos Multi-Agente* ($\mathcal{M}$AC), una aproximación al CBR basada en agentes. Cada agente individual en un sistema $\mathcal{M}$AC es capaz de aprender y solucionar problemas individualmente utilizando CBR con su base de casos individual. Además, cada base de casos es propiedad de un agente individual, y cualquier información de dicha base de casos será revelada o compartida únicamente si el agente lo decide así. Por tanto, este marco de trabajo preserva la privacidad de los datos y la autonomía de los agentes para revelar información.

Ésta tesis se centra en desarrollar estrategias para que agentes individuales con capacidad de aprender puedan incrementar su rendimiento tanto cuando trabajan individualmente como cuando trabajan como un ensemble. Además, las decisiones en un sistema $\mathcal{M}$AC se toman de manera descentralizada, dado que cada agente tiene autonomía de decisión. Por tanto, las técnicas desarrolladas en este marco de trabajo consiguen un incremento del rendimiento como resultado de decisiones individuales tomadas de manera descentralizada. Concretamente, presentaremos tres tipos de estrategias: estrategias para crear ensembles de agentes, estrategias para realizar retención de casos en sistemas multi-agente, y estrategias para realizar redistribución de casos.

Abstract

This monograph presents a framework for learning in a distributed data scenario with decentralized decision making. We have based our framework in Multi-Agent Systems (MAS) in order to have decentralized decision making, and in Case-Based Reasoning (CBR), since the lazy learning nature of CBR is suitable for dynamic multi-agent systems. Moreover, we are interested in autonomous agents that collaboratively work as ensembles. An ensemble of agents solves problems in the following way: each individual agent solves the problem at hand individually and makes its individual prediction, then all those predictions are aggregated to form a global prediction. Therefore, in this work we are interested in developing ensemble case based learning strategies for multi-agent systems.

Specifically, we will present the *Multi-Agent Case Based Reasoning* ($\mathcal{M}$AC) framework, a multi-agent approach to CBR. Each individual agent in a $\mathcal{M}$AC system is capable of individually learn and solve problems using CBR with an individual case base. Moreover, each case base is owned and managed by an individual agent, and any information is disclosed or shared only if the agent decides so. Thus, this framework preserves the privacy of data, and the autonomy to disclose data.

The focus of this thesis is to develop strategies so that individual learning agents improve their performance both individually and as an ensemble. Moreover, decisions in the $\mathcal{M}$AC framework are made in a decentralized way since each individual agent has decision autonomy. Therefore, techniques developed in this framework achieve an improvement of individual and ensemble performance as a result of individual decisions made in a decentralized way. Specifically, we will present three kind of strategies: strategies to form ensembles of agents, strategies to perform case retention in multi-agent systems, and strategies to perform case redistribution.

Chapter 1

Introduction

This monograph presents a framework for learning in a distributed data scenario with decentralized decision making. We have based our framework in Multi-Agent Systems (MAS) in order to have decentralized decision making, and in Case-Based Reasoning (CBR), since the lazy learning nature of CBR is suitable for dynamic multi-agent systems. Moreover, we are interested in autonomous agents that collaboratively work as ensembles. An ensemble of agents solves problems in the following way: each individual agent solves the problem at hand individually and makes its individual prediction, then all those predictions are aggregated to form a global prediction. Therefore, in this work we are interested in multi-agent learning techniques to improve both individual and ensemble performance. In this chapter we are going to motivate our framework and state our research goals. Then, we will present a road-map of the contents of this thesis.

1.1 Motivation

Data of interest is distributed among several sources in many real world applications. Therefore, traditional machine learning techniques cannot be directly used, since they assume a centralized access and control of the data. In order to deal with the distributed nature of data of interest, several approaches have been proposed.

A first approach is to collect data from the different data sources, and store it in a centralized repository, where machine learning techniques can be used. However, in many domains this approach is not desirable or even not feasible for a variety of reasons. For instance for property rights, bandwidth limitations, or because of management concerns (since data owners may not be wiling to cede their data to a centralized repository because they want to maintain control over their data).

A second approach is based on the fact that many machine learning techniques can be decentralized. For instance, certain decision trees techniques can

be used in a distributed way [17] by locally computing certain statistics at each data source, and then sending those statistics to a central repository where a decision tree can be learnt. However, this second approach only solves the problem of bandwidth limitation and is only applicable to machine learning techniques that can be decentralized.

The previous two approaches correspond respectively to *data warehousing* and *distributed machine learning*. Moreover, they both share two assumptions: a) that the *only* problem is that data is distributed, and b) that a single model of all the data is going to be constructed. Let us develop both issues in more detail.

In many applications the fact that data is distributed among several sources is not the only problem. The problem is that the different data sources may correspond to different partners or organizations, and that those organizations may consider their cases as assets and may not be willing to allow other organizations to have access to their data either because of ownership rights or management concerns. However, these organizations would be interested in benefiting from the collaboration with other organizations but keeping the control of their data.

A way to deal with the privacy rights and management concerns may be Multi-Agent Systems (MAS) [26, 43, 89], a sub-field of distributed artificial intelligence that studies how autonomous entities (a.k.a. *agents*) interact, in a collaborative or competitive way. Researchers in multi-agent systems focus mainly on architectures that can support agent systems [27], and on distributed mechanisms to coordinate multiple agents so that they can jointly accomplish a given task [43]. The intersection of learning and multi-agent systems is called *Multi-Agent Learning* (MAL) [81], and addresses the integration of learning in multi-agent systems. However, it is a relatively new field and a lot of work still remains to be done. Moreover, most of the work focuses on reinforcement learning and evolutionary algorithms.

We have to take into account the difference between a distributed algorithm and a multi-agent system: in a distributed algorithm there is a global *goal* (and there are several processes running in parallel to accomplish that goal), while in a multi-agent system each individual agent has its own goals. The joint goals emerge from the interaction among several agents following an interaction protocol: eventually a group of agents may collaborate together to solve a task, but only if that task is beneficial for each one's goals. Thus, multi-agent systems are a suitable tool to preserve the privacy and management concerns in the distributed data scenario, where each organization can be modelled as an agent that has control over its private data. Moreover, two organizations will only collaborate when they both are interested in collaboration.

Concerning the issue of building a single model, it is not obvious that building a single model of the data is always the best solution. For instance, ensemble learning is a subfield of machine learning based on constructing several models of the same data and then combine them in order to reduce error with respect to using a single model. Thus, at least in principle, having multiple models of data is better than having a single model. Ensemble learning methods are centralized and, given a training set, construct a set of different classifiers by

training each classifier with a variation of the training set or with a different learning method. Ensemble methods reduce error with respect to using a single model for three main reasons [23]: first, they enhance the expressive power of the classifiers (since the ensemble can express hypothesis that cannot be expressed with a single classifier); second, they reduce the impact of having a small training sample (since a small sample increases the likelihood of finding a wrong hypothesis, and the aggregation of several hypotheses is more likely to perform better); and third, they reduce the problem of getting stuck in a local minimum during learning (since each classifier is expected to find a different local minimum, and their aggregation is expected to perform better). Moreover, we will call the classification error reduction achieved by ensemble methods the *ensemble effect*.

A basic assumption of ensemble learning methods is a centralized control over the data. This assumption does not hold in multi-agent systems where control is decentralized, since each individual agent controls part of the data and each agent is autonomous. Therefore, ensemble learning techniques cannot be directly applied to build multiple models in a distributed data setting modelled using multi-agent systems. Another problem is that ensembles must satisfy some preconditions in order to perform well (that we will refer to as the "preconditions of the ensemble effect"), and in an open multi-agent system we have no guarantee that the individual agents satisfy those preconditions. Thus, if the benefits of the ensemble effect are desired, alternative techniques are needed.

In this work, we are going to present a framework to deal with learning in distributed data, based on multi-agent systems, and where we are interested in using multiple models of the data. Moreover, due to the open and dynamic nature of multi-agent systems, we are interested in Lazy Learning techniques, and specially Case-Based Reasoning (CBR) [1]. Lazy learning techniques are better suited for open and dynamic systems than eager learning techniques, since they are not sensitive to changes in the data, while eager learning techniques have to rebuild (or adapt) their models of the data every time that data changes.

Case-Based Reasoning (CBR) is a specific type of lazy learning, that consists of storing problem solving experiences (called *cases*) so that they can be reused to solve future problems. CBR basically relies on the assumption that similar problems require similar solutions. A typical CBR system solves a problem by retrieving cases stored in its case memory (called the *case base*) that are similar to the problem at hand, and reusing the solution of the retrieved cases to solve the problem. Once the proposed solution for the new problem has been revised, a new case is created and it can be retained in the case base. This problem solving cycle is known as the R4 model [1], that divides the activity of a CBR system in four processes: *retrieve*, *reuse*, *revise*, and *retain*.

Classical CBR considers a single case base with which to solve problems. Applying CBR to multi-agent systems arises several issues. For instance, the reuse process in the R4 model assumes that cases from a single case base have been retrieved. However, in a multi-agent system several agents may control different case bases. Moreover, agents may consider their case bases private, and thus the problem is not simply to retrieve cases from several case bases,

but how several agents (each one controlling its case base) can collaborate to solve problems using CBR, without violating neither autonomy of agents nor the privacy of data. Moreover, case retention is not obvious either: in a classical CBR system a case is retained into the case base after being solved. However, in a multi-agent system, where a group of agents may have collaborated to solve a case, it is not clear which agent or agents should retain that case. Therefore, at least two new issues appear: how to solve problems in a collaborative way, and how to perform retention in a collaborative way.

1.2 The Framework

In this thesis we will present the *Multi-Agent Case Based Reasoning Systems* ($\mathcal{M}$AC) framework for learning in distributed data settings with decentralized decision making. Agents in a multi-agent system (MAS) have autonomy of decision, and thus control in a MAS is decentralized. Moreover, $\mathcal{M}$AC systems take a social agents approach based on electronic institutions [27]. In electronic institutions, coordination among agents is performed by means of shared interaction protocols. Basically, an interaction protocol defines a set of interaction states, and the set of actions that each agent can perform in each interaction state. Each agent uses individual decision policies to choose form the set of possible actions at each interaction state. In the $\mathcal{M}$AC framework, agents collaborate by means of *collaboration strategies*, consisting on an interaction protocol and a set of individual decision policies. Thus, the electronic institutions offers us a framework where autonomy in decentralized decision making is preserved.

Each individual agent in a $\mathcal{M}$AC system is capable of individually learn and solve problems using CBR, with an individual case base. Moreover, each case base is owned and managed by an individual agent, and any information is disclosed or shared only if the agent decides so. Thus, this framework preserves the privacy of data, and the autonomy to disclose data. Therefore, the $\mathcal{M}$AC framework extends the case-based reasoning paradigm to multi-agent systems. Moreover, notice that since each individual agent is an individual case based reasoner, agents have the ability to learn individually.

The focus of this thesis is investigating ensembles of agents. Specifically, we are interested in studying how to organize agents into ensembles, and how they can collaborate to achieve the ensemble effect. For this purpose, we need to address other issues such as determining when an agent should solve a problem individually or organizing an ensemble, and determining which agents should be present in an ensemble. Moreover, we are also interested in studying how individual and ensemble performance can be improved. For this purpose, we need to address several other issues such as learning how to select the members of an ensemble, learning how to improve individual performance maintaining (or even improving) ensemble performance, determining how to redistribute cases among the agents to achieve better distributions (in terms of performance), and deciding which agents should retain new cases so that individual and ensemble performance improves.

In order to address those issues we will design collaboration strategies, i.e. we will design interaction protocols and individual decision policies. Thus, these collaboration strategies will allow agents to form ensembles and to improve their performance as a result of individual decisions made in a decentralized way. In the next section we will present a detailed list of our research goals in the $\mathcal{M}$AC framework.

1.3 The Goals

The main goal of the thesis is to study the effects of distributed data and decentralized individual decision making in learning processes, and specifically in a multi-agent setting where individual agents own different parts of the data.

Moreover, in this thesis we have also several goals related with CBR, multi-agent systems and ensemble learning:

- The first goal is the integration of the three related areas (ensemble learning, case-based reasoning and multi-agent systems) and formally define the Multi-Agent Case Based Reasoning framework ($\mathcal{M}$AC).
- How to achieve the *ensemble effect* in multi-agent systems by forming *committees of agents*. Thus, allowing agents to improve their performance as an ensemble as a result of their individually made decisions.
- Analyze the ensemble effect and its preconditions in a wide range of situations so that measures can be defined to characterize ensembles, and thus predicted their performance. This measures are required so that agents trying to behave as an ensemble can measure how well will they perform as an ensemble and decide which actions should be taken to improve their ensemble performance.
- Develop learning techniques to improve the performance of the agents (both individually and as an ensemble). Specifically, we are interested in two types of leaning: learning processes that allow agents to improve individual problem solving performance, and learning processes that allow agents to improve their collaboration, i.e. learning when to collaborate and with whom to collaborate.
- Extend the Case-Based Reasoning paradigm to deal with multi-agent systems, in such a way that agent autonomy, data privacy, and individual data control are preserved in the autonomous agents. The four processes of CBR (retrieve, reuse, revise and retain) have to be rethought. Specifically, in this thesis we focus on how reuse and retain can be adapted to work in multi-agent systems.
- Develop techniques to perform the reuse process of CBR in a decentralized way. Decentralized reuse would be preferable to decentralized retrieve under certain conditions, since decentralized reuse can preserve privacy of the data while decentralized retrieve cannot. Decentralized reuse

should be able to determine a global prediction through a collaborative process (based for instance in voting or in any other aggregation mechanism) among several agents that have performed the retrieval process individually.

- Develop techniques to perform the retain process of CBR in a decentralized way. Decentralized retain raises several new problems with respect to classical retain involving a single case base. For instance, using decentralized retain, a group of agents solving a problem has to decide not only if a case is going to be retained, but which agent or agents will retain it. Moreover, decentralized retain has to take into account the performance of the agents when they act as an ensemble, in addition to the performance of the agents solving problems individually.

- Develop techniques for decentralized data redistribution. Since in a multi-agent system we cannot make any assumption about the initial distribution of data among the agents, it would be interesting to study data redistribution techniques that rely in decentralized control and that preserve the autonomy of the agents. Redistribution techniques have the goal of achieving a distribution of data that improves both individual and ensemble performance.

1.4 The Thesis

In this section we will present a road map of the thesis, shortly summarizing the contents of the rest of the chapters and appendices. Figure 1.1 shows a condensed view of the contents of the thesis.

- **Chapter 2** presents an overview of the state of the art in the areas related to the research presented in this thesis. First, related work in Ensemble Learning is presented, emphasizing in the work related to the *ensemble effect* and on different methods for creating ensembles. Then, related work on Case-Based Reasoning is considered, specifically the work related to retain techniques and to explanations generation. Finally, recent work on multi-agent learning is reviewed. Specifically, four areas of multi-agent learning are reviewed: reinforcement learning and genetic algorithms in multi-agent systems (since these two are the most applied techniques to multi-agent learning) and also Case-Based Reasoning in multi-agent systems.

- **Chapter 3** presents the $\mathcal{M}$AC framework for ensemble case based learning. First, the relevant multi-agent systems concepts for our research are introduced. Specifically, collaboration strategies are defined and a formalism to specify interaction protocols is presented. Then, *feature terms*, the formalism used to represent knowledge in our work, are presented jointly with the NOOS Agent Platform, a LISP based agent platform specifically

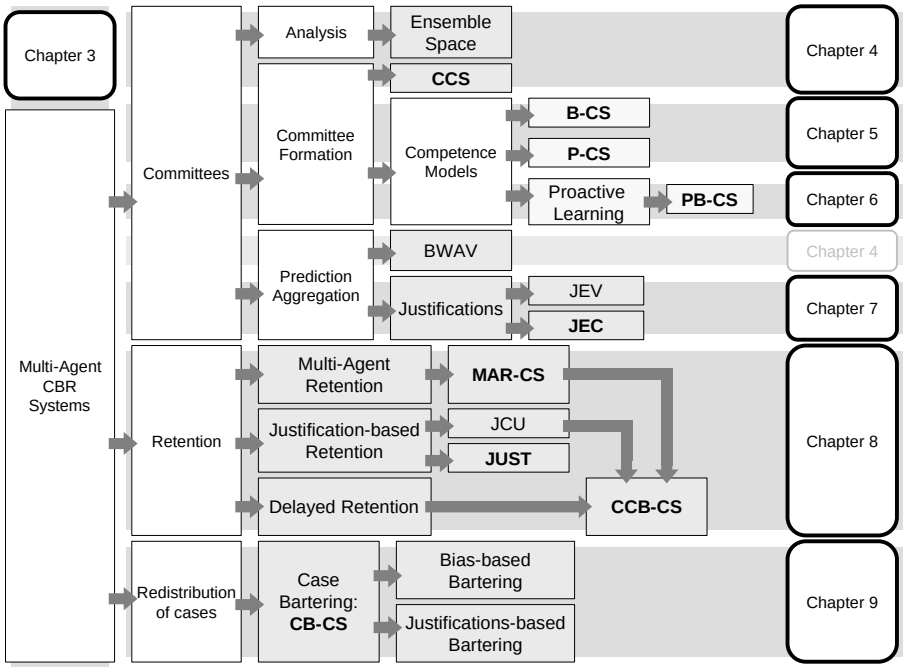


Figure 1.1: Graphical overview of the contents of this thesis.

designed to incorporate integrate learning and reasoning techniques using feature terms as representation language. Finally, the *Multi-agent Case Based Reasoning Systems* are formally defined and our approach to multi-agent learning is presented from a Case-Based Reasoning perspective.

- **Chapter 4** presents the concept of a *committee* (a group of agents that joins together to solve a problem using a voting system). A committee is the organizational form of an "ensemble of agents" from the point of view of multi-agent systems, defined to study the ensemble effect in multi-agent systems. Specifically, the *Committee Collaboration Strategy* with which a group of agents can act as a committee is presented, and the *Bounded Weighted Approval Voting* is introduced as a voting system specifically designed for committees of agents that use CBR to solve problems. Chapter 4 also presents the *ensemble space analysis*, an analytical tool to characterize committees of agents and that we will use in the rest of the thesis as a way to analyze the performance of a committee. Later, Chapters 5, 6, and 7 present extensions of the basic Committee Collaboration Strategy.

- **Chapter 5** presents the idea of the *dynamic committee* collaboration strategies, that are strategies that convene different committees of agents depending on the problem that has to be solved. Specifically, two different dynamic committee collaboration strategies, namely the *Peer Counsel*

Collaboration Strategy and the *Bounded Counsel Collaboration Strategy*.

- **Chapter 6** deals with how agents can learn to collaborate better. For that purpose we introduce *competence models*, functions that assess the likelihood of the solution provided by an agent (or set of agents) to be correct. Next, we present the *proactive learning* of competence models as a way in which individual agents can learn when to collaborate and with whom to collaborate. Competence models can be autonomously learnt by the agents interacting with other agents. Finally, the *Proactive Bounded Counsel Collaboration Strategy* is presented, combining dynamic committees with proactive learning.

- **Chapter 7** introduces the notion of *justification*. A justification is the explanation given by a CBR agent (or any other problem solving system) of why it has considered the solution of a specific problem to be correct. In this chapter, we use justifications to deal with the issue that it cannot be taken for granted that the agents in a $\mathcal{M}$AC system satisfy the preconditions of the ensemble effect. For that purpose, we will show that justifications can be examined by some agents to assess the confidence of the predictions made by other agents. Then, we will show how to use this information to define an aggregation mechanism to determine which is the solution with highest confidence, namely, the *Justification Endorsed Voting System* (JEV). Finally, we present the *Justification Endorsed Committee Collaboration Strategy* that uses JEV to improve the performance of committees by weighting the individual votes according to the confidence assessed to their predictions.

- **Chapter 8** addresses the issue of case retention in Multi-Agent Case-Based Reasoning Systems. Specifically, three ideas are introduced: collaboration strategies for case retention, the assessment of the case utility using justifications, and delayed retention. First, several collaboration strategies for case retention are presented, showing that they can outperform individual retention strategies. Then, we introduce the *Justification-based Case Utility* (JCU), a case utility function based on justifications that can assess how useful will be a case for an agent. Moreover, we present the *Justification-based Selection of Training Examples*, a case base reduction technique that uses JCU to generate a reduced case base with the same problem solving accuracy as the original one. Thirdly, we show that delayed retention can improve performance with respect to retention strategies that consider cases one by one. Finally, the *Collaborative Case Bargaining Collaboration Strategy* is presented as a retention strategy that combines the three ideas presented in the chapter.

- **Chapter 9** presents a new family of collaboration strategies that use the idea of *case bartering*. Case bartering is designed as a way to deal with the problem of finding a redistribution of cases among the case bases of the agents so that they perform better both as individuals and as a committee.

Specifically, we present two basic case bartering strategies: the *Bias Based Case Bartering Collaboration Strategy* and the *Justifications Based Case Bartering Collaboration Strategy*. The first one is inspired on the ensemble space analysis presented in Chapter 4, and is based on decreasing the bias in the individual case bases of the agents with the goal of boosting both the individual performance of agents and their ensemble performance. The second strategy is inspired in the case utility assessment based on justifications and allows each agent to obtain high utility cases with the goal of improving both their individual and ensemble performance.

- **Chapter 10** first summarizes the work presented in this thesis. Then, the contributions with respect to ensemble learning, case-based reasoning and multi-agent systems are presented. The chapter closes with a discussion of future lines of research.

- **Appendix A** presents a comprehensive list of the notation used in all the chapters of this thesis.

- **Appendix B** presents an overview of the NOOS agent platform, that we have used in our research.

- **Appendix C** presents a probability assessment technique used in the Proactive Bounded Counsel Collaboration Strategy (Chapter 5), and in the Justification-based Selection of Training Examples (Chapter 8). This technique determines a confidence interval for classification accuracy estimations.

1.5 Notation

In the remainder of this thesis we have followed the following notation conventions:

- A_i: is used for agents (different subindexes denote different agents).

- c_i: is used for cases.

- $\mathbf{R}$: boldface upper case letters are used for tuples (or records).

- $\langle f_1, ..., f_n \rangle$: angle-bracketed lists are also used for tuples (or records).

- $\mathbf{R}.f_i$: dot notation is used to refer to the value of the field f_i of the tuple $\mathbf{R}$.

- C: when elements of a certain kind are noted with a lower case letter, sets of such elements are noted with an upper case letter. For instance, since cases are noted with letter c, sets of cases — a.k.a. case bases — are noted with letter C.

- $\mathcal{A}$: when elements of a certain kind are noted with an upper case letter, sets of such elements are noted with a calligraphic upper case letter. For instance, since agents are noted with an upper case letter A, sets of agents are noted with a calligraphic letter $\mathcal{A}$.
- $\mathbb{A}$: when elements of a certain kind are noted with a calligraphic letter, sets of such elements are noted with a "blackboard bold" letter. Moreover, since elements noted with a calligraphic letter are usually sets, elements noted with blackboard bold letters are usually called "collections of sets" (for not using "sets of sets" that would be confusing).
- $\#(\mathcal{A})$: denotes the cardinality of the set $\mathcal{A}$.

Appendix A contains a comprehensive list of all the notation used throughout this thesis.

Chapter 2

State of the Art

In this chapter, we will present an overview of the state of the art in the areas related to our work. We have divided it into four areas:

- Ensemble Learning
- Case Based Reasoning
- Active Learning
- Multi-Agent Learning

The following sections describe each the work done in each one of the previous areas in detail.

2.1 Ensemble Learning

Usually, when we have a set of training examples and a set of classifiers trained from them, we have to select which is the best individual classifier. We can define several criteria, such as the classifier that minimizes the classification error in a test set. However, the bayesian learning theory tells us that if we want to minimize the classification error, the best that we can do is not to select a single classifier, we have to take into consideration all the possible classifiers and aggregate their results building an *ensemble* of classifiers where each individual classifier is weighted by its posterior probability given the training examples. This ensemble of classifiers is known as the *bayes optimal classifier*, and no other classifier can outperform it on average, since it maximizes the probability of a new problem to be classified correctly.

Obviously, it is not possible to make such an ensemble since the number of possible classifiers given a real problem is prohibitive. However, we can take a sample of all the classifiers and aggregate them hoping to reduce the classification error compared to the best individual classifier. That is exactly the idea behind *ensemble learning*. The classification error reduction obtained

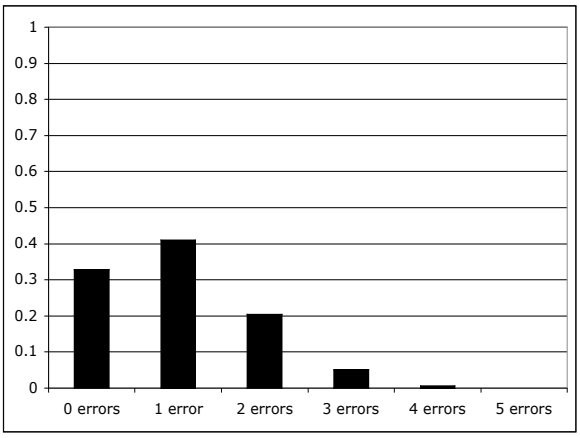


Figure 2.1: Probability of different agents erring at the same time.

by the ensemble methods (that we may call *ensemble effect*) is usually stated like this: the resulting error of the combined predictions made by several classifiers is lower than the error of the individual classifiers if the following conditions are met [36]:

1. The individual error of the classifiers is lower than 0.5

2. The individual error of the classifiers is uncorrelated (i.e. the individual classifiers don't err on the same problems).

The first condition states that the individual classifiers have to be minimally competent, and the second conditions means that the individual classifiers have to be different. Notice that if we combine identical classifiers, we do not gain anything, since the aggregated output will be identical to the output of any of the individual classifiers.

The error correlation between two classifiers A_i and A_j can be defined in the following way [34]:

$$C(A_i, A_j) = p(A_i(P) = A_j(P) | A_i(P) \neq S_P \vee A_j(P) \neq S_P)$$

where $A_i(P)$ represents the classification given by the classifier A_i for a problem P, and S_P is the true solution class of the problem P. Notice that the correlation lies in the interval $[0, 1]$, and the error correlation between a classifier and itself is 1. From now on, we will say that two classifiers are correlated if their individual errors are correlated.

2.1.1 Quantifying the Ensemble Effect

Let us consider why the ensemble effect works. Figure 2.1 shows the probabilities that a specific number of agents give incorrect answers in a 5 agents system where each individual agent has a probability of error 0.2 assuming that their predictions are not correlated. We see that the probability of all the agents answering the correct answer is $(0.8)^5 = 0.33$, the probability of a single agent making an error is $5 \times (0.2 \times (0.8)^4) = 0.41$, etc. Notice that if the individual agents work as a committee (i.e. when they want to solve a problem, every agent solves the problem, and the most voted solution is considered the final solution), it is not needed that all the individual agents solve the problem correctly, only a majority is needed. In this case with 5 agents, we only need that at least 3 agents predict the correct solution. If we compute the probability of the committee of 5 agents of predicting the correct solution, we obtain: $0.33+0.41+0.20 = 0.94$ (i.e. the probability of all the agents giving the correct solution plus the probability of 4 agents giving the correct solution plus the probability of 3 agents giving the correct solution). Therefore, just by aggregating the predictions of 5 individual agents whose individual error rate was of 20%, we have obtained a combined predictor with an error rate of only 6%.

Notice that to compute the benefits of the ensemble effect in the previous paragraph, we have assumed that the individual agents' predictions are not correlated and this is not true in general. Studying the benefits of the ensemble effect is specially relevant for multi-agent learning. Several studies have tried to quantify the benefits that can be obtained by the ensemble effect. For instance, Krogh and Vedelsby [49] found a decomposition of the error E of an ensemble in two terms: $E = \bar{E} - \bar{A}$, where $\bar{E}$ is the mean error of the individual classifiers and $\bar{A}$ is the *ambiguity*, a term that measures the degree of correlation (the less correlation, the more ambiguity). The obvious conclusion by looking at the formula is that the smaller the correlation, the smaller the combined error. Therefore, to obtain the smaller classification error, we need individual classifiers that have small classification error (to have a small $\bar{E}$), and that are not correlated (to have a great $\bar{A}$). Unfortunately, it is not possible to achieve both at the same time, and we have to find a tradeoff between individual accuracy and correlation. Other theoretical bounds on the benefits of the ensemble effect can be found in [47], [85] and [58].

2.1.2 The Basic Ensemble Method

The ensemble effect has motivated several methods for combining classifiers. The *Basic Ensemble Method* (BEM) was one of the first ones [67]. BEM is a basic way to combine predictions made with several neural networks by simply taking the average prediction among all the outputs of the different networks. To build the collection of networks, BEM proposed to train a set of networks with the same training data but using different initial weights in each network.

Assuming that the error of each individual network is not correlated with the error of the rest of networks, and that the errors of the networks have zero

mean in average, it is possible to prove that BEM reduces the mean square error by a factor of n (where n is the number of networks).

However, as these assumptions do not hold in general, an alternative formulation of the BEM called *Generalized Ensemble Method* (GEM) is also presented. GEM computes the best linear combination of all the networks in the ensemble. Notice that both BEM and the best single network are special cases of GEM, and therefore GEM will be always better or equal than BEM and than the best individual network.

2.1.3 Reducing Error Correlation

The previous methods (BEM and GEM) relied on the unstability of the neural networks in order to generate the different classifiers of the ensemble that were not correlated. However, to create ensembles based on other learning methods we need other ways to obtain individual classifiers that are not correlated.

Bagging was presented by Breiman [16] as a general method to build ensembles of classifiers with low error correlation. Bagging proposes to generate several training sets drawn at random from the original training set, and train each individual classifier with one of them. These new training sets are not disjunct, i.e. there are cases that are present in various training sets. The results of the classifiers are then averaged (as in BEM) for regression tasks or combined using a voting scheme for classification tasks.

Notice that Bagging works if by training two classifiers with a modification of the same training set, we can obtain radically different classifiers. However, this is not true for all the learning methods. In fact, learning methods can be grouped in two groups: *global* methods and *local methods*. Global methods are those that create a single and global model from all the training data (e.g. decision trees or neural networks). Global methods are very sensible to small changes in the training set, and therefore methods like Bagging work well with them. Local methods are those in which a small change in the training set do not affect the global model, but only a small area of the model (e.g. nearest neighbor methods, where a change in the training data only affects the area surrounding the changes). Zheng [16] and Bay [12] proposed a different way to achieve low error correlation among the classifiers based on making changes on the representation of the data instead of changing the training set. The changes in the representation can be achieved by allowing each individual classifier to have access only to a reduced subset of all the attributes of the training data. Using this method, local methods can also obtain benefits from the ensemble effect. Zenobi and Cunningham [33] analyze these kind of ensembles and define a strategy to build ensembles by creating individual classifiers minimally correlated based on training them with different feature subsets. They find that ensembles with low correlation work better than those ensembles in which the individual classifiers are selected only for their individual accuracy.

Finally, a third way to reduce error correlation is to modify the learning method itself, i.e. give each individual classifier a different learning method. This is applied for example by Ho [41] to the task of optical character recognition

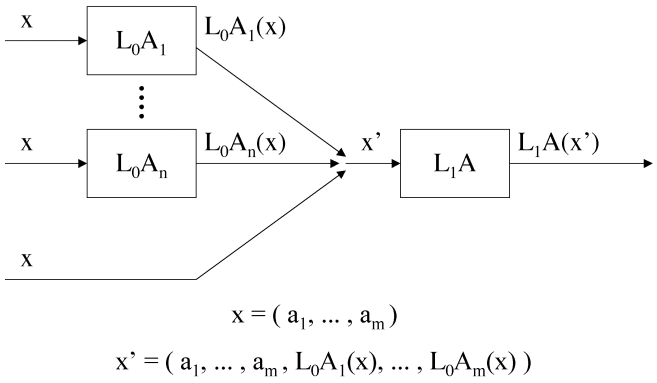


Figure 2.2: Stacked Generalization combination method.

and by Freitag [29] to the task of information extraction.

Summarizing, we have seen that there are three basic ways to reduce correlation among individual classifiers: changing the training sets, changing the representation of the data or changing the learning methods. In this monograph we will work with autonomous agents that have learned individually from different training sets (i.e. they have collected experience individually). The degree of correlation is an effect from the fact that the agents have individually collected their experience (i.e. cases). Therefore, we do not use any method to explicitly reduce the error correlation. We will assume that there is a given set of agents, with given case bases, and we will try to define strategies in which those agents can improve their performance with respect to working in isolation (sometimes reducing their error correlation, see Chapter 9).

2.1.4 Aggregating the Classifiers' Predictions

Until now, we have seen that the key in an ensemble to improve the classification accuracy is to have individual classifiers with low error correlation. However, there is another important issue: the way in which the individual classifiers' predictions are aggregated. BEM simply used the average of all the individual networks, and bagging uses the average or a simple voting scheme. However, more complex combination methods can be defined.

Stacked Generalization [87] was presented by Wolpert as a general method for combining classifiers. Given a set of classifiers $\{L_0A_1, ..., L_0A_n\}$, stacked generalization creates a *second level classifier* L_1A, that takes as input the outputs of the collection of base classifiers and learns to predict the correct solution. This second level classifier can be trained using the outputs of the base classifiers or by enlarging the set of available attributes in the original training data of the base classifiers with the predictions of the base classifiers (as shown in Figure 2.2).

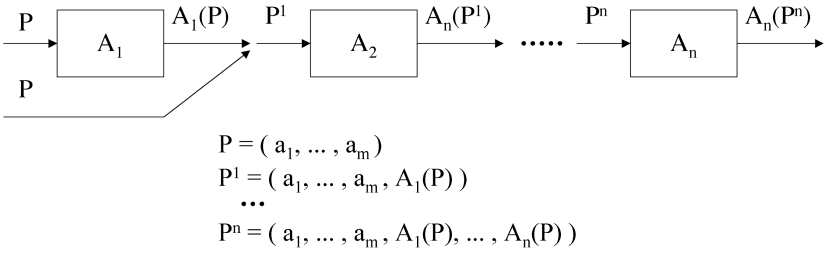


Figure 2.3: Cascade Generalization combination method.

Cascade Generalization [34] is presented by Gama as another general method for combining classifiers. Cascade Generalization is similar to Stacked Generalization in having a set of base classifiers but with the difference that they are run sequentially. At each step (i.e. after applying one classifier), the original data set is expanded with new attributes. These new attributes are the predictions given by the base classifiers as shown in Figure 2.3.

In general, both Stacked and Cascade Generalization fall in what it is called *Meta Learning*. Meta learning is introduced by Chan and Stolfo [19], who define three different combination strategies: the *combiner* strategy, the *arbitrer* strategy and the *hybrid* strategy. The combiner strategy is basically a reformulation of the stacking generalization, the arbitrer strategy proposes to generate a meta classifier that is considered also a base level classifier but whose output decides the solution class in case of a tie (when voting). The hybrid strategy is a mix of both, and proposes to train a meta classifier with just those training examples in which the base classifiers disagree and that will work as a combiner. Notice that cascade generalization can be seen as a sequence of combiner strategies with a single base classifier.

Finally, to complete the different ways to combine the predictions of several classifiers. There's a last category of methods based on *model selection* instead of model combination. In model selection methods, for each new problem that the ensemble has to solve, instead of aggregating the predictions of all the individual classifiers, the most reliable prediction for the given problem is selected. There are two basic ways to perform model selection: using a predefined rule to select among the classifiers or learning a specific one. Ting [84] defines a heuristic called *Typicality* that selects the most reliable classifier for each problem. Typicality can be defined as follows:

$$Typicality(x) = \frac{InterClassDistance(x)}{IntraClassDiscante(x)}$$

i.e. those instances that are very close to other instances in the same class and very far from instances of other classes are very typical, and those instances that are close to instances of other classes are atypical.

Another approach to model selection is taken by Koppel and Engelson [48]. For each base classifier, they propose to learn a meta classifier that is able to

predict whether the base classifier is likely to give a correct solution or not for a given problem. In fact, this meta classifiers learn the areas of expertise of the base classifiers. The most reliable base classifier's prediction (according to its meta classifier) is selected for each new problem.

This last family of combination methods (model selection) is specially interesting in a multi-agent setting as the one we propose since it can allow individual agents decide with which other agents in the system to collaborate by building models of the areas of expertise of them. In fact, Chapter 4 shows how to apply this technique in order to determine competent agent committees to solve specific problems.

2.1.5 Error Correcting Output Codes

We have seen that there are three ways to reduce error correlation (different training sets, different data representation and different learning algorithm) and three ways to combine the predictions (parallel combination, sequential combination and model selection). However, there is still another technique to build ensembles of classifiers: training each individual classifier to learn a problem different from the original learning task, but that when combined we can reconstruct a solution for the initial learning task.

For instance Fürnkranz [32] proposed to build a committee composed of individual classifiers where each classifier just learned to distinguish between two classes. If there are K classes, there will be $K \times (K-1)$ classifiers in the ensemble (one per each possible pair of classes). This approach is called *Pairwise Classification* or *Round Robin* classification.

A very related technique to *Pairwise Classification* is called the *Error Correcting Output Codes*) (ECOC), and was presented by Dietterich and Bakiri [24]. ECOC represent each solution class with a different codeword composed of several bits. Each bit of the codeword encodes a binary classification task corresponding to a unique partition of the classes. Each individual classifier of the ensemble learns to predict a specific bit of the codeword. The output class is selected as the class with the closest codeword to the codeword resulting of aggregating the bits predicted by the individual classifiers.

2.1.6 Boosting

Finally, *Boosting* was presented by Freund and Schapire [31], and has been one of the most successful ensemble methods. Boosting is a general method that attempts to "boost" the accuracy of any given learning algorithm. The pseudocode of *AdaBoost* (the most common boosting algorithm) is shown in Figure 2.4. As we can see in the figure, AdaBoost is an iterative algorithm with a predefined number of rounds N. At each round i, AdaBoost will train a new classifier A_i with the training set T_i. Moreover, each instance in T_i has a weight $T_i(k)$. The learning algorithm should give more importance to classify well those instances with higher weights. AdaBoost increases the weight of those instances incorrectly classified by the current classifiers in the ensemble. In that way,

Function AdaBoost ($\mathcal{T}$, N)
    Initialize $\forall_{k=1...\#(\mathcal{T})} T_1(k) = 1/\#(\mathcal{T})$.
    For i = 1, ... , N:
        Train a classifier A_i with the distribution T_i.
        Compute the error e_i of the classifier A_i.
        $w_i = e_i/(1 - e_i)$.
        Compute the new distribution: $T_{i+1}(k)$:
            Increase the weight of the failed instances.
            Decrease the weight of the correctly solved instances.
    EndFor Return Ensemble(P) = $\sum_{i=1...N} w_i A_i(P)$.
end-function

Figure 2.4: The AdaBoost algorithm. $\mathcal{T}$ is the training set, and N is the predefined number of classifiers.

the new classifiers created will focus in those areas of the problem space where the current members of the ensemble fail to find the correct solution. Finally, AdaBoost assigns a weight w_i to each classifier of the ensemble computed as a function of its classification error. Once the algorithm has finished, in order to classify a new problem P all the classifiers solve P and their predictions are aggregated using a weighted mean. AdaBoost assumes that the classification task is binary (i.e. there are only two classes). However, it can be extended to solve multiclass problems and there are multiple algorithms proposed such as AdaBoost.M1 [30], AdaBoost.M2 [30], AdaBoost MH [74] or AdaBoost MR [74].

Notice that since AdaBoost uses changes in the training set to reduce correlation, it will only work with global methods and not with local methods.

The weak learning methods most often used with AdaBoost are *decision stumps*. A decision stump is simply a decision tree consisting in a single node. Decision stumps can be binary or bayesian, i.e. they can simply output a solution class label (positive or negative) or output a solution class probability.

In addition, notice the parallelism between AdaBoost using bayesian decision stumps and the naive bayesian classifier: both algorithms create a collection of individual classifiers where each one predicts the solution class using a single attribute. The difference lies just in the way in which those individual classifiers are created, and the way in which their predictions are aggregated. Therefore, we can consider naive bayes as one of the first ensemble methods.

2.1.7 Summary

We have presented several ensemble methods, ad we can conclude that there are only four different ways of reducing correlation:

- Varying the training set,

- Varying the representation (different feature subsets),
- Varying the learning method, and
- Varying the learning task (pairwise classification or ECOC).

and three different ways of combining multiple classifiers:

- Parallel combination,
- Sequential (cascade) combination, and
- Model selection.

Finally, we have presented Boosting, an ensemble algorithm that uses variations in the training set and parallel combination to achieve the maximum possible accuracy given a learning algorithm and a training set.

In our experiments, the degree of error correlation among the individual agents will be given by the fact that they have learned individually, and that each one has collected experience on their own (i.e. they will have learned from different training sets). Moreover, we will study parallel combination and model selection techniques in order to aggregate the predictions of the individual agents.

One of the main differences between our approach and the ensemble methods explained in this section is that, in our approach, we assume that each agent is autonomous and that there is no centralized algorithm that can control them and that the case bases of the individual agents are given (and not generated from a single training set as in most ensemble methods). In the ensemble methods such as boosting and bagging, there is a central algorithm that has access to the entire data set, and that *creates* the ensemble, distributing the training set among the individual classifiers and defining the way in which they collaborate. Moreover, in our approach, no agent can be forced to collaborate with any other agent if the agent is not willing to. The agents will only collaborate with other agents if it is beneficial for them. Therefore, we can say that the existing ensemble methods in machine learning study how to *build* ensembles, while we focus on how can a *given* set of individual agents (classifiers) improve their performance by working as an ensemble.

2.2 Case Based Reasoning

Case Based Reasoning (CBR) is a specific kind of *lazy learning*. In contrast to learning methods that construct a general, explicit description of the target function when training examples are provided, learning in lazy learning methods consists of simply storing the training examples. Lazy learning techniques construct *local approximations* to the target function that in most cases only apply in the neighborhood of the problem, and never construct an approximation designed to perform well over the entire problem space [63].

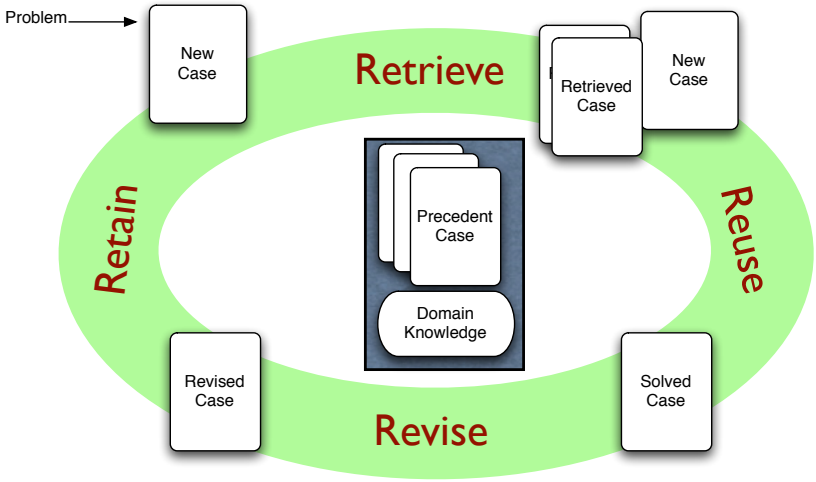


Figure 2.5: The Case Based Reasoning Cycle [1].

The activity of case based reasoning can be summarized in the CBR cycle [1] as shown in Figure 2.5. The CBR cycle consists of four stages: Retrieve, Reuse, Revise and Retain. In the *Retrieve* stage, the system selects a subset of cases from the case base that are relevant to the current problem. The *Reuse* stage adapts the solution of the cases selected in the retrieve stage to the current problem. In the *Revise* stage, the obtained solution is examined by an oracle, that gives the correct solution (as in supervised learning). Finally, in the *Retain* stage, the system decides whether to incorporate the new solved case into the case base or not.

Specifically, in this section we are going to focus in the Revise and Retain stages. For the Revise stage, we are interested on techniques that allow a CBR agent to build an explanation (to be presented to an human expert or to another CBR agent) that endorses the solution of the new case. For the Retain stage, we are interested in case retention strategies that permit a CBR agent to select which cases to store in its local case base in order to obtain a compact competent case base (i.e. a small case base that allows the agent to solve the largest range of problems possible). Related to case retention is the machine learning subfield of *active learning*. Active learning focuses on learning systems that have some kind of control over which examples to learn from.

In the remainder of this section we are going to present an overview of these two areas: selection of training examples (that includes both case retention and active learning) and explanation generation.

2.2.1 Selection of Training Examples

There are two main approaches to the selection of training examples (or cases): the CBR approach with the Retention process, and the active learning approach. In this section we will present the most relevant work to our work in both approaches.

2.2.1.1 Case Retention

While the goal of active learning is to avoid the labelling cost, CBR retention focuses only on minimizing the size of the case base of a CBR system without diminishing its performance. In earlier CBR systems, where small case bases where used, it was found that the performance of the system increased as new cases were added to the case base. However, in modern CBR systems, with large case bases, it has been found that adding new cases into the case base is not always beneficial. Smyth and Cunningham [77] analyze this problem and find that although lazy similarity-based methods (such as the ones typically used in CBR) do not usually suffer from overfitting when adding new cases into the case base, the efficiency of the system can degrade. The efficiency of the system can be divided in two factors: the retrieval time and the reuse time. While reuse time diminishes as the case base grows, retrieval time increases with case base size. Therefore, by adding new cases into an already saturated cases base, we only get the same problem solving performance, but with a reduced efficiency.

There are two different approaches to CBR case retention:

- Case addition: where the system decides whether to add a new case or not to the case base each time that a new case is received.
- Case deletion: the system always adds new cases to the case base, and when a determined size is reached (or when an efficiency measure reaches some threshold), some cases are decided to be deleted.

In the following we are going to present several strategies that fall in one of these two approaches. However, several strategies can be used as case addition or as case deletion strategies indistinctly.

Aha, Kibler and Albert [3] and later Aha [2] propose several algorithms for case based learning systems: CBL1, CBL2, CBL3 and CBL4. For our interests, the comparison of CBL1 and CBL2 is specially interesting (CBL3 and CBL4 only add robustness to noise and to irrelevant features):

- CBL1: This is the base level algorithm, where all the cases are added to the case base.
- CBL2: a case is added to the case base only it the system is not able to solve it.

The intuition behind CBL2 is that if the system is able to correctly solve a problem, that problem will not add any valuable information to the case base.

```
Algorithm CNN ( C )
    B = ∅
    changes = true
    While changes do
        changes = false
        For each c ∈ C do
            If the system cannot solve c with the cases in B them
                changes = true
                Add c to B
                Remove c from C
            EndIf
        EndFor
    EndWhile
    Return B
EndAlgorithm
```

Figure 2.6: The Condensed Nearest Neighbor (CNN) algorithm.

CBL2 is able to drastically reduce the amount of cases stored in the case base, but suffers from several problems: it is very sensible to noise and to the order in which the system receives new cases to be retained. CBL3 solves the noise problem, but is still sensible to the order in which the new cases are presented to the system. Moreover, notice that CBL2 is not an active learning strategy, since it assumes that the correct solution of a case is known before deciding whether to retain it or not.

Related to CBL2, Hart [37] presented the *Condensed Nearest Neighbor* (CNN), that tries to obtain a smaller subset B of cases from a case base C with the same problem solving power. CNN is outlined in Figure 2.6. Notice that CNN is basically an algorithm that applies CBL2 with the cases in C that are still not in B until all the cases in C can be solved with the new case base B.

Smyth and Keane [78] define several competence metrics for cases inside a case base that they later use to define case retention policies. Specifically, they define the following concepts:

- The *coverage* of a case c in a case base C as the set of cases $B \subseteq C$ that can be solved successfully using c.
- The *reachability set* of a case c is the set of cases $B \subseteq C$ with which the case c can be successfully solved.

With these two concepts, they define 4 kinds of cases: *pivotal* cases are those cases that whose reachability set is empty (i.e. those cases that cannot be solved by any other case in the case base), *spanning* cases are those cases that link together areas of coverage, *support* cases are a special case of spanning cases

Algorithm CaseBase Reduction (C)
    Compute the Coverage of all the cases in C
    Set $B = \emptyset$
    While $\#(B) < k$ do
        Select the case $c \in C$ with maximum coverage and add it to B
    EndWhile
EndAlgorithm

Figure 2.7: The Case Base reduction algorithm presented by Zhu and Yang.

and finally *auxiliary* cases are those cases whose coverage is subsumed by the coverage of one of the cases in their reachability set (i.e. auxiliary cases do not contribute at all in the coverage of the system). They propose a case deletion policy called *FootPrint deletion* consisting on deleting first the auxiliary cases, deleting only support cases if there are no auxiliary cases, only delete spanning cases if there are no support cases, and only delete pivotal cases if all the cases in the case base are pivotal.

Smyth and McKenna [79] try to solve one of the shortcomings of the Condensed Nearest Neighbor (CNN) algorithm. As we have said, CNN is very dependent of the order in which the cases are presented to the system. Smyth and McKenna define an ordering criterion based on an improvement over the competence definitions done by Smyth and Keane in [78]. They define the concept of *relative coverage*, that measures the contribution to the coverage of a single case (taking into account how much of the coverage of a case is shared by its neighbors). They propose the RC-FP (Relative Coverage Footprint) algorithm, that basically is the CNN algorithm but ordering the cases of C in descending order of relative coverage before presenting them to the system. In that way, cases with higher coverage are presented earlier to the system and have higher chances of being retained. In a later work, Smyth and McKenna [62] expand their family of retention algorithms with: RFC-FP (that sorts cases in ascending order of the reachability set), COV-FP (that sorts cases in descending order of the coverage set).

Leake and Wilson [54] have also expanded the family of CNN related strategies presenting RP-CNN, that works as CNN but using a measure called *Relative Performance* (RP) to sort the cases before presenting them to the CNN algorithm. Relative performance is an utility measure that takes into account the adaptation cost, i.e. the cost of adapting the solution of a case to solve a problem.

Zhu and Yang [90] propose a dual approach to that of Smyth and Keane based on case addition instead of deletion. They formulate the following problem: we want to obtain a subset of cases B of size k from a case base C with the maximum coverage. To solve that problem, they propose algorithm shown in Figure 2.7. They prove that this algorithm produces a new case base that

has at least a coverage of the 63% of the optimal reduced case base, while the original algorithm of Smyth and Keane cannot ensure that result.

Salamó and Golobardes [57] propose two deletion policies for CBR systems based on the rough set theory: *Accuracy-Classification Case Memory* ACCM and *Negative Accuracy-Classification Case Memory* NACCM. Both ACCM and NACCM are able to keep classification accuracy at a very high level (sometimes even improving the accuracy of the complete case base), but the reduction of cases obtained is not as big as algorithms such as CBL2. However, case bases obtained using CBL2 obtain much lower accuracies than those obtained with ACCM or NACCM.

2.2.1.2 Active Learning

Active Learning is the subfield of machine learning that studies strategies for a learner to select which examples are added to the training set to learn from. In contrast to active learning, we have *passive learning*, that is the traditional learning from examples framework where a learner simply accepts all the examples in the available training set. The goal of active learning is to minimize the labelling cost, i.e. active learning focuses the scenario where it is cheap to obtain unlabelled examples, but it is expensive to label them. This corresponds to many real life scenarios where labelling an example implies asking a human expert or running an expensive experiment in a production plant, etc. Specifically, there are two types of active learning:

- Example construction: the learner can *construct* examples, and then ask to a teacher/oracle to label them.
- Example selection: it is assumed that the learner cannot construct examples, but that has access to a large set of unlabelled examples. The learner will examine the accessible unlabelled examples, and only select those that appear to be the most informative. Then the learner asks the teacher/oracle to label those examples.

Many strategies have been proposed for both types of active learning. For instance, Cohn, Atlas and Ladner [20] propose a technique called *selective sampling*. Selective sampling requires the definition of a *region of uncertainty*. The region of uncertainty is the region of the problem space where the solution of the problems is still not defined given the known training data. Figure 2.8 shows an example of the region of uncertainty for the version space algorithm. Selective Sampling is an example construction active learning technique that proposes to construct examples only from inside the region of uncertainty, since that region is where the problems that can make the learner reduce its error lie all inside the region of uncertainty.

Selective sampling is able to improve the effectiveness of the learning process given that the region of uncertainty can be computed. However, it is not easy to compute the region of uncertainty for all machine learning methods. Cohn, Atlas and Ladner also define a method to approximate selective sampling in

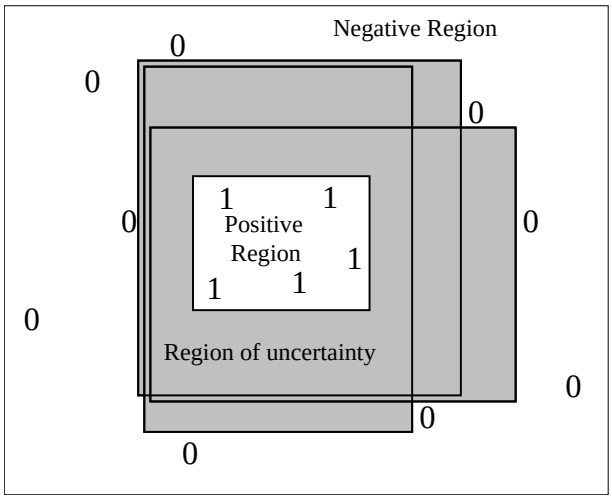


Figure 2.8: Region of uncertainty defined for a version space learning algorithm.

neural networks by assuming that the area where the output of a network is in the interval $[0.1, 0.9]$ is an approximation of the region of uncertainty. However, that approximation only works once the network has been trained with an initial (and representative) sample of examples.

Seung, Opper and Sompolinsky [76] propose that the Shannon information of an example can be a good active learning criterion and they propose an algorithm called *Query by Committee* to estimate that information value. Query by Committee works as follows: assume that the learner has already a training set T; then k subsets of T are drawn randomly, and k different classifiers are trained with each one of these subsets. Finally, the new example is classified with each of the new classifiers: the degree of disagreement among them is an approximation of the Shannon information of that example (given that the learning task is binary). Moreover, they make experiments with a concrete version of Query by Committee consisting on creating only 2 classifiers, and selecting those examples for which the two classifiers disagree. They show that the information content of each new example retained is constant, while in passive learning the information content of each new examples approaches zero as the number of examples increase.

Krogh and Vedelsby [49] define an active learning policy for ensembles of classifiers. They decompose the error of the ensemble in two terms: $E = \bar{E} - \bar{A}$ (as explained in Section 2.1.1). The ambiguity $\bar{A}$ is a measure of disagreement similar to that defined by Seung, Opper and Sompolinsky. Krogh and Vedeslby propose to measure the ambiguity of the ensemble for each new example, and add to the training set that with the highest ambiguity.

Finally, Argamon-Engelson and Dagan [7] propose another method for active learning based on *disagreements*. They propose a modified version of Query

by Committee where the entropy of the distributions of classifications voted for by the committee members is used as the active learning criterion. The entropy takes value one when all the committee members *disagree*, value zero when all agree, and intermediate values otherwise. They propose two policies for active learning: the *thresholded selection* and the *randomized selection*. In the thresholded selection, an example is selected if its entropy value is above a threshold, and in the randomized selection the entropy value is used as the probability to include the example into the training set.

Cohn, Ghahramani and Jordan [21] propose an active learning criterion based on the *bias and variance* [46] decomposition. The bias and variance decomposes the error made by a learner in three terms: the intrinsic error of the task (that cannot be eliminated), the learner bias, and the learner variance. As they work with probabilistic classifiers, the bias term is negligible (since the probabilistic classifiers being used can be considered unbiased). Therefore, the only term that can be decreased to reduce the classification error is the learner variance. Thus, they propose to select those examples that minimize the learner variance. The variance resulting of adding a new example to the training set is computed, and the example that minimizes the variance is selected. Moreover, although computing the variance term is expensive for many learning algorithms, they show it is accurate and not as expensive for statistical methods such as *mixture of gausians* and *Locally Weighted Regression*.

Lindenbaum, Markovich and Rusakov [55] propose that the active learning process can be represented as a game where there are two players: the learner and the teacher. The learner selects an example, and the teacher labels it. This defines a game tree. If we define that the active learning process finishes when a given number m of examples have been added to the learner's training set, the tree has depth $2n$. At each leaf, the learner has obtained a different training subset from the whole space of examples. The goal of a good active learning strategy is to select those actions for the learner that lead to the leaf where the expected classification accuracy of the learner is maximal. They propose a 1-level lookahead search algorithm to deal with active learning for nearest neighbor classifiers. Their method is more accurate, but has an increased cost: while a standard active selection active learning strategy has a cost N (where N is the size of the set of available unlabelled examples), the 1-level lookahead has cost N^2.

2.2.2 Explanation Generation

Explanation is a key feature of CBR systems that can increase their reliability. If a CBR system cannot provide an explanation of its answers, users can doubt of the results provided by the system. Wooley [88] makes an overview of explanation methods for expert systems. Wooley talks about the following uses of the explanations:

- Help systems used to describe how to use a specific product or command
- To increase user confidence in the results created by the systems

- Training or tutoring systems

Notice that explanation generation is always thought as a user-computer interaction, i.e. the explanations are generated to be shown to a human user. In our work, we are interested in agent to agent interactions, so that our CBR agents will generate explanations to be shown to other agents. Anyway, this section will provide a small overview of some existing techniques for explanation generation.

Cawsey [18] presents a model of explanation generation for tutoring systems based on explanatory dialogues. Cawsey develops a model of dialogues where the system has the dominant role. The system can ask questions to check the user's knowledge, and the user can interrupt the system with clarification questions.

Karsenty and Crezillion [44] present a framework for cooperative problem solving and explanation. They argue that explanations must be conceived of as part of a cooperative problem solving process, and not just as a parallel phenomenon that does not modify the course of the reasoning. They claim that usually the expert knowledge in expert systems is compiled, but that in order to be able to explain its results, and expert system should make explicit all his knowledge. Moreover, Karsenty and Crezillion also view the explanation process as a dialogue between the human user and the computer as Cawsey does, where both the user and the computer can ask questions to each other in order to clarify the explanation or to know knowledge leaks of each other.

Goel, Gómez, Grué, Murdok and Recker [35]use a Task-Method-Knowledge (TMK) language to represent explanations. They present a system called KRITIK, that can provide TMK explanations of the reasoning process, including the retrieved cases from the case base, the adapted solution and the intermediate steps in the adaptation process. The TMK language allows the system to provide a general explanation of the complete process, that the user can "unfold" to know more about specific issues.

Also related to explanations is the work on Explanation-Based Learning (EBL) [22]. EBL tries to imitate the human learning capability of generating new operational knowledge from domain knowledge and a single example e. The goal of the system applying EBL is to prove that an example e belongs to a particular concept c. This prove is performed using domain knowledge, and the trace of the prove is considered the *explanation* of why e belongs to that concept c. Once the explanation has been generated by the system, it can be generalized in order to cover more examples (similar to e, but not exactly equal). The purpose is to generalize as much as possible in order to cover a larger amount of examples while maintaining the proof valid. This explanation can be used in the future to determine if new examples belong to the concept c. Notice that a property of the generalized explanations generated by EBL is that they must be *operational*, i.e. the generalized explanation must be expressed using only operational predicates. The main difference of EBL and inductive learning is that EBL is *deductive*, i.e. the generalizations build using induction may or may not be correct, but a generalization built by EBL is always correct (assuming domain knowledge is correct). Moreover, it could happen that domain knowl-

edge is not complete, and therefore the system cannot always build a proof. In EBL, the generalizations are used in order to obtain "practical" knowledge from domain knowledge that can be used to solve problems, or to obtain knowledge that can speed up the problem solving process. This is the main difference with our work on explanations, since in our work the explanations are used to allow an agent to revise the solution obtained for a given problem by some other agent by examining the explanation.

In our work, we are interested in a specific kind of explanations (that we will call *justifications*). Namely, we are interested in symbolic justifications generated by CBR agents to justify the solutions that the agent find to solve problems. Therefore, we are interested in CBR methods that can provide such symbolic justifications of the solutions. The LID(Lazy Induction of Descriptions) method, presented by Armengol and Plaza [9], is a CBR method that uses a symbolic similarity term that can be used as a justification. LID builds the symbolic similarity term while solving a problem P in the following way: initially, LID creates a void similarity term $\mathbf{J}$; then the most relevant feature f of the problem P is selected, and added to $\mathbf{J}$. The set of all the cases in the cases base that have the same value in the feature f as the the problem P are called the *discriminatory set*. Once the discriminatory set is computed, LID selects the next most relevant feature of P, adds it to $\mathbf{J}$ and reduces the discriminatory set by removing the cases not having the same value than P in the feature f. This process continues until all the cases in the discriminatory set belong to a single solution class or until all the features in the problem have been used or are irrelevant. Therefore, when LID solves a problem, builds a symbolic similarity term ($\mathbf{J}$), that contains all the relevant features contained both in the problem to solve and in the cases retrieved to solve the problem. LID uses a heuristic measure to determine which are the most relevant features to solve the problem; thus, the symbolic similitude term contains all the *relevant* information common to the problem and to the retrieved cases (the discriminatory set). Therefore, a symbolic similitude term $\mathbf{J}$ returned by LID after classifying a problem P in a solution class S_k can be seen as a justification of why LID has considered that the correct solution class for P was S_k.

2.2.3 Summary

In this section we have presented the related work of Case Based Reasoning and Active Learning to our research. First, we have presented the active learning and case retention techniques, that can be summarized as follows:

- Active Learning: tries to minimize the labelling cost of adding new examples to the training set. There are two types of active learning:
  - Example construction: the learner generates examples and asks for its label to the teacher.
  - Example selection: the learner selects examples from a pool of unlabelled examples.

- Case Retention: focuses on reducing the case base size in order to increase the efficiency of the system, but without diminishing its problem solving performance (i.e. tries to build competent and compact case bases). There are two types of case retention strategies:
  - Case addition strategies: most of them based on the CBL2 algorithm.
  - Case deletion strategies: most of them based on the CNN (Condensed Nearest Neighbor) algorithm.

In our work, we will present case base retention strategies for multi-agent CBR systems inspired both in active learning and case retention techniques. Specifically, we will present strategies inspired in example selection active learning, and in addition/deletion case retention strategies.

Moreover, we have also presented work related to explanation generation. Most of the existing work on explanation generation focuses on generating explanations to be presented to a human user. However, in our research, we are interested on CBR agents that can provide explanations to be presented to other CBR agents. We will call *justifications* to such explanations. Moreover, as in some of the explanation generation techniques presented here, justifications will play a main role in cooperative problem solving (being the novelty that we work in cooperation between CBR agents and not between a CBR system and a human user).

We have seen that explanations are also used in EBL in order to obtain new domain knowledge. The main difference between the use of explanations (or justifications) in EBL and in our framework is that in EBL the explanations are used to expand the domain knowledge while in our framework we use them to allow an agent to revise the solution obtained by some other agent for a problem.

Finally, we have also seen that there are CBR methods, such as the LID method, that are able to generate a justification of their answers. In our work we are going to extensively make use of this kind of justifications in order to improve the agent interaction in multi-agent CBR systems.

2.3 Multi-Agent Learning

Multi-agent learning is the intersection between multi-agent systems and machine learning. In fact, we can say that any system where we have learning and multiple agents is a multi-agent learning system. However, in the literature multi-agent learning is usually not used to designate any multi-agent system where learning is used but is restricted to that kind of learning that is possible only because several agents are present. Moreover, multi-agent learning is a broad area and contains several subfields. An overview of learning in multi-agent systems can be found in [81], where Stone and Veloso make a classification of multi-agent systems by increasing order of complexity and explain the opportunities for applying machine learning techniques to each kind of multi-agent systems.

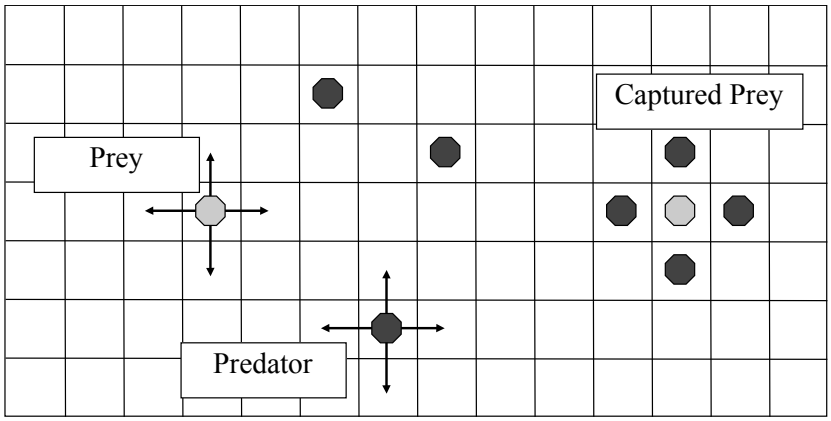


Figure 2.9: The predator prey domain.

In this section we will present an overview of only the main subfields of multi-agent learning that are related to our work. Namely, we will talk about multi-agent reinforcement learning (that is the most common machine learning technique applied to multi-agent systems), genetic algorithms and CBR (Case Based Reasoning). Moreover, we will also mention some isolated works that are not contained in any of these subfields but are particularly interesting for our work, i.e. those related to learning to solve classification tasks in multi-agent systems.

2.3.1 Multi-Agent Reinforcement Learning

One of the main questions in multi-agent learning is whether individual agents can benefit from collaborating with other agents in the system to increase their performance, i.e. whether collaborative agents learn faster or better than isolated agents. Tan [83] tried to answer this question by experimenting with reinforcement learning in multi-agent systems. Specifically he performed experiments in the predator-prey scenario (also known as the pursuit problem), where there are several predators with a single prey in a grid world and the goal of the predators is to capture the prey (as shown in figure 2.9). Tan defined three ways in which the learning agents can cooperate: sharing perceptions, sharing episodic information (i.e., perception, action and reward) and sharing learned knowledge. A group of agents that share perceptions has a better view of the environment since what an agent cannot see maybe is observable by some other agent, and therefore the learning process may speed up. When a group of agents share episodic information or learned policies, all of them can potentially converge to the same policy (since they all have the same episodic information from where to learn), and the learning time is reduced by a factor of n (where n is the number of agents in the system). Moreover, Tan explored what happens when the individual agents have to learn to perform a joint task, i.e. a task

where several agents must coordinate their actions to reach a common goal. The result was that the agents taking into consideration other agents (and that shared perceptions and episodes) learned better than independent agents.

Another early multi-agent reinforcement learning work is that of Mataric [59], who used reinforcement learning to learn social rules for a group of autonomous robots. She defined three kinds of reinforcement: individual reinforcement (individual progress to goal), imitation reinforcement (if other agents do the same actions that I do, that's a positive reinforcement), and partner reinforcement (if an action produces a direct positive reinforcement in another agent, this is also a positive reinforcement). Using these three types of reinforcement, the agents were able to outperform other agents that used only individual reinforcement.

Both Mataric and Tan did not pay attention to the essential difference between single agent learning and multi-agent learning and applied standard reinforcement learning techniques to the multi-agent setting. The main theoretical difference was not stated until the work of Littman [56]: In a single agent scenario an agent can apply reinforcement learning hoping to converge to an optimal policy since the environment is stationary. However, when there are other agents in the environment (that can also be applying learning), the optimal policy depends on the behavior of the other agents and the environment is not stationary anymore. Therefore, the convergence property of single agent reinforcement learning does not apply. To overcome this problem, Littman [56] proposed to model multi-agent reinforcement learning as Markov games instead of using Markov decision processes (as it is in single agent scenarios). Markov games extends the formalism of reinforcement learning to include multiple agents with interacting or competing goals. Moreover, Littman proposed an algorithm called minimax-Q in order to find optimal policies in multi-agent scenarios. However, Littman's work is restricted to zero-sum games.

Hu and Wellman [42] expanded Littman's framework to general-sum games. The notion of optimal policy loses sense in this more general framework, and Hu and Wellman focused on finding Nash equilibriums [65]. In a Nash equilibrium, each agent's choice is the best response to the other agents' choices.

Sen and Peng [11] also improved the minimax-Q algorithm by proposing the minimax-SARSA that performs better than minimax-Q in general-sum games.

Bowling and Veloso [14] defined two properties that multi-agent reinforcement learning algorithms should met: rationality and convergence. Rationality requires that if the other agents' policies converge to stationary policies, then the learning algorithm should converge to a policy that is the best response to the other agent's policies; in addition, convergence requires that the learner policy converges to a stationary policy. Bowling and Veloso analyzed some of the previous multi-agent reinforcement learning approaches and concluded that no previous algorithm (including minimax-Q and variants) met both properties at the same time. Then, they presented the WoLF principle (Win or Learn Fast) and an algorithm that uses it: WoLF-PHC (a hill climbing algorithm). WoLF is a method for changing the learning rate of an algorithm to encourage convergence. They proved that using WoLF, a rational algorithm converges for

a restricted class of iterated matrix games. Moreover, they empirically showed that WoLF-PHC is rational and converges on a set of stochastic games. Bowling and Veloso also have designed GraWoLF [13], a WoLF based algorithm that uses gradient ascend specifically designed for multi-robot learning. Moreover, Banerjee and Peng [10] proposed PDWoLF, an alternative version of WoLF that dominates WoLF in performance (i.e. PDWoLF converges faster). Another algorithm that tries to improve the convergence speed of is EXORL, presented by Suematsu and Hayashi [82] and based on the extended optimal response principle.

Finally, Tumer et al. [86] focus in a completely different problem of multi-agent reinforcement learning: the "alignedness" of the reinforcement functions of the agents with their goal. In problems where the solution is a sequence of actions, individual actions have individual reinforcements: if the positive reinforcement only arrives at the end of the sequence, the problem can be difficult to learn. However, if the reinforcement of the individual actions is also positive when the sequence of actions is leading the agent to the goal, we can say that the individual action reinforcements are *aligned* with the goal. In multi-agent settings this is even worse, since actions from several agents may have to be combined in order to achieve a collective goal. Tumer et al. used de Collective Intelligence (COIN) framework to address the problem of designing reinforcement functions for the individual agents that are both aligned with the global goal and that are learnable (i.e. the agents can see how their behavior affects their reward).

2.3.2 Genetic Algorithms for Multi-Agent Systems

Genetic algorithms have been extensively used to autonomously evolve behavior in agents. The work of Steels [80] to evolve new functionalities in autonomous robots is an example. Moreover, there are also a number of authors that have studied how to evolve coordination among autonomous agents using genetic algorithms. Haynes et al. [40] is an example of such a work where they apply a particular genetic programming technique, Strongly Typed Genetic Programming (STGP), to learn coordination strategies in the predator/prey domain. STGP is a technique to restrict the search space, by using it the resulting strategies are more human-readable than the strategies resulting from standard genetic programming. In an extension of that work Haynes and Sen [39] address competitive co-evolution, where groups of agents with competing goals evolve at the same time.

Competitive co-evolution was studied in detail by Rosin [72], where two populations with conflicting goals are evolved (the "hosts" and the "parasites"). Another work in co-evolution is that of Matos et al. [60], where populations of agents (buyers and sellers) evolve different negotiation strategies to maximize individual benefit.

Summarizing, we can say that genetic algorithms can be directly applied to all those multi-agent scenarios where just an appropriate set of parameters for each individual agent must be obtained (as in the case of the negotiation

strategies of Matos et al.[60]). However, if we want to evolve non-parametric strategies (i.e. evolve programs), we need to use some technique (such as STGP) to reduce the search space in order to obtain good results.

2.3.3 Multi-Agent Case Based Reasoning

Case Based Reasoning has also been applied in multi-agent systems to solve some scenarios where reinforcement learning or genetic algorithms are typically used. For instance, Haynes et al. [38] applies CBR to the predator prey problem where they allow each predator to learn cases of the behavior of other agents. Using the stored cases, a predator can predict the movement of the other predators, and coordination is greatly improved.

The first work in multi-CBR systems was presented by Prasad, Lesser and Lander [71]. They focus on a system where a set of individual agents have collected experience by their own, and thus can have a local view of each problem (i.e. each agent is just interested in a subset of features of each case, and has just stored them). When an external user (or agent) wants to make a query, maybe there is no single agent that has all the information required. Prasad, Lesser and Lander present a decentralized "negotiated case retrieval" technique that allows the group of agents to retrieve the appropriate information from each individual case base and aggregate in order to answer a query.

Another perspective for cooperative CBR is the *Federated Peer Learning* (FPL) framework presented by Plaza, Arcos and Martin [69]. In FPL each individual agent uses the same representation for the information (therefore, no translation phase is needed), each individual agent is able to completely solve problems but as each agent has collected experience (i.e. cases) individually, each agent may be specialized in some area of expertise. The difference with the work of Prasad, Lesser and Lander is that Plaza, Arcos and Martin assume that each individual agents has a complete view of each problem. Moreover, they explain two modes of cooperation in FPL: DistCBR and ColCBR. In DistCBR an agent can send problems to other agents to be solved by them, and each agent that receives a problem from another agent solves it using its individual knowledge and problem solving methods. In ColCBR, an agent can send problems to other agents and specify the concrete solution method that has to be used. Notice that in ColCBR other agents are just an extension of the agent's memory. In our work, we will also follow the FPL framework but with the difference that in FPL previous work, the retrieval is performed in a distributed fashion while in our work the agents perform individual retrieval and it is the reuse stage that is performed in a distributed way, i.e. in our framework, each agent will individually solve a problem using only their local case bases, but then, the all the solutions found by a group of agents will be aggregated to obtain a final solution.

Soh and Luo [45] deal also with a system composed of several individual agents where each agent owns a private case base in a very similar framework to FPL. When the case base of an agent does not contain any good case to solve a given problem, maybe there's another agent's case base that contains

it. Soh and Luo propose a combination of individual learning and cooperative learning: the agents solve problems only with the cases contained in their local case bases, but keep some heuristics on the quality of each case. When a case is found as having poor quality, a cooperative learning protocol is engaged to obtain a better case from another agent to replace the old one. The assumption behind Soh and Luo's work is that cooperative learning is expensive, and they try to engage it only when it is really needed. Moreover, they see cooperative learning as an off-line problem, i.e. it is never engaged while solving a problem.

Mc Ginty and Smyth [61] present a framework called Collaborative Case Based Reasoning (CCBR) that also can be viewed as FPL. The main difference of Mc Ginty and Smyth's work is that in CCBR an agent maintains the problem solving authority by solving all the problems locally. When an agent A_i has to solve a problem that falls outside his area of expertise, the problem is sent to the other agents. If some of the other agents have relevant cases to solve the problem, they are sent to A_i, who will then solve the problem using those cases retrieved from other agents. Notice that every agent is responsible of evaluating whether a problem falls within its area of expertise and that this approach is also focus on distributed retrieval. Mc Ginty and Smyth successfully apply their CCBR framework to personalized route planning, where each individual agent contains experience in solving plans for a specific user.

Leake and Sooriamurthi [51, 52] present another framework, called *multi-case-base reasoning* (MCBR), very similar to CCBR. MCBR deals with distributed systems where there are several case bases available for the same task. Moreover, each case base may not correspond to exactly the same problem, or may reflect some different user preferences, etc. Therefore cases must be adapted to be moved from one case base to another. Leake and Sooriamurthi explain that there are two different approaches to deal with multiple case bases: an "eager" approach consisting on merging and standardizing all the case bases in a single and centralized one, and a"lazy" one consisting on working with a single local case base and only retrieving cases from external case bases when needed. There is a common problem to the eager and the lazy approach: cross-case base adaptation, i.e. how to modify one case belonging to an external case base to fit in the local case base, while the lazy approach has an added problem: case dispatching.

Leake and Sooriamurthi [50] present several strategies to perform cross-case base adaptation: *no adaptation*, *linear interpolation* and *local approximation* (a case based adaptation technique). And two strategies to perform case dispatching: *threshold-based dispatching* and *case-based dispatching*. Later [53] they show that by combining the proper case dispatching and cross-case base adaptation techniques, the lazy MCBR approach can outperform the eager MCBR approach since the lazy system still keeps control of when solving problems using the local cases and when solving problems using the external (and adapted) cases. Moreover, they also show that the lazy approach is much less sensitive to noise in the external case bases, since the dispatching policy can learn to to dispatch less cases to the external case bases.

If we look back to the CBR cycle [1] shown in Section 2.2, we can see that

all the work presented in this section about multi-agent case based reasoning focuses on distributing the Retrieval stage of the CBR cycle. They all focus on a scenario where there exist a set of case bases (in some frameworks each case base is owned by an individual agent), and the main problem is to decide whether to retrieve cases locally, or dispatch the case to another agent's case base to retrieve cases there. But in all the scenarios explained here, once the cases are retrieved from the external case base, they are send to the agent that wanted to solve the case, and are reused by him. This is the main difference with our work, in our work the Retrieval Stage is performed locally, while it is the Reuse stage that is distributed, i.e. each agent has only access to the case in his individual case base, and cannot retrieve cases from an external agent's case base. In Chapter 4 we will present this distributed Reuse strategies in detail.

Finally, even in this exposition seems to picture Leake and Sooriamurthi and Mc Ginty and Smyth's work as previous to ours, it is not. It is contemporary related work. In fact, we published our first works in multi-agent CBR systems [70, 66] in the year 2001, the same year than the first works of Leake and Sooriamurthi and Mc Ginty and Smyth's work were presented.

2.3.4 Classification in Multi-Agent Systems

Some researchers have applied different learning methods than reinforcement learning, genetic algorithms or case based reasoning to multi-agent systems to solve classification or regression tasks.

Modi and Shen [64] defined the *distributed classification task*. Basically, the distributed classification task is similar to the scenario presented by Prasad, Lesser and Lander in [71], where there are a set of agents, each one with a local case base, and each agent has only access to a subset of attributes of the problems. However, Modi and Shen assume that all the agents have the same cases, and that each one only sees a subset of the attributes. Moreover, they assume that there are unique identifiers for each case shared by all the agents. Modi and Shen present two algorithms to solve the distributed classification task: DVS (Distributed Version Space) and DDT (Distributed Decision Trees). DVS is a distributed version of the VS (Version Space) algorithm, that converges to the same solution than VS without the agents revealing the information contained in their case bases to the rest of agents. DDT is a distributed decision tree learner that also keeps private the information of the local case bases. The only information that DVS and DDT broadcast to the other agents are the case identifiers. In our work we will also try to keep private the information contained in each agent's case base. However, we consider that having unique identifiers for the cases too strong an assumption for many domains, and therefore we will not support it. For this reason our agents cannot make use of DVS or DDT.

2.3.5 Summary

Summarizing, we have seen that there are two main problems/approaches in multi-agent learning:

1. Learning to behave in a society of learning agents, where the environment is not stationary (since the other agents also have learning capabilities).

2. Learning to take benefit of the other agents in order to solve problems.

Most of the work in multi-agent reinforcement learning focuses on the first one. Their main problem is that the scenario is not stationary, and they try to find strategies for learning agents that converge and that converge to good policies. Moreover, most work on genetic algorithms for multi-agent system also focus on this kind of problems.

However, work on multi-agent case based reasoning focuses on the second problem, i.e. we have a set of agents, each one with its local case base, and they have to learn to obtain the highest benefit from the other agent's case bases. The work presented in this section deals with the problem of distributed retrieval, i.e. how to know when the local case base is not competent enough and and it would be better to retrieve cases from an external case base. An exception to this rule is the work of Haynes et al. [38], that attack the first type of multi-agent learning problems using a case based approach.

In our work, we will focus on the second kind of multi-agent learning problems. Moreover, in our work, we do not perform distributed retrieval since each agent has only access to its individual case base and does not have access to the cases of other agents. Instead, we focus on distributed reuse, where individual agents solve a problem individually by CBR and then those predictions built by the individual agents are aggregated to build one global prediction. Moreover, notice the parallelism between this kind of multi-agent learning and ensemble learning. In both frameworks, there are groups of problem solving agents (classifiers) that can collaborate to solve problems. However, the main difference is that ensemble learning assumes centralized algorithms that have control over all the individual classifiers, while in multi-agent learning, there is no such centralized control.

Chapter 3

A Framework for Multi-Agent Learning

In this chapter we will introduce the multi-agent learning framework used throughout this monograph. First we will present the specific multi-agent system model used. Then, we present the formalization used to specify interaction among agents. We will also present the knowledge representation that we have chosen and how individual agents are implemented in our framework. After that, we will define the notion of *Multi-Agent Case Based Reasoning Systems* ($\mathcal{M}$AC) as the kind of multi-agent systems we are interested with. Basically, $\mathcal{M}$AC systems are multi-agent systems where each individual agent uses Case Based Reasoning (CBR) to solve problems and learn from experience. Inside the framework of $\mathcal{M}$AC systems, we will then introduce our approach to multi-agent learning and which are our research goals inside this framework. Finally, we present some capabilities that the CBR agents inside $\mathcal{M}$AC systems need in order to be able to apply the techniques that we will present in the following chapters of this monograph.

3.1 Multi-Agent Systems

A multi-agent system (MAS) can be defined as a loosely coupled network of problem-solving entities working together to find answers to problems that are beyond the individual capabilities or knowledge of the isolated entities [25].

One of the main features of multi-agent systems is autonomy. Individual agents have their own goals and will not act against these goals. Therefore, multi-agent systems will not follow the instructions provided by a distributed algorithm but will take their own decisions. A major difference between distributed applications and multi-agent systems lies in the notion of *goal*. In a distributed application the goal is given by that same application, and every process in the application works following a distributed algorithm to achieve that goal. In a multi-agent system the goals are individual to each agent. The

"joint goals" emerge from their interaction following an interaction protocol. Eventually a group of agents can join together and collaborate to solve a task if and only if that is beneficial for each one of the individual agents.

Another basic feature of multi-agent systems is sociality, i.e. the capability of agents to communicate with other agents (or with human users). For this purpose, agents need both: an *Agent Communication Language* (ACL) and a *Knowledge Representation Language*. Two agents can exchange messages only if they share an ACL, and they can only communicate knowledge if they share a knowledge representation language with which to represent it. Exchanging messages allow the agents to communicate, coordinate and cooperate. Moreover, when defining communication among agents, two situations can arise:

- That the individual agents share a common knowledge representation language
- That each individual agent has, in general, a different knowledge representation language

In the first situation (where agents share a common knowledge representation language), communication is easier, since both agents already share a common language. If two agents do not use the same knowledge representation language, then a common language must be defined and each agent must know how to translate information from its internal language to the shared one and viceversa.

Moreover, sharing a knowledge representation language is not enough, since the representation language only specifies the syntax with which knowledge is represented, while concepts in a domain are defined by an ontology. An *ontology* establishes a mapping between expressions in the knowledge representation language and concepts of the domain. Therefore, two agents that share a common ontology can assign the same meaning to the messages coming from the other agent. Communication between systems that have different ontologies is an open problem and multi-agent system developers currently assume that a common ontology exists for the application domain.

Environment awareness is also a requirement for an agent since an agent is usually a situated entity inside an environment. In our framework, agents are situated inside a social environment, not a physical one. The social environment is composed basically by the rest of agents in the multi-agent system (and the human users). Therefore, perception and action is equivalent to receive and send messages from or to other agents or users. Such social environments are formally studied in the field of *electronic institutions* [27].

Moreover, it is important to note that our research focus on collaborative multi-agent systems, where the individual agents do not compete against each other, thus we will not consider competitive environments such as auctions in our multi-agent learning framework.

Let us now define specifically how the interaction among agents works in our multi-agent framework, the formalism that we are going to use to define agent interaction, and the knowledge representation language shared by the agents to perform communication.

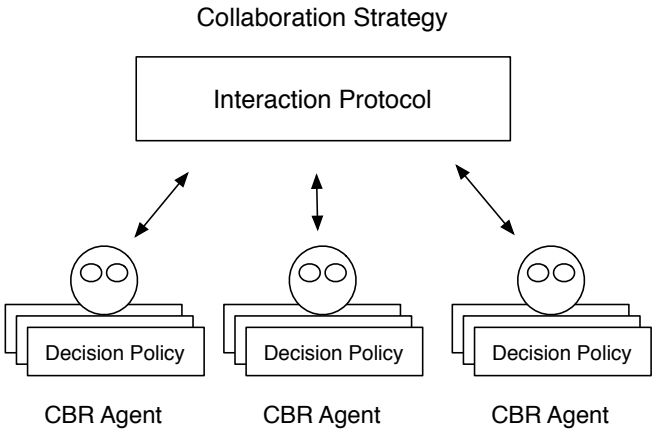


Figure 3.1: A collaboration strategy consists of an interaction protocol and a set of individual decision policies.

3.1.1 Collaboration Strategies

In our framework, all the interaction among agents is performed by means of *collaboration strategies*.

Definition 3.1.1. *A* collaboration strategy $\langle I, D_1, ..., D_m \rangle$ *defines the way in which a group of agents inside a* $\mathcal{MAC}$ *collaborate in order to achieve a common goal and is composed of two parts (see Figure 3.1):*

- *an interaction protocol* I
- *a set of individual decision policies* $\{D_1, ..., D_m\}$.

The *interaction protocol* of a collaboration strategy defines a set of interaction states, a set of agent roles, and the set of actions that each agent can perform in each interaction state. Each agent use its individual *decision policies* to decide which action to perform, from the set of possible actions, in each interaction state. Moreover, instead of specifying specific decision policies $\{D_1, ..., D_m\}$, a collaboration strategy may specify generic decision policies that each individual agent should personalize or simply impose some constraints in the specific individual decision policies used by the agents. Each agent is free to use any decision policy that satisfies those constraints.

Moreover, the agent communication language that we are going to use in our framework is based in the *speech act theory* [75], i.e. that speaking (communicating or sending messages) is acting. Therefore, each message (or illocution) is composed of 4 elements:

- Illocutionary particle: specifies the intention of the message, i.e. requesting, informing, etc.

- Sender: the agent sending of the message.
- Receiver: the agent receiving of the message.
- Content: the information that the *sender* wants to communicate to the *receiver* with the intention specified in the *illocutionary particle*. The content is expressed in the shared knowledge representation language of the agents.

For example, *Request*(A_i, A_j, m) is a message where *Request* is the illocutionary particle, A_i is the sender of the message, A_j is the receiver of the message and m is the content of the message.

Speech act theory suits the social environment in which the agents are situated in our framework (composed just by the rest of agents in the system and by the human users), since the set of actions that an agent can perform in each interaction state is in fact a set of possible messages that can be meaningfully sent.

We have used the ISLANDER formalism [28] (also available online at http://e-institutor.iiia.csic.es/bib/publications.php) to specify the interaction protocols in our framework. In the remainder of this section, we will briefly summarize the ISLANDER notation (see [28] for a more detailed explanation) and we will specify a basic agent interaction protocol using this notation as an example.

3.1.2 Interaction Protocol Specification

In the ISLANDER notation, an interaction protocol is specified as a directed graph, composed of four basic elements: *States*, *Transitions* (connecting two states in an oriented way), *Illocutions* (containing a sender, the receivers, and optionally some content parameters), and *Conditions* (specifying whether an illocution is valid or not depending on some restrictions). In order to specify all the above, the following notation is used:

An interaction state is noted w_i (notice that an interaction state makes reference to the state of the protocol, and not to the internal state of the individual agents taking part in it). An *illocution* is noted p_i, and has the form: $p_i =$ *ip(sender, receiver,...)* (where *ip* is an illocutionary particle). A *Condition* is denoted as c_i, and is composed of a set of constraints linked by conjunctive or disjunctive relations.

Interaction states are linked by *transitions*. Each Transition is labelled by an *illocution* and optionally a *Condition* (p_i/c_j). An *illocution* p_i is only valid when its *Condition* c_j is satisfied. If a transition labelled with an illocution p_i links to states s_1 and s_2 means that when the illocution p_i is sent, the protocol will move from state s_1 to state s_2. Moreover, some *Transitions* are labelled with a *Timeout* $[!t]$ instead of an *illocution*, meaning that when the time t is reached after the arrival to the current state, the transition is in effect and the system moves to the next state.

A variable is noted $?x$. When we want to denote the value of a previously bound variable, we write $!x$. And when we want to obtain the set of values that

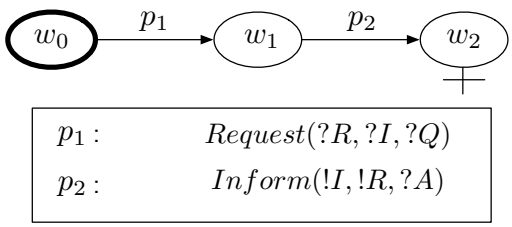


Figure 3.2: Specification of a basic Request-Inform protocol.

a variable has taken we write: $!w_i w_j x$; this expression denotes the set of values taken by the variable x during the last transition of the protocol from w_i to w_j. And if we want to obtain the set of all the instantiations of a variable from the beginning of the protocol, we write $^*!x$.

In the remainder of this monograph, all the interaction protocols will be specified using this notation.

3.1.2.1 Interaction Protocol Specification Example

Figure 3.2 shows the formal specification of a simple interaction protocol. The interaction protocol specified is a basic Request-Inform protocol, where an agent requests some information from another agent, and the requested information is send back to the initial agent.

The protocol is composed of 3 interaction states: w_0, w_1, and w_2. The initial state is w_0 and w_2 is the final state. There are two agents taking part in the protocol, the agent that request information (the requester) is referred by variable R and the informer is referred by variable I. Notice that in message p_1 all the variables are preceded by a ? symbol, to indicate that they can take any value: $?R$ and $?I$ will be instantiated to the requester and the informer agents and $?Q$ will be instantiated with the content of the message sent from the requester to the informer. Notice also that in message p_2 variables $!R$ and $!I$ are preceded by a ! symbol. This means that they must take the same value taken in message p_1 (i.e. the agent that receives message p_1 must be the agent that sends p_2).

The three interaction states are connected by two transitions. The first transition connects state w_0 with w_1, and is labelled with message p_1. Therefore, when the requester agent sends message p_1 to the informer agent, the protocol will move to the interaction state w_1. In the same way, state w_1 is linked with state w_2 with a transition labelled with message p_2. Therefore, when the informer agent sends message p_2 to the requester, the protocol will move to the interaction state w_3 (that is a final state) and the protocol will end.

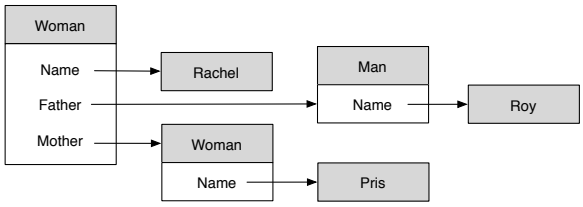


Figure 3.3: Example of a simple feature term.

3.2 Knowledge Representation and Agent Platform

In our work, we have chosen the *feature terms* formalism for knowledge representation. All the cases and problems stored by the agents will be represented as feature terms. Moreover, the agents have been implemented in the NOOS [4] agent platform, which supports feature terms. In this section we are going to describe both feature terms and NOOS.

3.2.1 The Feature Terms Representation Language

Feature Terms (*ψ-terms*) are a generalization of the first order terms. The main difference is that in first order terms the parameters are identified by position, while in a feature term the parameters (called *features*) are identified by name. A first order term has the form $person(x_1, x_2, x_n)$ (for example, an instantiation of a first order term is $person(rachel, roy, pris)$), while a feature term has the form $person[name \doteq x_1, father \doteq x_2, mother \doteq x_3]$.

Another difference is that feature terms have a *sort* hierarchy. For instance, the previous example uses the sort *person*. These sorts can have subsorts (e.g. *man* and *woman* are subsorts of *person*) creating a hierarchy of sorts. Formally, we have a set of sorts $\Gamma = \{\tau_1, ..., \tau_n\}$, where each sort τ_i is a symbol. Sorts have an informational order relation $\leq$. The sort $\bot \in \Gamma$ always exist and is the smallest (i.e. the one that contains less information) sort. Notice that $\tau_1 \leq \tau_2$ means that τ_1 contains less information than τ_2 (i.e. that τ_1 is more general than τ_2). For instance, if we have the sorts *person* and *woman* (with the expected meanings), we have that $person \leq woman$.

The minimal element $\bot$ is called *any* and represents the minimum information; therefore, *any* is the top sort of the sort hierarchy: all the sorts are subsorts of *any*. The values of the features of a feature term are also feature terms (with their own sorts). When a feature has value $\bot$ we say that the feature is undefined. When a feature term has no defined features (i.e. all of its features have value $\bot$) it is called a *leaf*.

For instance, Figure 3.3 shows a graphical representation of a feature term.

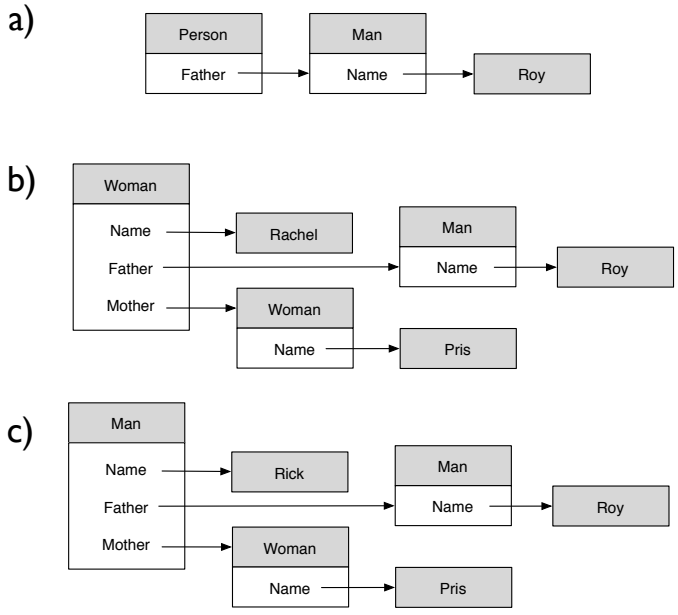


Figure 3.4: Three feature terms: a) subsumes both b) and c).

Each box in the figure represents a node. Nodes are labelled by sorts, and on the lower part, all the features that have a value different than *any* are shown. The arrows mean that the feature on the left part of the arrow takes the node on the right as value. In that example, the left-most node is the *root* node. We will note by $\tau(\psi)$ the sort of the root node of a feature term ψ.

In Figure 3.3 the nodes labelled with *Rachel*, *Roy* and *Pris* are leaf nodes. The left most node is the root node of ψ, and therefore $\tau(\psi) = woman$.

From the $\leq$ relation among sorts an information order relation ($\sqsubseteq$) between feature terms, called *subsumption*, can be defined [4]:

Definition 3.2.1. *A feature term* ψ_1 subsumes *another feature term* ψ_2*, noted as* $\psi_1 \sqsubseteq \psi_2$*, if all the information in* ψ_1 *is also present in* ψ_2*, i.e. if the following two conditions are met:*

1. $\tau(\psi_1) \leq \tau(\psi_2)$.
2. *For each defined feature* $f_i = x_i$ *in* ψ_1*, exists a feature* $f_i = x'_i$ *in* ψ_2 *such that* $x_i \sqsubseteq x'_i$.

We can also say that ψ_2 satisfies ψ_1 or that ψ_1 is a generalization of ψ_2. For instance, Figure 3.4 shows three feature terms, the feature term shown in Figure 3.4.a subsumes both feature terms in Figure 3.4.b and Figure 3.4.c because the

sort of the root of 3.4.a is *Person*, that is more general than *Woman* and *Man* and because every feature in 3.4.a subsumes the corresponding feature of 3.4.b and 3.4.c. Moreover, notice that 3.4.a can be interpreted as "a person whose father is called Roy", and both 3.4.b and 3.4.c are two specific persons, one called Rachel and the other called Rick, whose father is Roy.

See [6] for further detail in the feature term formalism.

3.2.2 The NOOS Agent Platform

The NOOS agent platform [4] is an agent programming environment that supports agents designed in the NOOS representation language to communicate, cooperate, and negotiate using FIPA standards and the methodology of Agent-Mediated Institutions [68, 27, 28] developed at the IIIA-CSIC.

The NOOS language has been designed to implement machine learning methods using the feature term formalism to represent information, and specifically to easily implement Case Based Reasoning (CBR) methods inside agents. See Appendix B for a more detailed explanation on the NOOS agent platform and a description of how $\mathcal{M}$AC systems presented in this monograph have been implemented.

3.3 Multi-Agent Case Based Reasoning Systems

In the previous section we have introduced multi-agent systems and the specific assumptions that we make about multi-agent systems in our work. Moreover, in the work presented in this monograph, we focus on applying lazy learning techniques—and specifically Case Based Reasoning (CBR)— to multi-agent systems. For this purpose, we are going to define a specific type of multi-agent systems: the *Multi-Agent Case Based Reasoning Systems* ($\mathcal{M}$AC):

Definition 3.3.1. *A* Multi-Agent Case Based Reasoning System *($\mathcal{M}$AC)* $\mathcal{M} = \{(A_1, C_1), ..., (A_n, C_n)\}$ *is a multi-agent system composed of* $\mathcal{A} = \{A_i, ..., A_n\}$*, a set of CBR agents, where each agent* $A_i \in \mathcal{A}$ *possesses an individual case base* C_i*.*

Each individual agent A_i in a $\mathcal{M}$AC is completely autonomous and each agent A_i has access only to its individual and private case base C_i. A case base $C_i = \{c_1, ..., c_m\}$ is a collection of cases. Each agent has (in general) its own CBR method(s) to solve problems using the cases stored in its individual case base. Agents in a $\mathcal{M}$AC system are able to individually solve problems, but the can also collaborate with other agents to solve problem in a collaborative way.

In this framework, we will restrict ourselves to analytical tasks, i.e. tasks, like classification, where the solution of a problem is achieved by selecting a solution class from an enumerated set of solution classes. In the following we will note the set of all the solution classes by $\mathcal{S} = \{S_1, ..., S_K\}$. Moreover, we will note the problem space by $\mathcal{P}$, that contains all the problems that can be

described in a particular application domain. Therefore, a case can be defined as:

Definition 3.3.2. *A* case $c = \langle P, S \rangle$ *is a tuple containing a case description* $P \in \mathcal{P}$ *and a solution class* $S \in \mathcal{S}$.

Notice that case descriptions are defined over the problem space $\mathcal{P}$. In the following, we will use the terms *problem* and *case description* indistinctly. Therefore, we can say that a case consists of a case description plus a solution class, or that a case is a problem/solution pair.

We will use the dot notation to refer to elements inside a tuple. e.g., to refer to the solution class of a case c, we will write $c.S$. Moreover, we will also use the dot notation with sets, i.e. if C is a set of problems, $C.P$ refers to the set of problems contained in the cases in C, i.e. $C.P = \{c.P | c \in C\}$.

3.4 Individual Problem Solving

A CBR agent A_i in a $\mathcal{M}$AC must be able to solve problems individually by using the cases stored in its individual case base C_i. In general each agent is free to use any CBR method to solve problems. In our experiments, we have used many different methods to solve problems: Nearest Neighbor, K-Nearest Neighbor, Decision Trees, and LID[9] (See Section 2.2.2).

The only requirement on the CBR method that an agent in a $\mathcal{M}$AC uses is that after solving a problem P, and agent A_i must be able to build a *Solution Endorsement Record*:

Definition 3.4.1. *A* Solution Endorsement Record *(SER) is a tuple* $\mathbf{R} = \langle S, E, P, A \rangle$ *where the agent* A *has found* E *(where* $E > 0$ *is an integer) cases endorsing the solution* S *as the correct solution for the problem* P.

If the CBR method of an agent can return more than one possible solution class, then a different SER will be built for each solution. For example, if an agent A_i is using a 5-Nearest Neighbor method, and when solving a problem P, the 5 nearest neighbors are the cases c_1, c_2, c_3, c_4, c_5, where c_1, c_2 have S_1 as the solution class, and the rest have S_2 as the solution class. Then, A_i will build the following SERs: $\{\langle S_1, 2, P, A_i \rangle, \langle S_2, 3, P, A_i \rangle\}$, i.e. agent A_i has found 2 relevant cases to solve P that endorse the solution S_1 as the correct solution, and 3 that endorse solution S_2. Any CBR method from which output a set of SERs can be built is suitable to be used by a CBR agent in a $\mathcal{M}$AC.

The SERs have the function of standardizing the way in which agents represent the outcome of the problem solving regardless of the CBR method used. In this way, agents can communicate the individually found solutions for problems.

3.5 An Approach to Multi-Agent Learning

We are interested in studying certain kinds of multi-agent learning using the $\mathcal{M}$AC framework. In this monograph, we will work on cooperative settings,

where the agents solve classification tasks. Our goal is to apply lazy learning techniques to multi-agent systems in order to show that, through collaboration, individual learning agents and multi-agent systems can improve their performance (by improving performance, we understand increasing classification accuracy, or reducing problem solving time). Both learning and collaboration are ways in which an agent can improve individual performance. In fact, there is a clear parallelism between learning a collaboration in multi-agent systems, since they are ways in which an agent can deal with its shortcomings. Let us show which are the main motivations that an agent can have to learn or to collaborate:

Motivations to learn:

1. Increase the quality of solutions (e.g. accuracy),
2. Increase efficiency,
3. Increase the range of solvable problems.

Motivations to collaborate:

1. Increase the quality of solutions (e.g. accuracy),
2. Increase efficiency,
3. Increase the range of solvable problems,
4. Have access to resources that only other agents can use.

Therefore, learning and collaboration are very related. In fact, with the exception of motivation to collaborate number 4 above, they are two extremes of a continuum of strategies to improve performance. An agent can choose to increase performance by learning, by collaborating ,or by finding an intermediate point that combines learning and collaboration in order to improve performance. In this work, we will present a framework in which both learning and collaboration will be studied as ways to increase the performance of CBR agents. Specifically, in this work we will focus on how can an individual learning agent improve its performance by collaborating with other agents, how can a group of learning agents improve group performance by collaborating among them, and finally how can a learning agent decide whether it is better to work individually or to work within a group.

In order to investigate how individual agents' performance can be improved by collaborating with other agents we will experiment with different collaboration strategies that allow individual agents to improve their individual predictions. For instance, we will define collaboration strategies that allow agents to barter cases among them, so that an individual agent can improve the contents of its local case base, thus leading to an improvement of individual prediction. Information coming from other agents can also be useful in two situations: when an agent has to decide whether a new case should be added to its local case base or not; and when it has to decide whether a case should be deleted from its local

base base or not. Moreover, typically a CBR system does not add to the case base every case it has access to, but has a *retention policy* (or a case base maintenance policy) to decide which cases to add into (or to delete from) the case base. In a $\mathcal{M}$AC system, since cases discarded by some individual agent may be useful to other agents, we will investigate collaboration strategies that take advantage of this possibility in order to improve the quality of the cases retained and thus improving future performance. Namely, we will study two families of strategies: collaboration strategies for case retention and collaboration strategies for case bartering. Retention strategies focus on how agent decide which cases to add to the case base and Bartering strategies are strategies that allow agents in a $\mathcal{M}$AC system to barter cases so that every agent achieves a better individual case base.

Moreover, improving group performance is also one of our goals. We are interested in collaboration strategies that allow groups of agents to benefit from the *ensemble effect* (Section 2.1). In order to benefit from the ensemble effect, agents in $\mathcal{M}$AC systems will form *committees*. A committee is a group of peer agents that work together in order to obtain a solution for a problem based on the individual agents' predictions for that problem. Our research is related to ensemble learning since an agent in a committee can be considered as an individual classifier in an ensemble, but it is worth noticing that committees are different from ensembles in important aspects: ensemble learning techniques typically consist of a centralized algorithm that has access to the entire data set, that creates a set of classifiers, distributes the data set among them, and finally defines the way in which those classifiers are coordinated and how the individual predictions of the classifiers are aggregated.

In a $\mathcal{M}$AC system, an agent that wants to form a committee cannot *create* new agents, and has to work with the agents that are already in the $\mathcal{M}$AC system. Moreover, there is no single agent that has access to all the data: the data is already distributed among the individual agents' case bases and no agent can individually decide how to redistribute data among the agents. Moreover, in $\mathcal{M}$AC systems agents are autonomous, and therefore no centralized algorithm can force any agent to collaborate with other agents if they are not willing to. Therefore, agents must coordinate in a decentralized way (by means of collaboration strategies) in order to form committees. Notice that in $\mathcal{M}$AC systems there is a distinction between coordination and prediction aggregation: coordination refers to how the information flows among agents and prediction aggregation refers to the internal methods that the agents use in order to integrate the predictions coming from other agents. Coordination is specified in a collaboration strategy in the form of an interaction protocol and prediction aggregation is defined in the form of decision policies.

The main differences among ensemble learning and $\mathcal{M}$AC systems are summarized in Table 3.1. Moreover, one of the main goals of this work is to study the ensemble effect under different conditions: namely, when data and control are decentralized. We are interested in determining which are the conditions for the ensemble effect to take place in committees of CBR agents, i.e. which properties must the case bases satisfy and what strategies can the individual

Ensemble Learning	$\mathcal{MAC}$ Systems
Centralized control	Decentralized control by means of individual decisions
Can create classifiers	Preexisting agents/classifiers
Centralized distribution of data	Preexisting data distribution, each agent has a local view of the data
Centralized prediction aggregation	Prediction aggregation defined by individual decision policies

Table 3.1: Differences between ensemble learning and $\mathcal{MAC}$ systems.

agents follow in order to obtain the maximum benefit from the ensemble effect.

Specifically, this monograph will mainly focus in the following problems:

- How can individual agents obtain information from other agents in order to improve individual predictions,
- When collaborating with other agents is beneficial (i.e. to form committees),
- How to decide which agents should join the committee,
- How the agents' individual case bases characteristics affect the performance of the committee,
- How to infer a global prediction by aggregating individual predictions of autonomous agents agents (this aggregation issue is also present in ensemble learning),
- How can agents improve individual case base by using retention and maintenance policies that profit from collaborating with other agents,
- How can all the above goals can be achieved in a decentralized way as a result of the individually taken decisions of the agents in a $\mathcal{MAC}$ system.

In order to attack the previous problems, we will present several techniques and collaboration strategies based on a number of agent capabilities. The remainder of this section presents the extra capabilities of the agents in a $\mathcal{MAC}$ must have in order to be able to use all the techniques and collaboration strategies presented in this monograph. Specifically, we will present two capabilities: *competence assessment* and *justification generation*. Finally, at the end of this chapter we will present a CBR view of our approach, explaining how the techniques that we will present in the rest of this monograph fit inside the CBR framework.

3.5.1 Competence Models

The *competence assessment* capability is the ability of an agent to build *competence models*. Moreover, in $\mathcal{M}$AC systems where agents individually solve problems, and can send problems to solve to other agents, *competence models* are a must.

Definition 3.5.1. *A* competence model $M_A(P) \rightarrow [0,1]$ *assesses the likelihood that the prediction made by an agent (or a set of agents) A for a problem P is going to be correct.*

A competence model has several uses:

- An agent A_i can use a competence model M_{A_i} of itself to assess whether it can solve a problem individually, or it needs help of other agents to solve the problem together. A competence model M_{A_i} of the agent A_i used by A_i itself is called a *self-competence* model.
- Competence models can be used to decide which agent is more likely to be helpful in solving a problem P.
- An agent can also use a competence model $M_{\mathcal{A}^c}$ of a committee of agents $\mathcal{A}^c$ to assess whether that specific committee of agents is competent enough to solve a problem.

In Chapter 4 we are going to explain in more detail how the ability to use competence models can be exploited in $\mathcal{M}$AC systems in order to build good committees. Specifically, we will show that competence models are useful to define the individual decision policies needed in some collaboration strategies. Moreover, we are going to present a proactive learning technique that allows individual agents to learn competence models for specific collaboration strategies.

3.5.2 Justification Generation

In multi-agent systems, where the performance of an agent can depend on the predictions provided by other agents, it would be desirable that each agent can provide a justification of their predictions. For instance, if an agent A_i that wants to solve a problem P decides (using its own competence model) that it is better to ask another agent A_j to solve it, A_i completely depends on the correctness of the solution provided by A_j for the problem P. In this situation, it is desirable that A_j can provide A_i with a justification of its prediction. In this way A_i could examine the justification provided and assess whether the solution provided by A_j is reliable or not. Specifically, we can define a *justification* as:

Definition 3.5.2. *A* justification J *built by an agent to solve a problem P that has been classified into a solution class* S_k *is a description of the relevant information of P used for predicting* S_k *as the solution class.*

In our work, we have used two methods that are able to build justifications: LID, and decision trees. Let us illustrate the meaning of a justification with two examples.

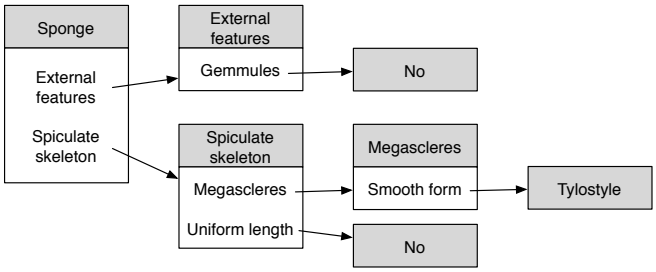


Figure 3.5: A symbolic justification generated by the LID CBR method in the marine sponges classification domain.

3.5.2.1 Building a Justification with LID

As we have explained in Section 2.2.2, LID is a CBR method that builds a symbolic similitude term J using the feature term formalism. Figure 3.5 shows an example of a justification built using LID. This similitude term J contains the relevant features in which the problem P is similar to the retrieved cases. Therefore, we can say that the problem has been classified into a specific solution class S_k because it had in common all the information present in J with the retrieved cases, i.e. J is the justification of having classified P into the solution class S_k. Specifically, the justification shown in Figure 3.5 has been built for solving a problem in the *marine sponges* domain, where to solve a problem implies classifying a marine sponge in its correct family. The justification shown can be interpreted as: the *Sponge* has been classified in family S_k because "There are *no Gemmules* in the *External features* of the *sponge*, the *Smooth form* of the *Megascleres* of the *Spiculate skeleton* of the *Sponge* is of type *Tylostyle*, and the *Spiculate skeleton* of the *Sponge* has *not* a *Uniform length*".

Notice that a justification J built by LID will always subsume the problem and the description of the retrieved cases, i.e. $J \sqsubseteq P$ and $\forall_{i=1...n} J \sqsubseteq c_i.P$ where $c_1, ..., c_n$ is the set of retrieved cases. That is to say, a justification J is a generalization of the problem and of the retrieved cases. Therefore since lazy learning builds a local approximation of the target concept (Section 2.2), a justification J can be seen as a symbolic local approximation built by LID to solve the problem P.

3.5.2.2 Building a Justification with a Decision Tree

As we will show, decision trees can generate justifications in a very straightforward way. When a problem is solved using a decision tree, only one branch of the tree is followed till a tree leaf l is found. The set of nodes of the tree in that branch contain the conditions that the problem matches and that the decision tree has used to classify the problem in the leaf l. Therefore, that is the information that should appear in the justification. Figure 3.6 illustrates the process

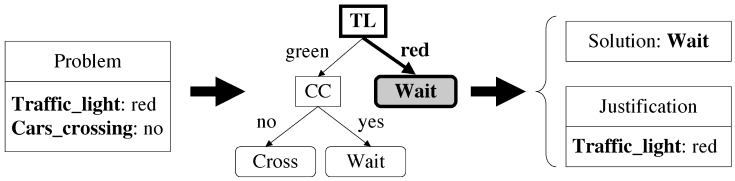


Figure 3.6: Example of a justification generation by a decision tree in a toy problem.

for a toy problem. In the figure we can see a problem (containing two features: *traffic_light* and *cars_crossing*)), and a decision tree that predicts whether we can cross a street or not. The problem is classified using the decision tree, and only he attribute *traffic_light* is used to decide that the problem belongs to the class *wait*. Therefore, the justification will contain only the *traffic_light* feature, as shown in the right-most part of the figure. The interpretation is that "the problem belongs to the class *wait* since *traffic_light* has the value *red* ".

Notice that standard decision trees algorithms (such as C4.5) cannot be directly used in CBR since a decision tree does not store the cases used to learn it. However, decision trees can be used as indexing mechanism for CBR by storing the cases in the leaves of the tree. Moreover, the method we have presented to generate justifications is not restricted to any specific decision tree algorithm and, for instance, lazy decision trees could also be used.

3.5.2.3 Justifications in $\mathcal{M}$AC Systems

Justifications can enrich agent interaction in many ways, as we will show in the remainder of this monograph. However, building a justification is not enough in a $\mathcal{M}$AC system: the agents need a way to communicate their justifications to other agents. For this purpose, we define the *Justification Endorsement Records*.

Definition 3.5.3. *A* Justification Endorsement Record *(JER) is a tuple* $\mathbf{J} = \langle S, J, P, A \rangle$ *where an agent* A *considers the solution* S *as the correct solution for the problem* P *endorsed by the justification* J*.*

A JER defines a standard way in which an agent can communicate a justification to another agent (it is the equivalent of a solution endorsement record (SER) when there is a justification involved). Section 7.2 provides an more in detail analysis of justifications and its role in multi-agent CBR systems.

3.5.3 A CBR View of the Multi-Agent Learning Framework

The CBR cycle consists of four processes [1] (See Figure 3.7): Retrieve, Reuse, Revise and Retain. The *Retrieve* process selects a subset of cases from the case

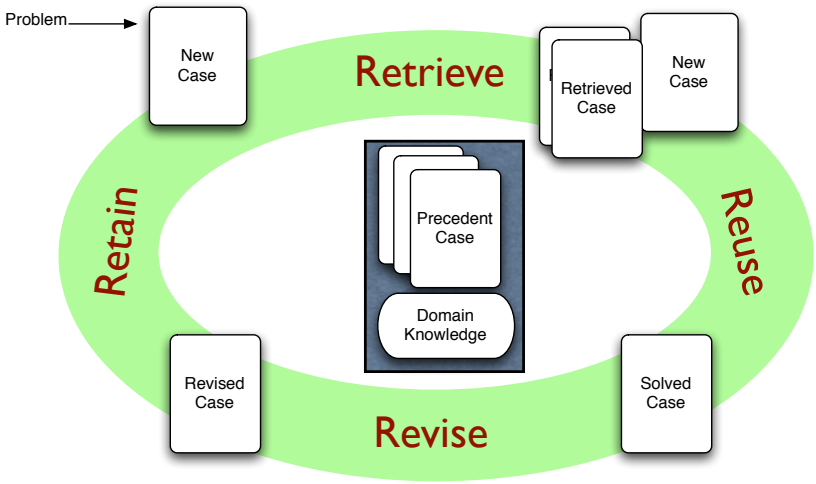


Figure 3.7: The Case Based Reasoning Cycle [1].

base that are relevant to the current problem. The *Reuse* process generates the solution to the new problem (by some specific technique) from the solutions of the retrieved cases. In the *Revise* process, the obtained solution is examined by an oracle, that gives the correct solution (as in supervised learning). Finally, in the *Retain* process, the system decides whether to incorporate the new solved case into the case base or not.

Reuse techniques can be based on adaptation of the solutions of all (or some of) the retrieved cases or in voting schemes if the CBR system is solving classification tasks. For instance, k-Nearest Neighbor (k-NN) methods can use a voting mechanism among the retrieved cases to decide the final prediction. In general, the k retrieved cases by a k-NN method have several different solution classes. The reuse process of a k-NN method consists of an integration of the solution classes of the k retrieved cases, a common way to perform this integration is using a voting scheme. The most used method for reuse in k-NN is majority voting, that consists of determining how many cases of each different solution class have been retrieved, and the final prediction is the class with more cases. However, more complex voting schemes can be defined.

In the $\mathcal{M}$AC framework, the Retrieve process is performed individually, since each agent has access to the contents of its individual case base but not to that of other agents. Therefore, our framework does not have a distributed Retrieve process (as other approaches do [71, 69, 61, 51, 52, 50]). The result of the Retrieve process performed by an agent in a $\mathcal{M}$AC system is reified in the form of a set of SERs (solution endorsement records) or JERs (justification endorsed records).

When several agents join a committee to solve a problem, each one performs individually the Retrieve process, while the Reuse process is performed in a dis-

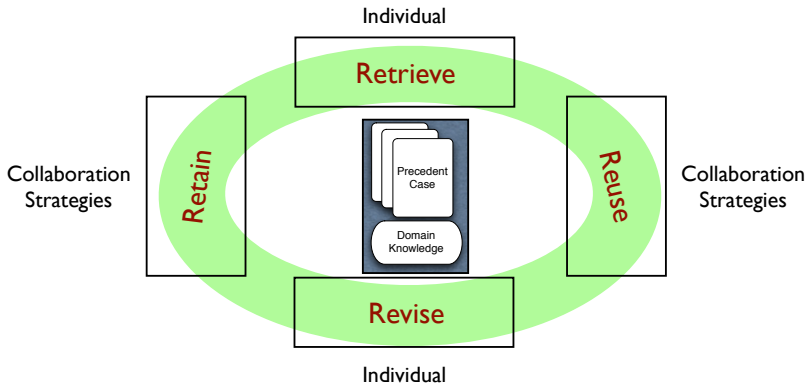


Figure 3.8: Distributed and individual processes of the CBR cycle in the $\mathcal{M}$AC framework.

tributed way. In our framework we focus on classification tasks, and therefore the Reuse process consists on selecting a specific solution class $S_k \in \mathcal{S}$ for the problem. The selection of this solution class is performed in a committee by means of a voting scheme. In the voting scheme, each agent will cast votes endorsing different solution classes based on the result of the individually performed Retrieve process. The resulting solution of the voting scheme will be yielded as the final prediction for the problem.

During the Retain process, a CBR system incorporates new cases into the case base. Every agent can have an individual retention policy to decide which cases to incorporate to the case base and which cases to discard. As cases discarded by some agents can be useful to other agents, in Chapter 8 we will also define collaborative retention strategies that allow agents in a $\mathcal{M}$AC system to take more benefit of the cases they have access to.

Summarizing, in our multi-agent learning framework, from the four processes in the CBR cycle, Retrieve and Revise are performed individually by each agent in the system while Reuse and Retain are distributed processes performed following the collaboration strategies that we present in this monograph, as shown in Figure 3.8. Namely, Chapters 4, 5 and 7 present collaboration strategies to perform distributed Reuse and Chapters 8 and 9 present collaboration strategies to perform distributed Retention.

3.6 Summary

The main goal of our research is to apply lazy learning methods (and specifically CBR methods) to multi-agent systems where cooperation can take place. For this purpose, in this chapter we have introduced the framework for *Multi-agent Case Based Reasoning* ($\mathcal{M}$AC) systems. $\mathcal{M}$AC systems offer a good framework

to study learning and collaboration in multi-agent systems. Learning in $\mathcal{M}$AC systems is achieved by having CBR agents that individually use CBR to solve problems. Moreover, those CBR agents can cooperate following what we call *collaboration strategies* (See Section 3.1.1).

A very related area to our research is ensemble learning. Specifically, we are interested in the *ensemble effect*, responsible for the accuracy improvements in ensemble learning. Inside $\mathcal{M}$AC systems we will study *committees*, that are groups of agents that join together to solve a problems in a similar way to ensemble methods (that is, benefitting from the ensemble effect) but preserving the autonomy of the CBR agents. We are interested on studying how CBR agents in a $\mathcal{M}$AC system can decide when to for committees and how agents an decide which agents should join the committee in order to benefit from the ensemble effect.

Committees allow CBR agents to improve performance by collaborating with other agents. However, another way to improve individual performance of CBR agents is by improving their individual case bases. Inside $\mathcal{M}$AC systems we are interested in studying collaboration strategies that improve the individual case bases of the agents. Namely, we will study two families of strategies: collaboration strategies for case retention and collaboration strategies for case bartering. We are interested on studying case retention strategies that can improve the retention process of CBR agents with respect to retention techniques that do not perform collaboration. Another approach to improve individual case bases are the case bartering strategies that are based on exchanging cases among agents. We are interested in studying case bartering strategies that by redistributing the cases among the individual case bases improve both individual and committee performance.

Finally, we have analyzed $\mathcal{M}$AC systems from the viewpoint of the CBR cycle since we are specially interested in CBR techniques. A key characteristic of collaboration to solve problems in $\mathcal{M}$AC systems (i.e. committees) with respect to other multi-agent CBR works is that the retrieve process is performed individually and collaboration takes place in the reuse process, i.e. committees are not a technique to perform distributed retrieval. Moreover, collaborative retention strategies and bartering strategies take place during the retention process of the CBR cycle.

Chapter 4

Committee Collaboration Strategies

In this chapter we are going to introduce the notion of *committees*. Basically, a committee is a group of agents that join together in order to find the solution of a given problem by means of a voting mechanism. Each individual agent member of a committee individually gathers evidence about the solution of the problem, and then a voting system is used to obtain a global prediction from the individually gathered evidence. Specifically, we will present the *Committee Collaboration Strategy* and empirically evaluate its performance.

4.1 Introduction

Committees provide a way in which individual agents can collaborate with other agents in order to improve classification accuracy thanks to the ensemble effect (see Section 2.1). One of the main goals of this chapter is to study the improvement on classification accuracy achieved thanks to the ensemble effect by individual agents that collaborate by forming committees. We will study the ensemble effect in several multi-agent scenarios, with the goal of determining which are the conditions under which the ensemble effect takes place.

Although this approach is very related to the ensemble methods presented in Section 2.1 there is a fundamental difference: while ensemble methods focus on *creating* ensembles in order to improve the classification accuracy of a given learning method, our approach focuses on the following scenario: *given* a set of agents (that can be considered as the individual classifiers), how can they collaborate in order to take benefit from the ensemble effect. Bagging or Boosting are methods that specify the way in which the individual classifiers are generated. In our approach we do not assume anything about the individual classifiers. Each individual agent has individually collected its own case base, and we cannot assume anything about them. Therefore, the difference can be summarized as follows: ensemble methods *build* a set of classifiers to finally build

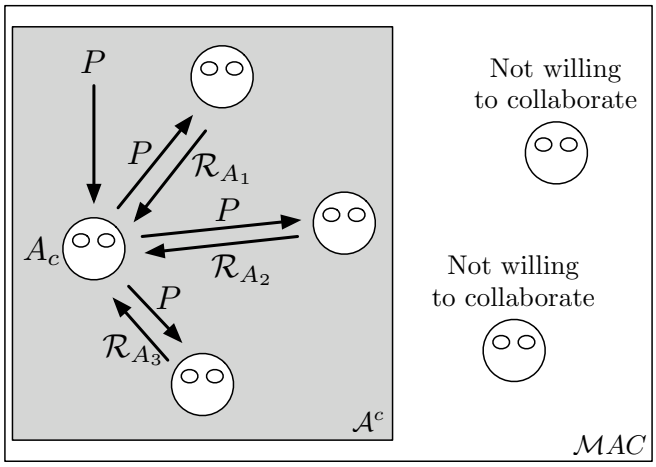


Figure 4.1: Illustration of a $\mathcal{MAC}$ system where an agent A_c is using CCS in order to convene a committee to solve a problem.

an ensemble, while our approach focuses on defining collaboration strategies so that a given set of agents can work as an ensemble without compromising their autonomy.

The remainder of this chapter is structured as follows: first, we present the the *Committee Collaboration Strategy*. Then, we will present the voting system used in the Committee Collaboration Strategy to perform the distributed Reuse process. Finally, experiments in several scenarios are presented to evaluate the performance of the Committee collaboration strategy under different situations and using several voting systems.

4.2 The Committee Collaboration Strategy

This section presents the *Committee Collaboration Strategy* (CCS) that allows a group of agents to benefit from the ensemble effect by collaborating when solving problems. Let us first define what is a committee.

Definition 4.2.1. *A* Committee *is a group of agents that join together to predict the solution of a problem P. Each agent individually gathers evidence about the solution of P and then contribute to the global solution by means of a voting process.*

When a committee of agents solves a problem, the sequence of operation is the following one: first of all, an agent receives a problem to be solved and convenes a committee to solve the problem; the problem is sent to all the agents in the committee and every agent preforms the Retrieval process individually;

after that, instead of performing Reuse individually, each agent reify the evidence gathered during the Retrieve process about the likely solution(s) of the problem in the form of a collection of SERs. The Reuse process is performed in a collaborative way by aggregating all the SERs to obtain a global prediction for the problem using a voting process (see Section 4.3).

The *Committee Collaboration Strategy* allows the agent members of a $\mathcal{M}$AC system to act as a committee. Formally, let $\mathcal{M}$ be a $\mathcal{M}$AC system composed on a set $\mathcal{A}$ of agents and $A_i \in \mathcal{A}$ an agent that wants to solve a problem P using the committee collaboration strategy. We will call A_i the *convener* agent, since it is the one that receives a problem and convenes the committee. Since autonomy is one of the main features of agents the convener agent cannot force the rest of agents to join the committee. As a consequence, before convening a committee, an agent A_i that wants to use the Committee Collaboration Strategy first has to send an invitation to join the committee to the agents in the $\mathcal{M}$AC system. Each agent that accepts the invitation sends back to A_i an acceptance message. Thus, the convener agent plus those accepting the invitation constitute the set of agents $\mathcal{A}^c \subseteq \mathcal{A}$ that effectively form the committee.

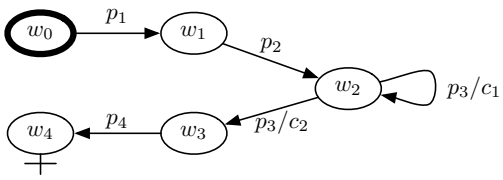
Specifically, the Committee Collaboration Strategy is composed by an interaction protocol and an individual decision policy:

Definition 4.2.2. *The* Committee Collaboration Strategy*(CCS) is a collaboration strategy $\langle I_{CCS}, D_V \rangle$, where I_{CCS} is the CCS interaction protocol shown in Figure 4.2 and D_V is a decision policy based on any voting system that can be used to aggregate the evidence gathered by the individual agents into a global prediction.*

Notice that the protocol I_{CCS} is completely defined by the Committee Collaboration Strategy while the decision policy D_V is not. The Committee Collaboration Strategy only specifies that D_V must be a voting system that can be used to aggregate the evidence gathered by the agents into a global prediction. In Section 4.3 we will specialize CCS with specific voting systems.

The interaction protocol I_{CCS} is formally described in Figure 4.2 and applies to a set of agents $\mathcal{A}^c$ that have committed to join a committee. The protocol consists of five states and w_0 is the initial state. When a user requests an agent A_i to solve a problem P the protocol moves to state w_1. When A_i broadcasts the problem P to all the other agents in the system the protocol moves to state w_2. Then, A_i waits for the SERs coming from the rest of agents in $\mathcal{A}^c$ while building its own SERs; each agent sends its SERs to A_i in the message p_3. Figure 4.1 shows a $\mathcal{M}$AC system where a set of agents $\mathcal{A}^c$ have accepted the invitation of a convener agent to form a committee. Moreover, some of the agents in the $\mathcal{M}$AC system are unwilling to collaborate and have not accepted the invitation.

When the SERs from the last agent are received the protocol moves to w_3. In w_3, A_i will apply the voting system defined in the individual decision policy D_V (with all the SERs received from other agents and the SERs built by itself) to obtain an aggregate solution. Finally, the aggregate solution S will be sent to the user in message p_4 and the protocol will move to state w_4 that is a final state. Notice that the number of messages received by the convener agent is



p_1 :	$Request(?User, ?A_i, ?P)$
p_2 :	$Request(!A_i, \mathcal{A}^c, !P)$
p_3/c_1 :	$Inform(?A_j, !A_i, ?\mathbf{R})/$ $\lvert !w_0w_1\mathbf{R}\rvert < \#(\mathcal{A}^c) - 2$
p_3/c_2 :	$Inform(?A_j, !A_i, ?\mathbf{R})/$ $\lvert !w_0w_1\mathbf{R}\rvert = \#(\mathcal{A}^c) - 2$
p_4 :	$Inform(!A_i, !User, ?S)$

Figure 4.2: Interaction protocol for the *Committee* collaboration strategy.

controlled by conditions c_1 and c_2. c_1 is satisfied when the number of messages received from other agents have been less than $\#(\mathcal{A}^c) - 2$, i.e. that there are still more than one agent that have yet to submit their SERs (notice that the total number of agents that need to submit SERs is $\#(\mathcal{A}^c) - 1$ since the SERs built by the convener agent do not have to be sent). c_2 is satisfied when exactly $\#(\mathcal{A}^c) - 2$ messages have been received, i.e. when there is exactly one agent that has to send their SERs. Therefore, when condition c_2 is met and a message p_3 with SERs is received, all the SERs have been received and the protocol moves to state w_3.

From a CBR point of view, the Committee collaboration strategy is a way to distribute the *Reuse* process (see Section 2.2). While *Retrieve* is performed individually by each agent, the Reuse stage uses the voting system in order to reach a global prediction in a collaborative way. As stated in Section 2.3.3, this is one of the main differences with other distributed CBR approaches that perform Retrieve as a distributed process [71, 69, 61, 51, 52, 50].

Next section presents the voting system that we will use in the rest of this monograph to aggregate the SERs in a committee.

4.3 Bounded Weighted Approval Voting

In this section we are going to present a voting system called *Bounded Weighted Approval Voting* (BWAV) that we will use in in the rest of the monograph. However, Section 4.5.4 defines several alternative voting mechanisms and empirically compares them with BWAV.

The principle of the the BWAV voting system is that each agent votes for

solution classes depending on the number of retrieved cases for each class. The larger the number of retrieved cases endorsing a solution class, the stronger that an agent will vote for that class. For this purpose we have defined BWAV, a voting system where each agent has 1 vote that can be fractionally assigned to several solution classes. For example, for a specific problem, an agent can vote 0.4 for a solution class and 0.6 for another solution class.

Formally, let $\mathcal{A}^c$ be a set of agents that have individually performed the Retrieve process for P and want to aggregate their individual collections of SERs using the BWAV voting system. Let $\mathcal{R}^c = \{\mathbf{R}_1, ..., \mathbf{R}_m\}$ be the set of SERs built by the n agents in $\mathcal{A}^c$ to solve the problem P. Notice that each agent is allowed to submit one or more SERs (see Section 3.4): in fact, an agent will submit as many SERs as different solution classes are present in the retrieved cases to solve P. Let $\mathcal{R}_{A_i} = \{\mathbf{R} \in \mathcal{R}^c | \mathbf{R}.A = A_i\}$ be the subset of SERs of $\mathcal{R}$ created by the agent A_i to solve problem P. The vote of an agent $A_i \in \mathcal{A}^c$ for a solution class $S_k \in \mathcal{S}$ is the following:

$$Vote(S_k, P, A_i) = \begin{cases} \frac{\mathbf{R}.E}{c+N} & \text{If } \exists \mathbf{R} \in \mathcal{R}_{A_i} | \mathbf{R}.S = S_k, \\ 0 & \text{otherwise.} \end{cases} \tag{4.1}$$

where c is a normalization constant that in our experiments is set to 1 and $N = \sum_{\mathbf{R} \in \mathcal{R}_{A_i}} \mathbf{R}.E$ is the total number of cases retrieved by A_i. Notice that if an agent A_i has not created a SER for a solution class S_k, then the vote of A_i for S_k will be 0. However, if A_i has created a SER for S_k, then the vote is proportional to the number of cases found endorsing the class S_k, i.e. $\mathbf{R}.E$.

To understand the effect of the constant c we can rewrite the first case of Equation 4.1 as follows (assume that $\mathbf{R}$ is the SER built by A_i for the solution class S_k):

$$Vote(S_k, P, A_i) = \frac{\mathbf{R}.E}{N} \times \frac{N}{c+N}$$

Since N is the total number of cases retrieved by A_i, the first fraction represents the ratio of the retrieved cases endorsing solution S_k with respect to N (the total number of cases retrieved by A_i). The second fraction favors the agent that has retrieved more cases, i.e. if A_i has only retrieved one case, and it is endorsing S_k, then the vote of A_i for S_k will be $Vote(S_k, P, A_i) = \frac{1}{1+1} = 0.5$; moreover, if the number of retrieved cases is 3 (and all of them endorsing S_k), then the vote is $Vote(S_k, P, A_i) = \frac{3}{1+3} = 0.75$. Notice that the sum of fractional votes casted by an agent is bound to 1, but in fact it is always less than 1 and, the more cases retrieved, the closer to 1. Finally, notice that if $c = 0$ the sum of votes is always 1.

We can aggregate the votes of all the agents in $\mathcal{A}^c$ for one class by computing the ballot for that class:

$$Ballot(S_k, P, \mathcal{A}^c) = \sum_{A_i \in \mathcal{A}^c} Vote(S_k, P, A_i)$$

and therefore, the winning solution class is the class with more votes in total, i.e.:

$$Sol(\mathcal{S}, P, \mathcal{A}^c) = \arg\max_{S_k \in \mathcal{S}} Ballot(S_k, P, \mathcal{A}^c) \tag{4.2}$$

This voting system is a variation of *Approval Voting* [15]. In approval voting, each voter defines a subset of preferred (approved) classes with respect to the rest of (non-approved) classes. The set of approved classes are the classes that the voter votes for. All the votes for the approved classes have the same weight, i.e. a voter cannot assign weights to their votes (as in BWAV). Formally, Approval Voting can be defined as:

$$ApprovalVote(S_k, P, A_i) = \begin{cases} 1 & \text{If } \exists \mathbf{R} \in \mathcal{R}_{A_i} | \mathbf{R}.S = S_k, \\ 0 & \text{otherwise.} \end{cases} \tag{4.3}$$

i.e. an agent votes for all the classes for which there is at least one case in the set of retrieved cases with that class. Moreover, all the votes for the voted classes have the same weight, 1.

There are two differences between Approval Voting and BWAV: the first one is that in BWAV agents can weight the votes for each class, and the second one is that in BWAV the sum of those weights, for any agent, has an upper bound of 1.

We can now specialize the Committee Collaboration Strategy with the BWAV voting system as follows.

Definition 4.3.1. *The* BWAV Committee Collaboration Strategy *(CCS_{BWAV}) is a collaboration strategy $\langle I_{CCS}, D_{BWAV} \rangle$, where I_{CCS} is the CCS interaction protocol shown in Figure 4.2 and D_{BWAV} is the BWAV voting system used to aggregate the evidence gathered by the individual agents into a global prediction.*

The Committee Collaboration Strategy can be also specialized to be used with any other voting system (such as Approval Voting). In the remainder of this monograph we will use CCS_{BWAV} to refer to the Committee Collaboration Strategy using the BWAV system, CCS_{AV} for the Committee Collaboration Strategy using the Approval Voting system, and in general CCS_V for any voting system V that we can define. When we write simply CCS, we are referring to CCS_{BWAV}, since BWAV is the voting system that we have chosen to be used by the agents in our experiments. However, we have also considered other voting systems such as Approval Voting or Majority Voting. Section 4.5.4 defines several voting systems that are experimentally compared with BWAV.

4.4 Characterizing Committees

One of the goals of our work is to determine when collaboration is beneficial, i.e. when an individual agent can achieve a higher classification accuracy by

convening a committee than working individually. The performance of a committee strongly depends on the case bases of the individual agent members of the committee. One of the goals of our work is to study the relationship between the content of the case bases of the individual agents in a committee and the classification accuracy of the committee. For this purpose, in this section we will present a characterization of individual case bases and then we will characterize committees of CBR agents. This characterization will be useful in the experiments section in order to find dependencies between committee characteristics and classification accuracy

4.4.1 Individual Case Base Characterization

Let us start by characterizing the contents of the case base C_i of a single agent A_i. A case base can be characterized by several measures, but we will concentrate on two basic ones, namely *Case Base Completeness* and *Case Base Bias*.

4.4.1.1 Case Base Completeness

Case Base Completeness refers to the degree in which a problem space or a data set is covered by the cases contained in a case base.

Before defining Case Base Completeness, let us clarify some concepts. A problem space $\mathcal{P}$ is the (finite or infinite) space that contains all the problems that can be described in a specific application domain. A data set $\mathcal{D}$ can be described as a set of cases $\mathcal{D} = \{c = \langle P, S\rangle | c.P \in \mathcal{P}'\}$ that have been built by labelling a sample of problems $\mathcal{P}' \subseteq \mathcal{P}$ of a problem space with their respective solutions. Moreover, we will write $\mathcal{D}.P$ to refer to the sample of problems from which a data set has been constructed.

Case Base Completeness can be defined with respect to a data set $\mathcal{D}$ or with respect to a problem space $\mathcal{P}$.

Definition 4.4.1. Case Base Completeness $\mathbb{C}$ *of a case base C_i with respect to a data set $\mathcal{D}$ of size M is:*

$$\mathbb{C}(C_i) = \frac{\#(C_i)}{M}$$

where $\#(C_i)$ represents the number of cases in C_i.

When the case base contains all the cases in the data set $\mathbb{C}(C_i) = 1$ and, as the number of cases in the case base diminishes, Case Base Completeness approaches 0. Notice that when computing Case Base Completeness with respect to a data set $\mathcal{D}$, a case base cannot contain a case not belonging to the data set, i.e. $C_i \subseteq \mathcal{D}$.

Definition 4.4.2. Case Base Completeness $\mathbb{C}$ *of a case base C_i with respect to a problem space $\mathcal{P}$ and with respect to a similarity function s defined over $\mathcal{P}$ is:*

$$\mathbb{C}(C_i) = E_{P \in \mathcal{P}}(\max(\{s(P, c.P) | c \in C_i\}))$$

where $s(P, c.P)$ is a similarity function between a problem P and a case c, and $E_{P \in \mathcal{P}}(x)$ represents the mathematical expectation of x over all possible problems P in the problem space $\mathcal{P}$.

The interpretation of this measure of Case Base Completeness for infinite domains is "the expected similarity of the most similar case for an arbitrary problem in the domain". Specifically, Case Base Completeness with respect to a problem space $\mathcal{P}$ measures the following: let P be a problem in the problem space $\mathcal{P}$, if we measure $s' = \max(\{s(P, c.P) | c \in C_i\})$, we obtain the similarity value of the most similar case in the case base C_i. If s' is a value close to 1, the closest case in C_i is very similar to P and we can say that the area of the problem space $\mathcal{P}$ surrounding P is well covered. However, if s' is a value close to 0, the closest case in C_i is very different to P and we can say that there is a "hole" in the case base C_i in the area of the problem space surrounding P since there are no cases with high similarity to P in C_i.

Definition 4.4.2 computes the expectation of s' over all the problem space, i.e. the average similarity of the most similar case for all the problems in the problem space. If Case Base Completeness has a value close to 1, the closest case in C_i for any problem in the problem space is expected to have a high similarity with the problem, and therefore we can say that all the problem space is well covered. Lower values of Case Base Completeness indicate that there are areas where the expected closest case in C_i has a lower similarity with the problem, i.e. areas that have a "competence hole". Case Base Completeness estimates the size of the areas with competence holes (the lower the Case Base Competence, the larger the competence hole areas).

Notice that if the similarity measure is defined to return values in the interval $[0, 1]$ the Case Base Completeness measure will also return values in $[0, 1]$.

Moreover, since $\mathcal{P}$ may be an infinite or a very large problem space, computing the exact value of the expectation may not be feasible. However, the expectation can be approximated by taking a finite sample $\mathcal{P}' \subseteq \mathcal{P}$ and computing the Case Base Completeness as follows:

$$\mathbb{C}(C_i) \simeq E_{P \in \mathcal{P}'}(\max(\{s(P, c.P) | c \in C_i\}))$$

Moreover, if we consider that all the problems in the sample $\mathcal{P}'$ are equiprobable, and therefore have probability $\frac{1}{\#(\mathcal{P}')}$, the previous expression can be reduced to:

$$\mathbb{C}(C_i) \simeq \frac{1}{\#(\mathcal{P}')} \sum_{P \in \mathcal{P}'} (\max(\{s(P, c.P) | c \in C_i\}))$$

i.e. the average of the similarities between each problem $P \in \mathcal{P}'$ and the most similar case in the case base C_i.

Albeit both definitions (Definition 4.4.1 and Definition 4.4.2) are not equivalent, the important thing is that Case Base Completeness represents the degree in which the problem space is covered in a case base. Notice that Case Base Completeness defined by Definition 4.4.1 defines the Case Base Completeness

with respect to an finite data set, i.e. the completeness is 1 when a case base contains all the examples in the data set. However, Definition 4.4.2 defines the Case Base Completeness with respect to an infinite problem space (i.e. it defines how well a problem space is covered by a case base) and depends on the specific similarity measure s used.

In this monograph we will only always compute completeness with respect to a concrete data set, so we will use the Definition 4.4.1 of Case Base Completeness.

4.4.1.2 Case Base Bias

Case base Bias refers to the degree in which a case base has some areas of the problem space undersampled (while other parts may be oversampled).

Since Case Base Bias involves computing whether there are undersampled or oversampled parts in the problem space, we will define the Case Base Bias with respect to any given partition $\Pi = \{\pi_1, ..., \pi_m\}$ of the problem space $\mathcal{P}$, such that $\cup_{\pi_i \in \Pi} \pi_i = \mathcal{P}$ and $\forall_{1 \leq k \leq m, 1 \leq j \leq m, k \neq j} \pi_k \cap \pi_j = \emptyset$. This partition can be defined explicitly (where each part π_i is an explicit list of examples) or implicitly (where each part π_i is defined by some property, i.e. π_i is defined as the set of problems that satisfy some property). The implicit definition is useful when the problem space is infinite or too large and making an explicit partition of the space is not feasible.

Definition 4.4.3. Case Base Bias $\mathbb{B}$ *of a case base* C_i *with respect to a partition* Π *of a data set* $\mathcal{D}$ *of size* M *is:*

$$\mathbb{B}_{\Pi}(C_i) = \sqrt{\sum_{m}^{k=1} \left(\frac{\#(\{c \in C_i | c.P \in \pi_k\})}{\#(C_i)} - \frac{\#(\pi_k)}{M} \right)^2}$$

To define Case Base Bias with respect to a problem space $\mathcal{P}$, we need to substitute the ratio $\frac{\#(\pi_k)}{M}$ (that cannot be computed directly if $\mathcal{P}$ is infinite) by $Prob(P \in \pi_k)$, i.e. the probability that a random problem P belongs to a given part π_k.

Definition 4.4.4. Case Base Bias $\mathbb{B}$ *of a case base* C_i *with respect to a partition* Π *of a problem space* $\mathcal{P}$ *is:*

$$\mathbb{B}_{\Pi}(C_i) = \sqrt{\sum_{m}^{k=1} \left(\frac{\#(\{c \in C_i | c.P \in \pi_k\})}{\#(C_i)} - Prob(P \in \pi_k) \right)^2}$$

Notice that Case Base Bias is zero when the ratio of cases in each part is the same in the case base C_i than in the problem space $\mathcal{P}$.

Figure 4.3 shows the graphical interpretation of Case Base Bias for a partition with 3 parts $\Pi = \{\pi_1, \pi_2, \pi_3\}$. Two vectors are shown in the figure, one represents the distribution of cases among the parts of a partition Π in a data set $\mathcal{D}$ and the other one represents the distribution of cases in a case base C_i.

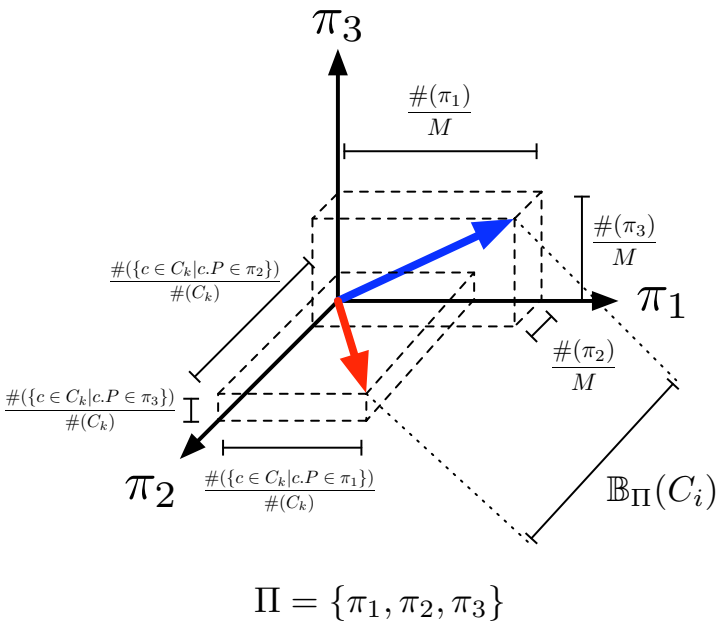


Figure 4.3: Graphical Visualization of Case Base Bias.

Each axis represents the number of cases in a part π_i. The larger the number of cases in C_i or in $\mathcal{P}$ in a part π_i, the larger the component of its vector in the axis representing the part π_i will be. Notice that $\frac{\#(\{c \in C_k | c.P \in \pi_k\})}{\#(C_k)}$ is exactly the component in the axis of the part π_k of the vector representing C_i and $\frac{\#(\pi_k)}{M}$ represents the component in the axis of the part π_k of the vector representing $\mathcal{D}$. Therefore, Case Base Bias measures exactly the distance between the extremes of these two vectors. Therefore, if m is the number of parts, Case Base Bias lies in the interval $[0, \sqrt{m}]$.

Moreover, Case Base Bias can be computed with respect to any partition Π, but we will focus on a specific kind of Case Base Bias ($\mathbb{B}_\mathcal{S}$) in which the partition is defined by the correct solution class of the problems: $\Pi_\mathcal{S} = \{\pi_1, ..., \pi_K\}$, where π_i contains all the problems with correct solution class S_i and only problems with correct solution class S_i.

Definition 4.4.5. Case Base Bias with respect to the solution classes $\mathbb{B}_\mathcal{S}$ *of a case base* C_i *is the Case Base Bias of* C_i *computed with respect to the partition* $\Pi_\mathcal{S}$.

Let us illustrate the concept of $\mathbb{B}_\mathcal{S}$ with an example. Imagine an artificial data set consisting in 8 examples $\mathcal{P} = \{P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8\}$ with two possible solution classes $\mathcal{S} = \{S_1, S_2\}$, such that four problems have solution class S_1 (P_1, P_2, P_3, P_4) and four problems have solution class S_2 (P_5, P_6, P_7, P_8).

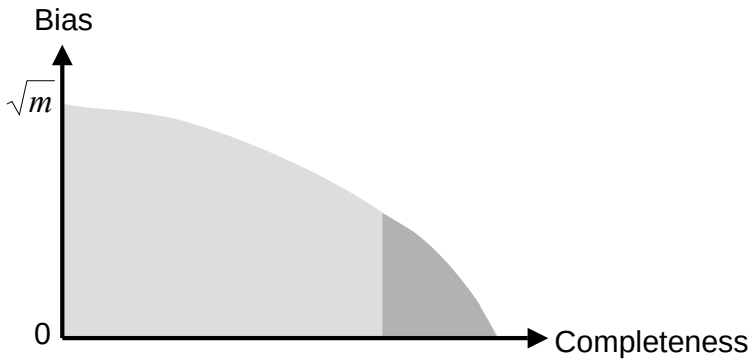


Figure 4.4: Space of possible different characterizations of individual case bases using completeness and bias.

The partition based on solution classes is therefore $\Pi_{\mathcal{S}} = \{\pi_1 = \{P_1, P_2, P_3, P_4\}, \pi_2 = \{P_5, P_6, P_7, P_8\}\}$. Imagine now a case base $C_1 = \{P_1, P_2, P_5, P_6\}$ of an agent A_1, and a case base $C_2 = \{P_1, P_5, P_6, P_7\}$ of an agent A_2. If we compute their $\mathbb{B}_{\mathcal{S}}$ values, we obtain:

$$\mathbb{B}_{\mathcal{S}}(C_1) = \sqrt{\left(\frac{2}{4} - \frac{4}{8}\right)^2 + \left(\frac{2}{4} - \frac{4}{8}\right)^2} = 0$$

$$\mathbb{B}_{\mathcal{S}}(C_2,) = \sqrt{\left(\frac{1}{4} - \frac{4}{8}\right)^2 + \left(\frac{3}{4} - \frac{4}{8}\right)^2} = \sqrt{0.125} = 0.354$$

The case base C_1 has half of the cases with solution S_1 an half of the cases with solution S_2 (as in the complete data set) and therefore has a Case Base Bias of 0. However, in the case base C_2 only one fourth of the cases have solution S_1 and three fourths have solution S_2, while in the complete data set, the distribution of cases per class is one half and one half. Therefore, the Case Base Bias is different than zero, and it is in fact 0.354.

In the rest of this monograph we will use the term Case Base Bias to refer to $\mathbb{B}_{\mathcal{S}}$ of Definition 4.4.5.

4.4.1.3 Case Base Characterization Space

Figure 4.4 shows the space of possible different characterizations of individual case bases using the notions of completeness and bias. The grey area shows the range of possible values that bias and completeness can take. Not all the possible combinations of Completeness and Bias are possible, as Case Base Completeness increases, Case Base Bias cannot take arbitrarily high values since if the

Completeness of a case base is high, the case base cannot have an arbitrarily low sampled areas. Figure 4.4 shows the qualitative relationship between Case Base Bias and Case Base Completeness.

Moreover, case bases with high completeness (i.e. that have a good coverage of the problem space) and with low bias achieve higher classification accuracy values. If a case base falls in the area painted with darker grey in Figure 4.4, classification accuracy is expected to be high. However, it is not clear that the best case base is that with Case Base Completeness 1 and Case Base Bias 0 (the bottom-right extreme in the figure) since, once a case base has a good sample of the problem space, adding an arbitrarily large number of cases to it (i.e. increasing even more the Completeness) does not necessarily increase classification accuracy, while certainly reduces the efficiency of the CBR system (see Section 2.2.1.1).

4.4.2 Committee Characterization

Let us now extend the definitions presented for individual case bases to characterize committees of CBR agents. To characterize a committee of CBR agents, we are going to introduce three new measures: *Committee Completeness*, *Committee Bias* and *Committee Redundancy*.

4.4.2.1 Committee Completeness

Committee Completeness is measured based on the idea that a case c *belongs* to a committee $\mathcal{A}^c$ if at least one of the agents in the committee has c in its case base. Let us call $\mathcal{C}^c$ to the collection of case bases of the agents in $\mathcal{A}^c$. Therefore, the collection of all the cases that belong to a committee is $U^c = \cup_{C_k \in \mathcal{C}^c} C_k$.

Definition 4.4.6. *The* Committee Completeness *of a committee $\mathcal{A}^c$ with respect to a data set $\mathcal{D}$ of size M is:*

$$\mathbb{C}(\mathcal{A}^c) = \frac{\#(U^c)}{M}$$

Definition 4.4.7. *The* Committee Completeness *of a committee $\mathcal{A}^c$ with respect to a problem space $\mathcal{P}$ is:*

$$\mathbb{C}(\mathcal{A}^c) = E_{P \in \mathcal{P}}(\max(\{s(P, c.P) | c \in U^c\}))$$

In the rest of this monograph we will always measure Case Base Completeness with respect to a specific data set, and therefore we will use Definition 4.4.6 for Committee Completeness.

4.4.2.2 Committee Bias

Committee Bias is defined as the average Case Case Bias of the member agents of the committee.

Definition 4.4.8. *The* Committee Bias *of a committee $\mathcal{A}^c$ with n agents with respect to a partition Π of a problem space $\mathcal{P}$ is:*

$$\mathbb{B}_{\Pi}(\mathcal{A}^c) = \frac{1}{n} \sum_{C_i \in \mathcal{C}^c} \mathbb{B}_{\Pi}(C_i)$$

Moreover, in the rest of this monograph we will focus in the Committee Bias $\mathbb{B}_{\mathcal{S}}$ computed with respect to the partition $\Pi_{\mathcal{S}}$ induced by the solution classes:

Definition 4.4.9. Committee Bias with respect to the solution classes $\mathbb{B}_{\mathcal{S}}$ *of a committee $\mathcal{A}^c$ of n agents is the Committee Bias computed with respect to the solution class partition $\Pi_{\mathcal{S}}$.*

4.4.2.3 Committee Redundancy

Committee Redundancy is a measure that only applies to committees and cannot applied to individual case bases. Intuitively, redundancy measures the degree of overlapping existing between the case bases of the agent members of a committee. If the case bases of the individual agents are identical, redundancy is maximum since the case bases of the individual agents are completely overlapped. When the case bases of the individual agents are disjunct, i.e. there is no case shared by two agents' case bases, redundancy is zero.

Definition 4.4.10. *The* Committee Redundancy *of a committee $\mathcal{A}^c$ with n agents and with a collection of case bases $\mathcal{C}^c$ is:*

$$\mathbb{R}(\mathcal{C}^c) = \frac{(\sum_{C_i \in \mathcal{C}^c} \#(C_i)) - N}{N * (n-1)}$$

where $N = \#(U^c)$, i.e. the total number of different cases belonging to a committee $\mathcal{A}^c$.

For example, imagine a committee composed of 2 agents $\mathcal{A}^c = \{A_1, A_2\}$ with case bases: $C_1 = \{c_1, c_2, c_3\}$ and $C_2 = \{c_2, c_4, c_5\}$. In order to compute the Committee Redundancy we have that the total number of cases in the committee is $N = \#(C^c) = \#(\{c_1, c_2, c_3, c_4, c_5\}) = 5$, and therefore:

$$\mathbb{R}(\{C_1, C_2\}) = \frac{(\#(C_1) + \#(C_2)) - N}{N * (n-1)} =$$

$$\frac{(3+3) - 5}{5 * (2-1)} = 0.2$$

The redundancy is different than zero since the case c_2 is shared by the two agents.

With these three measures (Committee Completeness, Committee Bias and Committee Redundancy) we can now define the committee characterization space.

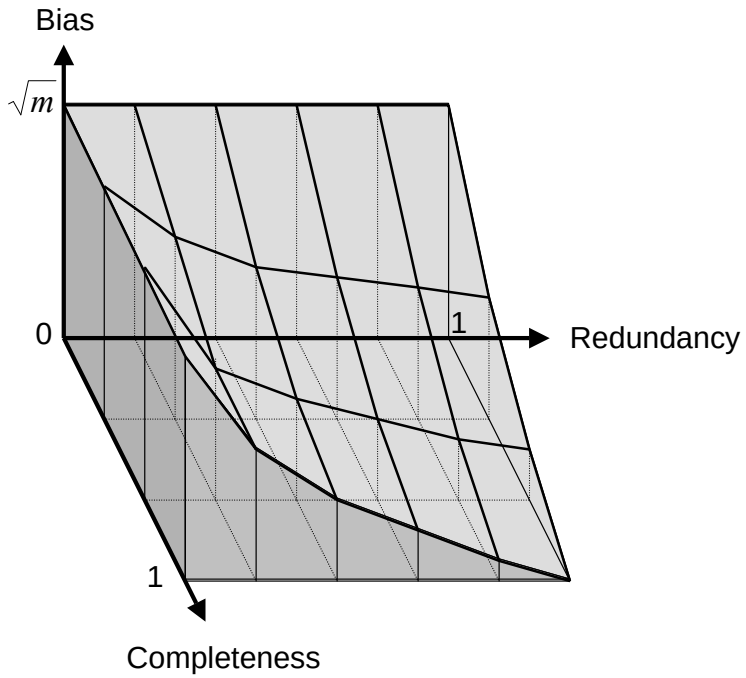


Figure 4.5: *Ensemble Space*: the space of possible committees characterized by Committee Completeness, Committee Bias and Committee Redundancy.

4.4.2.4 Ensemble Space

Figure 4.5 shows the space of possible committees characterized by Committee Completeness, Committee Bias and Committee Redundancy, that we call the *Ensemble Space*. The grey space shows the possible values that the Committee Completeness, Committee Bias and Committee Redundancy can take. If we compute the Completeness, Bias and Redundancy of a specific committee we will obtain a point in the Ensemble Space.

Notice in Figure 4.5 that not all the combinations of values are possible; for instance, if Committee Redundancy and Committee Completeness are both high, the value of Committee Bias cannot be arbitrarily high. The reason is that a high Committee Completeness means that the problem space is well covered with the cases belonging to the committee, and a high Committee Redundancy means that there is a high overlapping among the agents' case bases. Therefore, for a committee to have high Committee Completeness and high Committee Redundancy, each individual agent must also have high individual Case Base Completeness, which implies that their individual Case Base Bias cannot be arbitrarily high, and as the Committee Bias is the average of the individual

Case Base Bias values, the Committee Bias cannot be arbitrarily high either.

Moreover, we are interested in studying the relation between Committee Completeness, Committee Bias and Committee Redundancy and the committee classification accuracy. While in the individual case base characterization the area with high completeness and low bias achieves the higher classification accuracy values, which area is optimal in a committee is not obvious. In the following section we present experiments with committees that fall in different parts of this Ensemble Space, and we analyze the achieved results to determine in which regions of the Ensemble Space fall the committees that achieve higher classification accuracy values.

4.5 Experimental Evaluation

In this section we are going to empirically evaluate the Committee Collaboration Strategy (CCS). For this purpose, we are going to evaluate the classification accuracy (i.e. the percentage of correctly solved problems) of agents using the Committee Collaboration Strategy in several scenarios: using several CBR methods, several data sets, convening committees of several sizes, and committees falling in several areas of the Ensemble Space. Moreover, the accuracy achieved by CCS will be compared with the classification accuracy achieved by agents solving problems individually (without using any collaboration strategy).

We have used three different data sets to evaluate the committee collaboration strategy: *soybean*, a well known propositional data set, *zoology*, another propositional data set from the UCI machine learning repository, and *marine sponges*, a complex relational data set [9] not present in the UCI machine learning repository. Sponges have a complex structure, making them amenable to build complex justifications. The soybean data set consists of 307 examples, each one with 35 attributes (some of them with missing values), and there are 19 possible solution classes. The zoology data set consists of 101 examples, each one with 17 attributes and 7 solution classes. The sponges data set consists of 280 examples, each one with between 10 and 50 attributes (depending on its structure), and there are 3 solution classes.

In order to test the generality of the committee collaboration strategy, we have performed experiments using various learning methods. Specifically, we have made experiments with agents using LID Nearest Neighbor (1-NN) and 3-Nearest Neighbor (3-NN). If we recall the classification of learning methods that we made in Section 2.1.3, LID is a global method (and therefore is unstable), and 1-NN and 3-NN are local methods (and therefore, stable methods). Thus, we have made experiments with both type of methods to test the effect of the Committee Collaboration Strategy in each class of methods.

Moreover, we never mix agents using different learning methods in our experiments, i.e. during an experimental run all the agents use the same learning method (although the committee collaboration strategy allows that each agent use its own classification method as long as a SER can be built). The reason is that by mixing agents with different learning methods the error correlation

among the agents will diminish, thus (expectedly) increasing the classification accuracy of CCS. While this effect may improve overall accuracy, it will interfere in our evaluation of the improvement obtained by the collaboration of the members of a committee. Thus, we have decided not to mix different learning methods in order to be sure that all the improvements achieved in the experiments are due to CCS, and we will use agents with the same lazy learning method in the experiments reported henceforth.

In an experimental run, training cases from a data set are randomly distributed among the agents. Once all the cases are distributed among the agents, the test phase starts. During the test phase, for each problem P in the test cases an agent $A_i \in \mathcal{A}$ is randomly chosen, and P is sent to A_i. A_i will then use the Committee Collaboration Strategy to solve P.

All the results presented in this section are the average of 5 runs of a 10 fold cross validation. Therefore, in each run, the 90% of the cases of a test set are used as the training cases, and the other 10% are used as the test cases.

In order to evaluate the Committee Collaboration Strategy, we are going to present four sets of experiments, presented in the following four sections. The first set of experiments, presented in Section 4.5.1, will constitute the basic experiments to evaluate CCS. In these experiments, we are going to evaluate committees with Committee Completeness $\mathbb{C} = 1$, Committee Bias $\mathbb{B}_{\mathcal{S}} = 0$ and Committee Redundancy $\mathbb{R} = 0$, i.e. all the cases in the training set will be distributed among the individual agents case bases without repetitions. When we have a set of m training cases and $\mathcal{MAC}$ system composed of n agents, each agent will receive m/n cases in average. Notice that as we increase the number of agents the individual case base size will decrease. Individual accuracy is expected to diminish when the individual case base size diminishes (i.e. when the number of agents in an experiment increases). Therefore, in the experiments where we have used a large number of agents, classification accuracy is expected to be lower because of our experimental settings. In the second and third sets of experiments, presented in section 4.5.2 and 4.5.3, we will perform experiments with committees having different Committee Redundancy and Committee Bias in order to explore the Ensemble Space. The goal of these experiments is to determine which areas of the ensemble space contain committees with higher classification accuracy values. Finally, the fourth set of experiments, presented in Section 4.5.4, defines several voting systems different from BWAV and experimentally compares them.

4.5.1 Committee Evaluation under Uniform Conditions

In this section we are going to compare the classification accuracy of agents using the Committee Collaboration Strategy (CCS) and agents solving problems individually under what we call *uniform conditions*, i.e. when Committee Completeness $\mathbb{C} = 1$, Committee Bias $\mathbb{B}_{\mathcal{S}} = 0$ and Committee Redundancy $\mathbb{R} = 0$. To perform these experiments, we have proceeded in the following way: to experiment with a committee of n agents, we create a $\mathcal{MAC}$ system of n agents and randomly distribute the cases of the training set among the agents so that

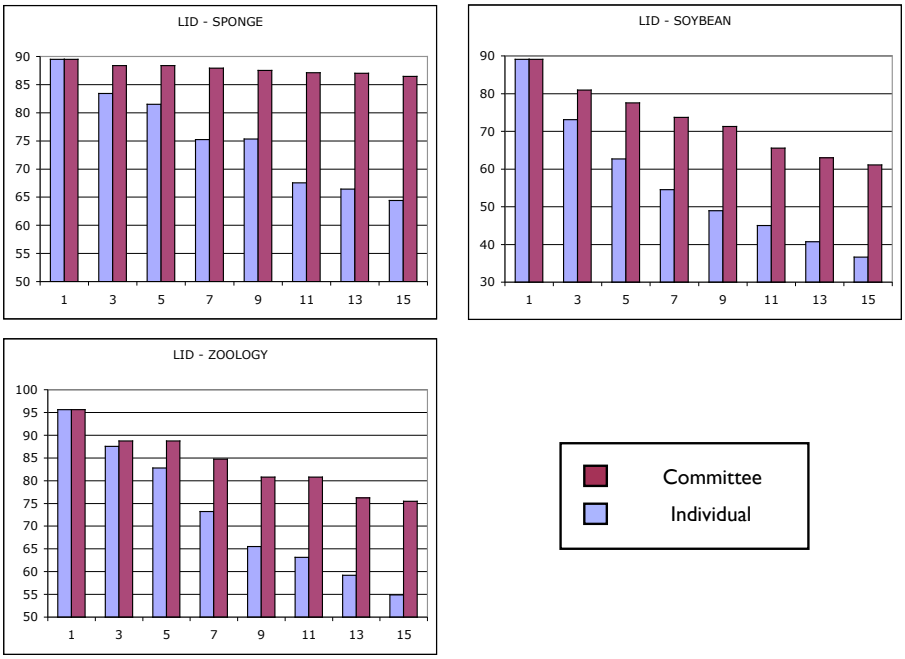


Figure 4.6: Classification accuracy comparison between agents using LID to solve problems individually and with the committee collaboration strategy.

$\mathbb{C} = 1$, $\mathbb{B}_{\mathcal{S}} = 0$, and $\mathbb{R} = 0$. Then, to evaluate the accuracy of agents, test cases arrive to a randomly chosen agent in the system, and then that agent solves the problem. When CCS is used to solve the problem, a committee composed of all the agents in the $\mathcal{M}$AC is convened (since in our experimentation all the agents agree to join the committee and therefore in a $\mathcal{M}$AC system of n agent, the committee convened is composed of exactly n agents).

Figures 4.6, 4.7 and 4.8 present the classification accuracy achieved by agents both individually and using the Committee collaboration strategies for 3 lazy learning methods (LID, 1-NN and 3-NN respectively).

Figure 4.6 shows the results for agents that use LID as lazy learning method. The vertical axis shows the average classification accuracy, and the horizontal axis shows the number of agents that compose the system. For each $\mathcal{M}$AC system two bars are shown, the left one shows the classification accuracy of agents solving problems individually and the right one shows the classification accuracy for agents using CCS. In all the data sets and in all the $\mathcal{M}$AC systems, we can see that agents using CCS outperform agents solving problems individually. The difference in classification accuracy is specially noticeable when the agents' individual case bases sizes are small. For instance, in the sponge data set, individual agents in a $\mathcal{M}$AC system composed of 15 agents achieve an accuracy of about the 64% while agents using the committee collaboration strategy achieve

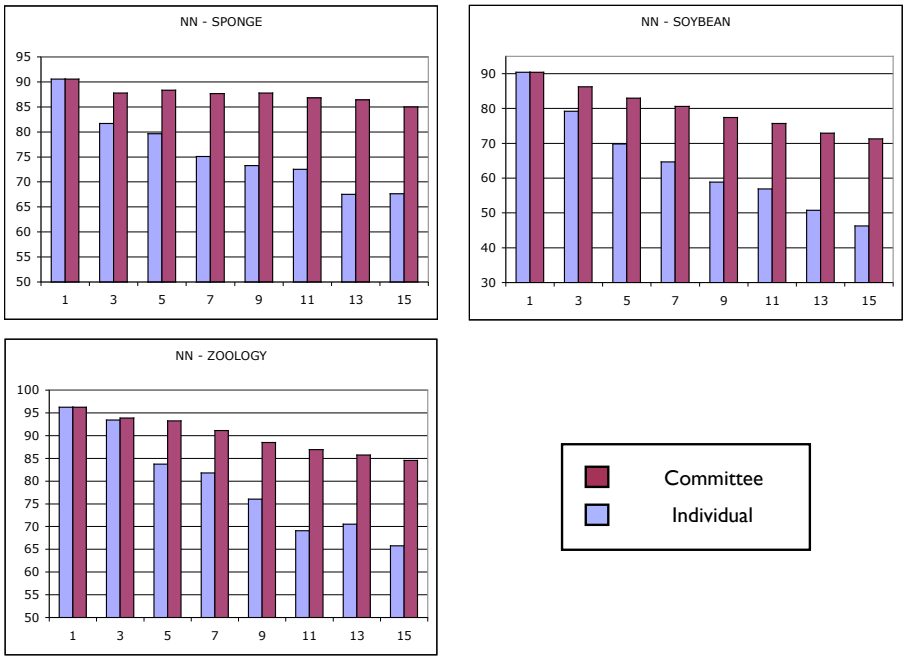


Figure 4.7: Classification accuracy comparison between agents using 1-Nearest Neighbor to solve problems individually and with the committee collaboration strategy.

an accuracy of about 87%. Therefore, when data is very fragmented among the individual agents, they have more incentive to collaborate by forming committees, since the difference in classification accuracy between the individual agents and CCS is greater (as happens in the experiments with a greater number of agents).

From this initial experiments we can draw two initial conclusions: the first one is that the ensemble effect works since the accuracy of CCS is always higher than the accuracy of the individual agents; and the second is that the gain in classification accuracy achieved by CCS thanks to the ensemble effect varies, since not all the committees of agents achieve the same accuracy gain. When the case bases of the individual agents are small (i.e. in the experiments with many agents), the difference in accuracy between CCS and the individual agents is greater than when the case bases of the individual agents are large (i.e. in experiments with few agents). This is because when agents have large case bases, their individual accuracy is already high, and the ensemble effect cannot improve much more that accuracy.

Notice that when each individual agent has very few cases (i.e. in the 11, 13 and 15 agents $\mathcal{M}$AC systems), the individual accuracies for the soybean and zoology data sets decrease drastically: it is below 40% in the soybean data set

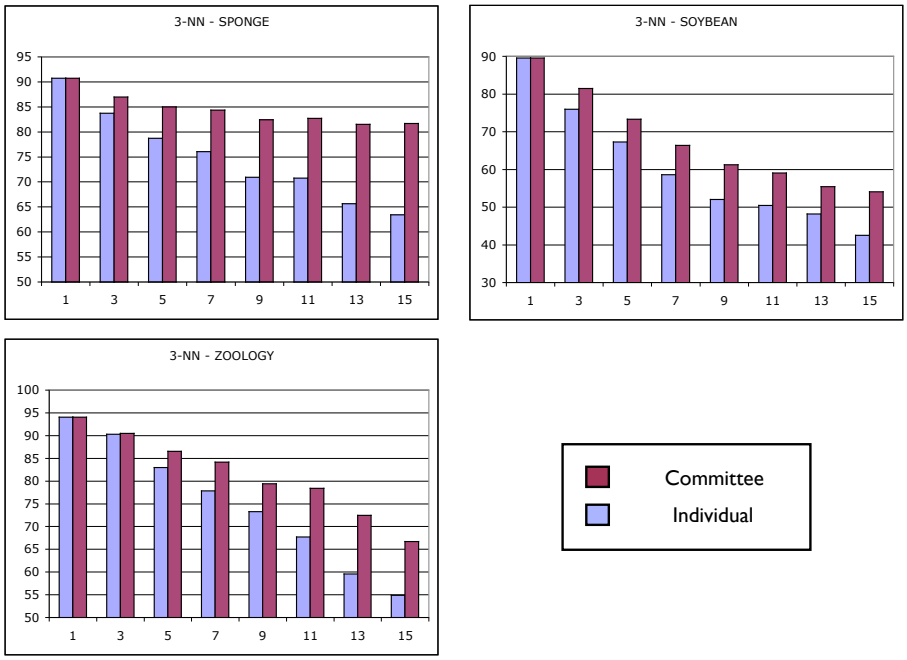


Figure 4.8: Classification accuracy comparison between agents using 3-Nearest Neighbor to solve problems individually and with the committee collaboration strategy.

and below 60% in the zoology data set. The reason is that when the number of agents in an experiment is large, their individual Case Base Completeness with respect to the data set is low (notice that although the Committee Completeness of all the committees evaluated in this section is $\mathbb{C} = 1$, this has nothing to do with the Case Base Completeness of the case bases of the individual agents). For example, with a 15 agents $\mathcal{MAC}$ system in the soybean data set, each agent has just 18.43 cases in average out of the 276 cases that the training set has for the soybean data set (having an average individual Case Base Completeness of $\mathbb{C} = 0.067$). Individual agents need a good sample of cases from the data set to have a good performance; therefore, when an individual case base is small the individual sample worsens, and therefore the individual accuracy decreases. Moreover, the global solution computed by a committee is computed as the aggregation of the SERs built by the individual agents. The estimated error of an ensemble (and therefore of a committee) [49] (presented in Section 2.1.1) is: $E = \bar{E} - \bar{A}$ (where $\bar{E}$ is the mean error of the individual classifiers and $\bar{A}$ is the *ambiguity*, that increases as error correlation decreases). Therefore, if the individual agents have a high classification error, $\bar{E}$ will be high and, unless the ambiguity $\bar{A}$ is high enough to compensate the high $\bar{E}$, the error of the committee E will also be high.

Agents	50	100	200	254
CB size	5.08	2.54	1.27	1.00
Individual	48.23	40.76	37.28	34.85
CCS	80.50	67.00	52.45	41.78

Table 4.1: Case base size and classification accuracy of agents solving problems individually and with CCS in very large $\mathcal{M}$AC systems.

Figure 4.7 shows the same experiments with agents that use a nearest neighbor (1-NN) method to solve problems. The results obtained are analogous to those of LID, and we can see that CCS always outperforms the individual agents. Specially when the individual agents have small case bases.

Finally, Figure 4.8 shows the results with agents that use 3-nearest neighbor (3-NN), again with the same results. Moreover, comparing the results obtained with 1-NN and with 3-NN, we can see that 3-NN obtains much worse results when the individual agents have small case bases. This is because k-NN methods do not work well with small case bases. To exemplify this result, just imagine an agent using 3-NN with a case base containing just 4 cases: two cases of class A, one case of class B and one case of class C. Clearly, that agent will *never* output class B or C as the solution for a problem, since there will never be a majority of B or C cases in the three nearest neighbors.

To conclude, we can say that the Committee Collaboration Strategy can help the agents in a $\mathcal{M}$AC system to improve their accuracy results. We have empirically demonstrated that the ensemble effect takes place in CCS since the accuracy of CCS is systematically higher than the accuracy of individual agents. Moreover, the increase between the classification accuracy of CCS and that of the individual agents is greater in systems where the individual CBR agents have small case bases and low individual accuracy. However, the classification accuracy of a committee composed by a lot of agents with small case bases is still lower than that of a committee composed of less agents but with larger case bases. Moreover, the following sections of this chapter precisely explore under which conditions the accuracy of a committee is high or low.

Moreover, the Committee Collaboration Strategy does not compromise the autonomy of each individual agent since an agent can always refuse to join a committee. Privacy conditions are also preserved, since cases are not shared (recall that in CCS is a strategy that performs distributed reuse and not distributed retrieval). Therefore, CCS allows agroup of CBR agents to collaborate for solving problems in a decentralized way, benefitting from the ensemble effect and without compromising the autonomy of the agents.

The next section presents experiments in scenarios where the data is very fragmented and each individual agent has very few cases.

4.5.1.1 Ensemble Effect with Inaccurate Agents

From the experiments reported in this section we can conclude that the smaller the individual case bases of the agents, the larger the difference in classification accuracy between the individual agents and CCS. We have performed further experiments to analyze how the ensemble effect evolves when data is distributed among a very large number of agents. For this purpose, we have performed experiments with agents using the LID method to solve problems in the sponge data set in $\mathcal{M}$AC systems composed of 50, 100, 200 and 254 agents (in the 254 agents $\mathcal{M}$AC system each agent only has one case in its case base).

Table 4.1 shows the case base size and classification accuracy of agents solving problems individually and with CCSfor systems composed from 50 to 254 agents. Agents in a system composed of 50 agents (with an average of 5.04 cases per agent) have an individual accuracy of 48.23% while the CCS accuracy is 80.5%. On the extreme, agents in a system with 254 agents (where each agent has only one case in its case base) have an average individual accuracy of 34.85% while the committee accuracy is 41.78%, that is exactly the classification accuracy of the *basic hypothesis*, i.e. predicting always the largest solution class (in the sponge data set, the largest solution class is S_k = *Hadromerida* since the 41.7% of the cases belong to that class).

These experimental results show that even in the extreme scenario with 254 agents, where each individual agent has only one case, the ensemble effect still takes place. In Section 2.1 we stated that the preconditions for the ensemble effect to take place were that the individual error of the classifiers must be lower than 0.5 and that the individual error of the classifiers must be uncorrelated, as stated by Hansen and Salamon [36]. Clearly, the individual error of the agents in our experiments with more than 50 agents is higher than 0.5 and the ensemble effect still takes place. The explanation is that Hansen and Salamon considered only the scenario where there are 2 possible solution classes. But in general, the average classification error of the individual agents must be lower than $1/K$ where K is the number of possible solution classes, i.e. they have to be more accurate than a random guesser. Notice that, as in the previous experiments the average error of the individual agents is always lower than 0.33 (and in the sponge data set that has 3 solution classes), the ensemble effect still should (and does) take place.

The next sections present experiments concerning committees in different areas of the Ensemble Space in order to analyze the ensemble effect in committees.

4.5.2 Committee Evaluation with Case Redundancy

In this set of experiments,we are going to evaluate how the performance of the Committee Collaboration Strategy is affected by the Committee Redundancy. For this purpose, we are going to perform experiment with committees with Committee Completeness $\mathbb{C} = 1$, Committee Bias $\mathbb{B} = 0$ and varying Committee Redundancy from $\mathbb{R} = 0$ to $\mathbb{R} = 1$.

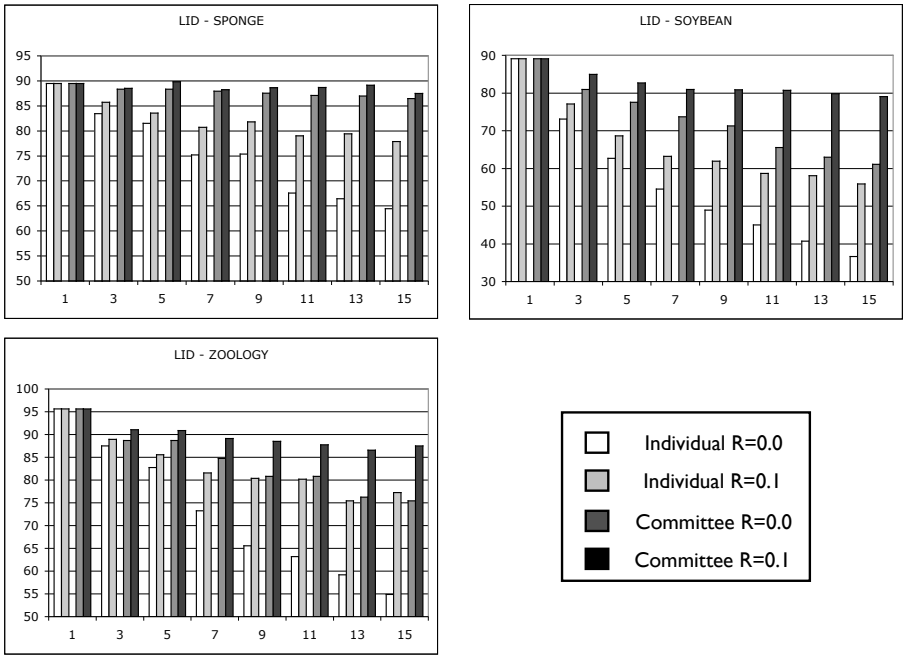


Figure 4.9: Redundancy effect on the Classification accuracy in agents using LID.

In order to test the effect of Committee Redundancy in the accuracy of CCS, we have repeated all the experiments presented in the previous section, with a degree of Committee Redundancy $\mathbb{R} = 0.1$. For instance, using the sponges data set in a $\mathcal{M}$AC system composed of 5 agents, each agent will have a case base with an average of 50.8 cases if $\mathbb{R} = 0.0$. However, for $\mathbb{R} = 0.1$ each agent will have a case base with an average of 71.12 cases. In the extreme, for $\mathbb{R} = 1.0$ each agent would have a case base with 254 cases, i.e. the whole training set. Moreover, we have also performed a set of experiments with a committee composed of 9 agents in the sponge data set using LID as the problem solving method, where we have measured the classification accuracy with Committee Redundancy values varying from 0 to 1 in steps of 0.05.

Figure 4.9 shows the results for agents that use LID as lazy learning method. The figure presents 4 different bars for each $\mathcal{M}$AC system: individual agents when the Committee Redundancy $\mathbb{R} = 0.0$, individual agents when the Committee Redundancy $\mathbb{R} = 0.1$, CCS with Committee Redundancy $\mathbb{R} = 0.0$ and CCS with Committee Redundancy $\mathbb{R} = 0.1$. Notice that when we say "individual agents when the Committee Redundancy is $\mathbb{R} = x$", we mean the accuracy of an individual agent member of a $\mathcal{M}$AC system such that if all the agents of the $\mathcal{M}$AC system form a committee, the Committee Redundancy of that committee is $\mathbb{R} = x$. Figure 4.9 shows that in $\mathcal{M}$AC systems with Committee Redundancy

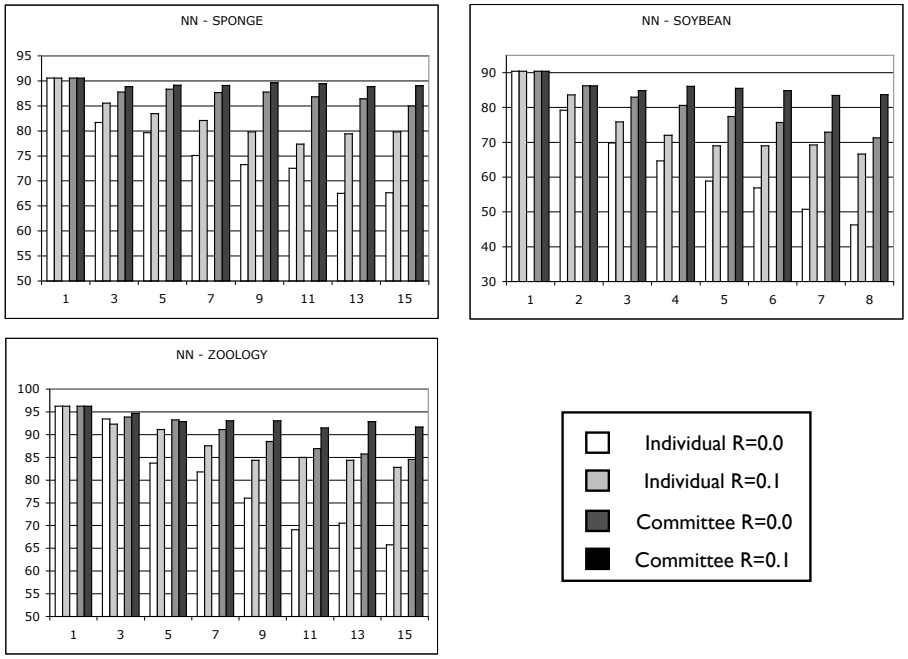


Figure 4.10: Redundancy effect on the Classification accuracy in agents using 1-NN.

$\mathbb{R} = 0.1$ the classification accuracy is systematically higher both for individual agents and for CCS. Specially in $\mathcal{M}$AC systems where individual agents have small case bases (i.e. $\mathcal{M}$AC systems with many agents), the increment in classification accuracy due to increasing redundancy is very noticeable. For instance, in a system composed of 15 agents in the sponges data set, individual agents with $\mathbb{R} = 0.0$ achieve a classification accuracy of 64.43% while with $\mathbb{R} = 0.1$ the classification achieved is 77.85%. The accuracy of CCS of 15 agents in the sponges data set is 86.45% with redundancy $\mathbb{R} = 0.0$ while it increases to 87.48% with $\mathbb{R} = 0.1$. Moreover, notice that the greater improvements in classification accuracy are achieved in the soybean and zoology data sets.

From this initial set of experiments we can draw an initial conclusion: in a committee with some redundancy ($\mathbb{R} = 0.1$) the individual agents have larger case bases (and therefore a better data sample) than the agents in a committee with no redundancy ($\mathbb{R} = 0.0$). Therefore, individual agents in the committee with $\mathbb{R} = 0.1$ have a greater individual classification accuracy. Recall that the decomposition of the error of a committee is $E = \bar{E} - \bar{A}$. The committee error E is lower in the committee with $\mathbb{R} = 0.1$ because the average individual agents' error $\bar{E}$ is lower than with $\mathbb{R} = 0.0$ and because the error correlation with $\mathbb{R} = 0.1$ does not increase with respect to $\mathbb{R} = 0.0$ (and therefore $\bar{A}$ does not decrease).

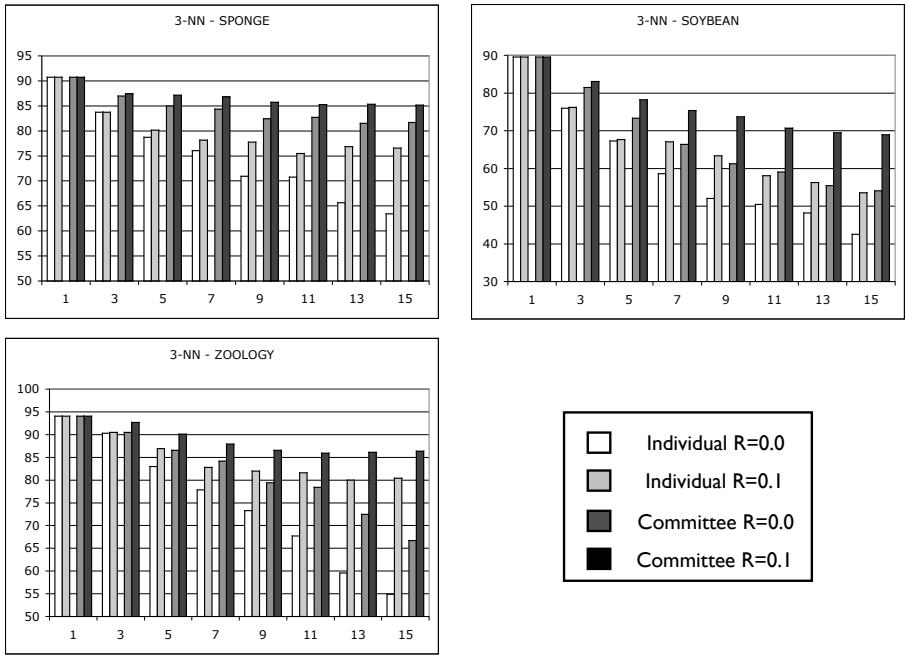


Figure 4.11: Redundancy effect on the Classification accuracy in agents using 3-NN.

Figure 4.10 shows the results for agents that use 1-NN as learning method. The figure shows that increasing the redundancy from $\mathbb{R} = 0$ to $\mathbb{R} = 0.1$ has the same effect on 1-NN than on LID(as shown in Figure 4.9). $\mathcal{M}$AC systems with Committee Redundancy $\mathbb{R} = 0.1$ always achieve higher accuracy values (both solving problems individually and with the Committee Collaboration Strategy) than $\mathcal{M}$AC systems with Committee Redundancy $\mathbb{R} = 0$. Moreover, the accuracy increment is larger in systems with many agents. The reason is that agents in a $\mathcal{M}$AC system with $\mathbb{R} = 0.1$ have larger case bases (and therefore a better data sample than in a $\mathcal{M}$AC system with $\mathbb{R} = 0.0$); clearly, in $\mathcal{M}$AC systems where individual agents have few cases, improving their data samples can boost individual classification accuracy (and therefore CCS accuracy).

Finally, Figure 4.11 shows the results for agents that use 3-NN as learning method. The figure shows that the accuracy improvement for a committee with $\mathbb{R} = 0.1$ with respect to $\mathbb{R} = 0.0$ is also larger in systems with many agents, where each individual agent has smaller case bases, and specially in the zoology data set. 3-NN performance is specially poor in the zoology data set, that is a small data set consisting only of 101 cases, when the individual agents have very small case bases.

As a summary of the previous results, we can say that with certain degree of redundancy data samples of the individual agents are better, and therefore

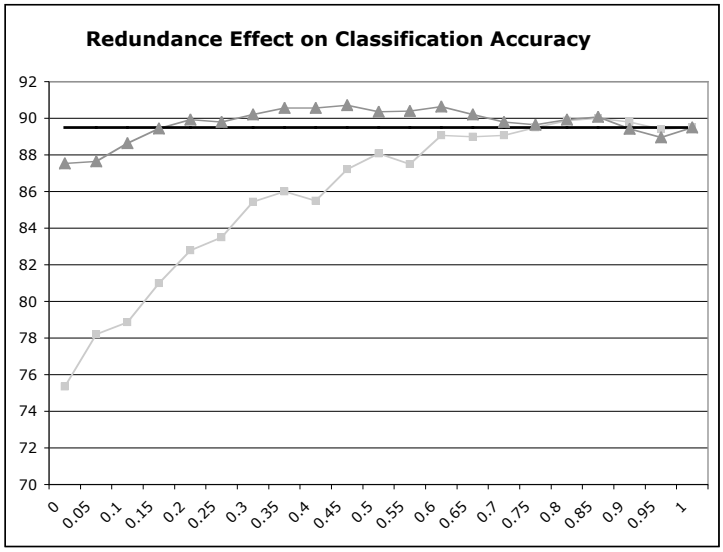


Figure 4.12: Redundancy effect in the classification accuracy of individual agents and in the Committee.

individual classification accuracy is higher than in $\mathcal{M}$AC systems without redundancy. Moreover, a high redundancy could cause (in principle) an increase in the error correlation among the individual agents and when error correlation increases, the ensemble effect diminishes. However, as we have seen for the experimental results, with Committee Redundancy $\mathbb{R} = 0.1$ the error correlation is not much greater than the error correlation with $\mathbb{R} = 0.0$ and the ensemble effect still works. Moreover, in order to determine what happens with accuracy and correlation as Committee Redundancy takes higher values, we have performed further experiments to analyze the effect of redundancy on the classification accuracy.

Figure 4.12 shows how the classification accuracy varies with Committee Redundancy values from 0 to 1 in a $\mathcal{M}$AC system composed of 9 agents of both the individual agents and of CCS for the sponges data set. The accuracy of a single agent with the whole case base is 89.5% (represented as an horizontal black line). Figure 4.12 shows that, as the Committee Redundancy grows from 0 to 1, the individual agents' accuracy systematically grows from 75.35% with Committee Redundancy $\mathbb{R} = 0.0$ to 89.5% with Committee Redundancy $\mathbb{R} = 1.0$. Notice that if $\mathbb{R} = 1.0$ all the agents in the system have all the cases in their case bases, and therefore their individual classification accuracy is identical and equal to that of a single agent with all the cases. The evolution of the CCS accuracy shows a more interesting behavior: when the degree of Committee Redundancy is not very high ($\mathbb{R} < 0.4$), the CCS accuracy also grows as the redundancy increases. Then, the accuracy of CCS remains stable until the

redundancy reaches high values ($\mathbb{R} \approx 0.6$), and then the accuracy decreases again. When the redundancy is maximum, $\mathbb{R} = 1.0$, the classification accuracy of CCS is identical to that of the individual agents, since all the agents are equivalent, i.e. all of them have all the cases.

Notice that for Committee Redundancy values between $\mathbb{R} = 0.2$ and $\mathbb{R} = 0.65$ (Figure 4.12) the classification accuracy of CCS is higher than the accuracy of a single agent with all the cases.

In order to explain this effect, let us recall again the decomposition of the error of a committee [49]: $E = \bar{E} - \bar{A}$. Increasing Committee Redundancy has two opposite effects: decreasing individual agents error that decreases the committee error, and increasing the correlation that increases the committee error. Figure 4.12 shows that when Committee Redundancy takes low values, the decrease of the committee error due to the decrease of the individual error compensates the (smaller) increase in error correlation. Moreover, when a certain degree of Committee Redundancy is reached, the increase on error correlation and the reduction of individual error reach an equilibrium where one compensates the other and the CCS error stabilizes. Indeed, it is in this plateau that the CCS accuracy surpasses that of a centralized single agent approach. Finally, when the Committee Redundancy approaches 1, the error correlation among the individual agents is too high and the ensemble effect decreases until reaching the accuracy of the individual agents when $\mathbb{R} = 1.0$).

The conclusion is that a certain degree of Committee Redundancy has a positive effect in the CCS accuracy. For low Committee Redundancy values, increasing Committee Redundancy also increases CCS accuracy. Moreover, at a certain value of Committee Redundancy, the CCS accuracy stabilizes and does not increase more as Committee Redundancy increases. For these Committee Redundancy values (where CCS accuracy is stabilized) the accuracy of CCS is higher than the accuracy of a single agent owning all the cases. However, as Committee Redundancy reaches the highest values (approaching 1), the accuracy of CCS decreases again, converging to the accuracy of a single agent owning all the cases. Therefore, in order to maximize the classification accuracy of a committee, a moderate degree of Committee Redundancy (not too high and not too low) is advisable.

4.5.3 Committee Evaluation with Case Base Bias

In this set of experiments, we will evaluate how the performance of the Committee Collaboration Strategy is affected by the Committee Bias. For this purpose, we have repeated the experiments presented in Section 4.5.1 with varying Committee Bias values. The classification accuracy achieved by the individual agents and CCS is presented.

A certain Committee Bias $\mathbb{B} = \beta$ does not have the same meaning in a data set than in another data set. Therefore, in the experiments presented in this section, we have used different Committee Bias values for each data set. For each data set we have chosen three values of Committee Bias representing three situations: low bias, medium bias and high bias. The values that we have

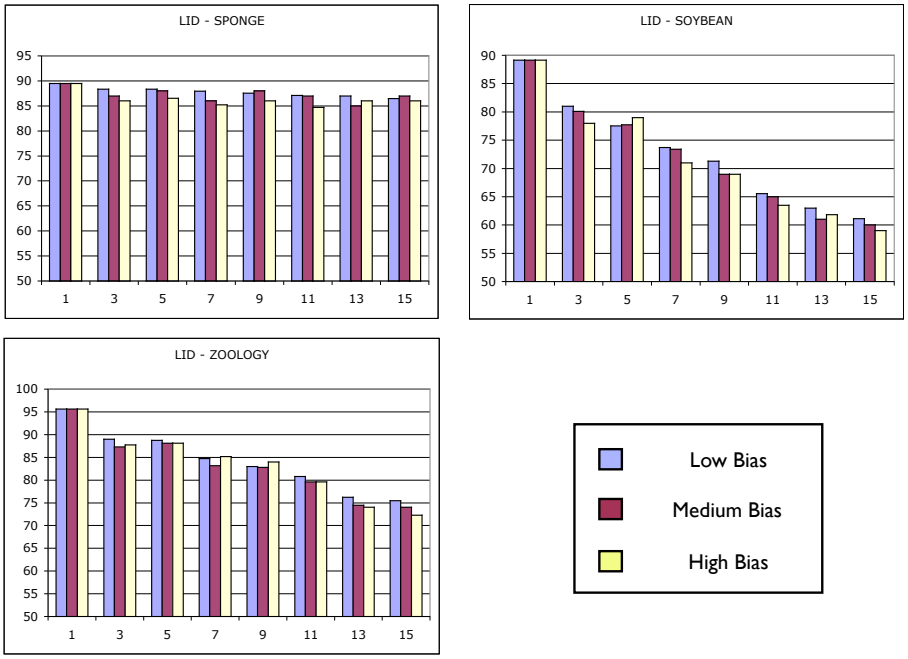


Figure 4.13: Bias effect on the CCS classification accuracy in agents using LID.

	Sponge	Soybean	Zoo
Low Committee Bias	0.00	0.00	0.0
Medium Committee Bias	0.32	0.12	0.22
High Committee Bias	0.45	0.17	0.32

Table 4.2: Committee Bias values used in the experiments.

chosen are shown in table 4.2. To choose those Committee Bias values, we have taken into account that the more solution classes that a data set has the more parts the solution class partition (with which Committee Bias is computed) has. Moreover, the larger the number of parts, the more difficult to obtain Committees with high Committee Bias. Therefore, in the soybean and zoo data sets, that have a higher number of solution classes than the sponge data sets, the Committee Bias values chosen are lower than those chosen for the spongers data set.

Figure 4.13 shows the classification accuracy for agents that use LID as learning method using CCS. This figure presents 3 different bars for each $\mathcal{M}$AC system: The left most bar shows the classification accuracy when Committee Bias $\mathbb{B}$ is low, the middle bar when $\mathbb{B}$ is medium, and the right most bar when $\mathbb{B}$ is high. Figure 4.13 shows a clear tendency of the CCS accuracy to decrease as Committee Bias increases. Although in some $\mathcal{M}$AC systems (like in the 5

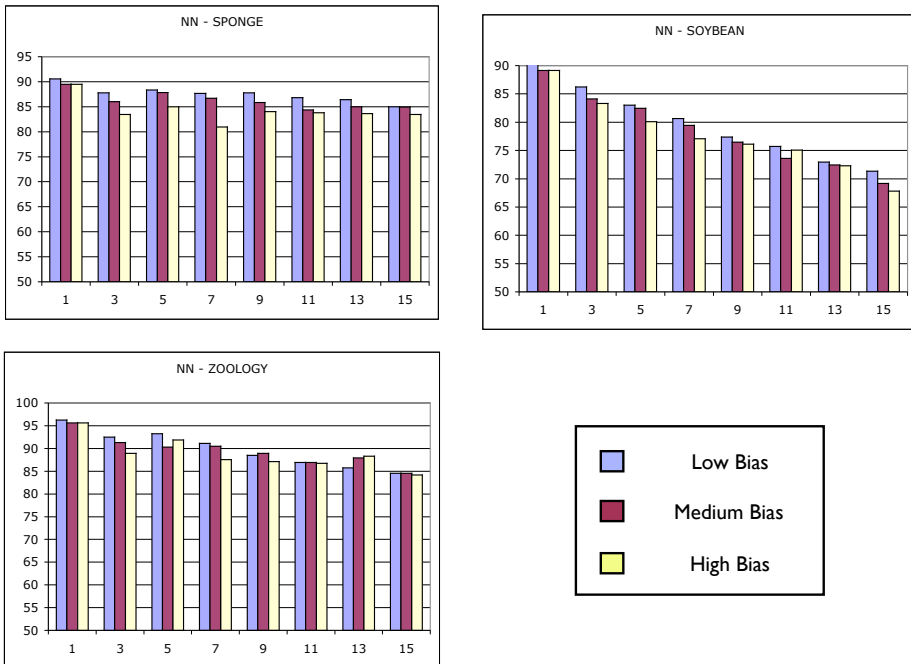


Figure 4.14: Bias effect on the CCS classification accuracy in agents using 1-NN.

agents system in the soybean data set or the 9 agents system in the zoo data set) accuracy has increased with Committee Bias, the global tendency is to decrease. Moreover, Figure 4.13 shows that the decrease in accuracy produced by Committee Bias does not depend on the number of agents in the committee, since the accuracy drop is present in all the $\mathcal{M}$AC systems experimented with. Moreover, notice that all the experiments presented in this section have been made with Committee Completeness $\mathbb{C} = 1$; therefore, in the 1 agent system, the Committee Bias is always $\mathbb{B} = 0.0$ since the agent owns all the cases in the training set.

Figure 4.14 shows the CCS results for agents that use 1-NN as learning method. The figure shows that CCS classification accuracy also decreases as Committee Bias increases. 1-NN has poorer results than LID under biased conditions in the sponge data set as we can see for the results in the 3 and 7 agents $\mathcal{M}$AC systems.

Finally, Figure 4.15 shows the CCS results for agents that use 3-NN as learning method. The figure shows that CCS classification accuracy also decreases as Committee Bias increases, but that 3-NN is perhaps the method that is less affected by Committee Bias. Except for the 3 and 7 agent $\mathcal{M}$AC systems in the sponge data set, accuracy reductions due to the increase of Committee Bias are smaller than those suffered by 1-NN or LID.

We can conclude that classification accuracy generally decreases as Commit-

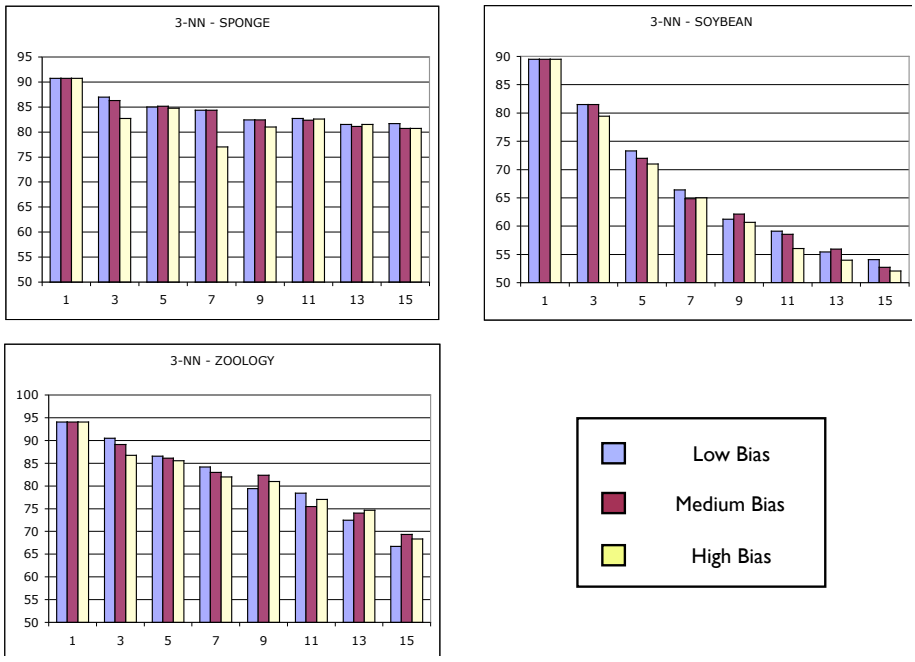


Figure 4.15: Bias effect on the CCS classification accuracy in agents using 3-NN.

tee Bias increases. Moreover, we have only presented the accuracy of the agents using the Committee Collaboration Strategy. In order to have a better intuition of the reasons of for which the accuracy diminishes, we will also present the average classification accuracy of the individual agents that take part in those committees.

Figure 4.16 shows the classification accuracy for individual agents that use LID as learning method. This figure presents three accuracy bars for each $\mathcal{MAC}$ system: for low Committee Bias $\mathbb{B}$ (left bar), for medium $\mathbb{B}$ (medium bar) and for high $\mathbb{B}$ (right bar). Notice that although individual accuracy is being measured we still use the term Committee Bias because we are measuring the individual accuracy in a $\mathcal{MAC}$ system where the Committee Bias would have certain value if all the agents join together in a committee. Figure 4.16 shows that individual accuracy decreases as Committee Bias increases in almost all the $\mathcal{MAC}$ systems presented and in all the data sets. Moreover, comparing the decrease in individual accuracy with the decrease of the accuracy of CCS in Figure 4.13 we can see that individual accuracy is much more affected by bias than the committee accuracy.

Figure 4.17 shows the classification accuracy for individual agents that use 1-NN as learning method. Committee Bias has the same effect in 1-NN than in LID, the individual accuracy diminishes. Moreover, as with LID, the decrease in individual accuracy due to Committee Bias is greater than the decrease in the

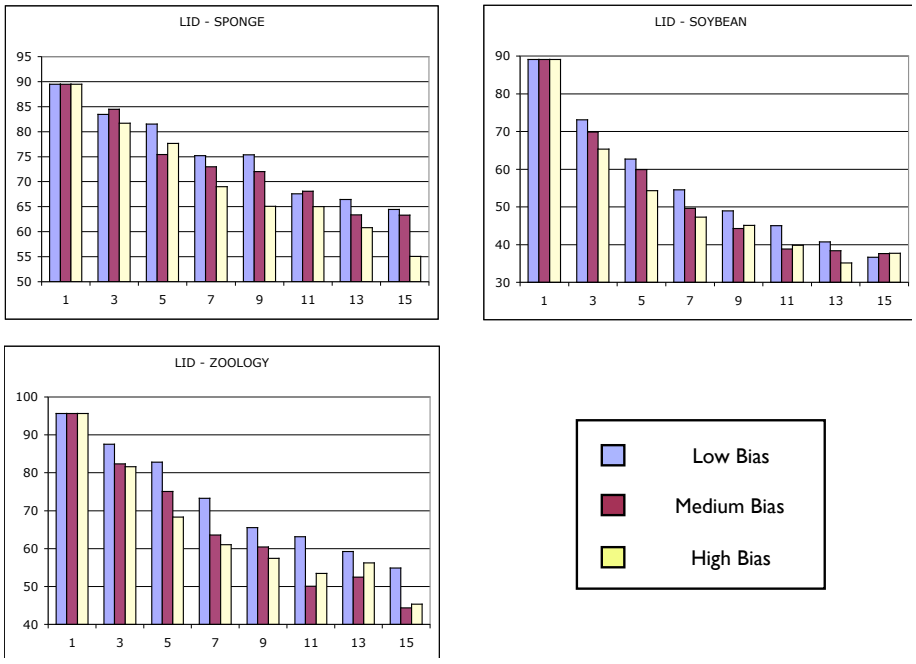


Figure 4.16: Bias effect on the Classification accuracy in agents using LID to solve problems individually.

accuracy of CCS (shown in Figure 4.14).

Figure 4.18 shows the classification accuracy for individual agents that use 3-NN as learning method with the same results as LID or 1-NN.

The experimental results presented in this section show that as the Committee Bias increases, the individual accuracy of the agents in a committee decreases. Moreover, the classification accuracy of the committee also decreases (albeit at a slower pace than the individual accuracy). This effect can be again explained in terms of the decomposition of the error of a committee: $E = \bar{E} - \bar{A}$ (where $\bar{E}$ is the mean error of the individual classifiers and $\bar{A}$ is the *ambiguity*, that increases as correlation decreases). The classification error of the committee does not increase as fast as the individual error because by increasing Committee Bias, the Case Base Bias of the individual agents increases. If Committee Completeness is $\mathbb{C} = 1.0$ and Committee Redundancy is $\mathbb{R} = 0.0$ (as in our experiments) and the Case Base Bias of all the individual agents in a committee is high, then clearly the type of Case Base Bias that each individual agent has is not the same, i.e. if an agent A_i has a high bias because has a lot of cases of a certain class S_k, the rest of agents in the system will have less cases of class S_k than A_i in average. Therefore, a high value of Case Base Bias with Committee Completeness $\mathbb{C} = 1.0$ and Committee Redundancy $\mathbb{R} = 0.0$ implies that the case bases of the individual agents a have different individual

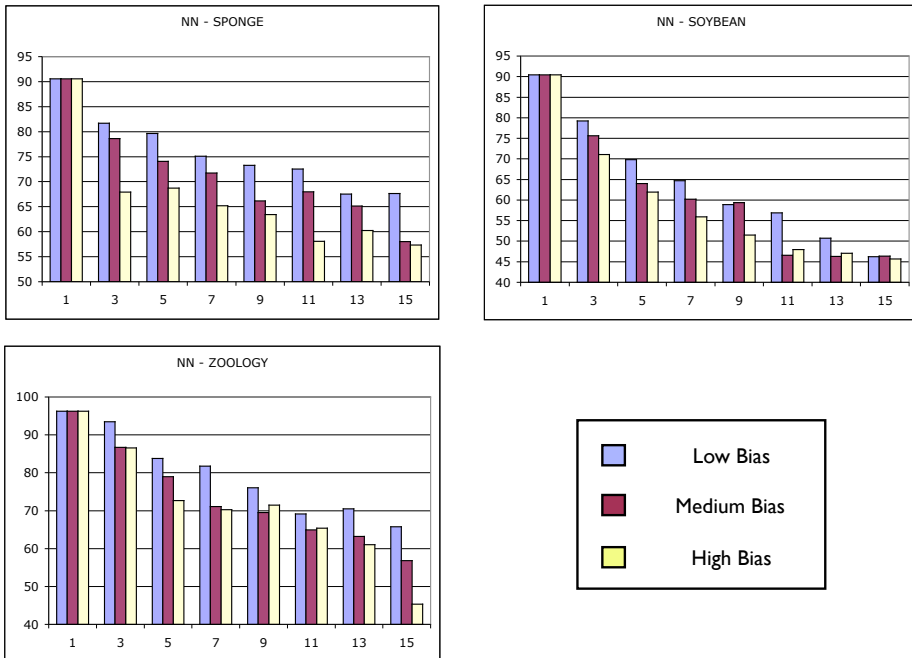


Figure 4.17: Bias effect on the Classification accuracy in agents using 1-NN to solve problems individually.

biases, leading to a decrease in the error correlation of the individual agents, i.e. an increase in the ambiguity $\bar{A}$. This decrease in the error correlation compensates a little the increase in classification error of the individual agents $\bar{E}$, but not enough to maintain the committee error E at the same level of accuracy achieved by $\mathcal{MAC}$ systems with no Committee Bias.

Summarizing, the effect of Committee Bias both in the Committee Collaboration Strategy and in the individual agents is to decrease classification accuracy In order to address this problem Chapter 9 introduces a technique (Case Bartering) capable of improving the performance of committees in the presence of bias.

4.5.4 Voting System Comparison

The Committee Collaboration Strategy strongly depends on the aggregation mechanism used to obtain an aggregated prediction from the predictions built by the individual agents. Therefore, in this section, we are going to test the performance of the committee using different voting systems. Specifically, we have made experiments using 4 different voting systems:

- BWAV,

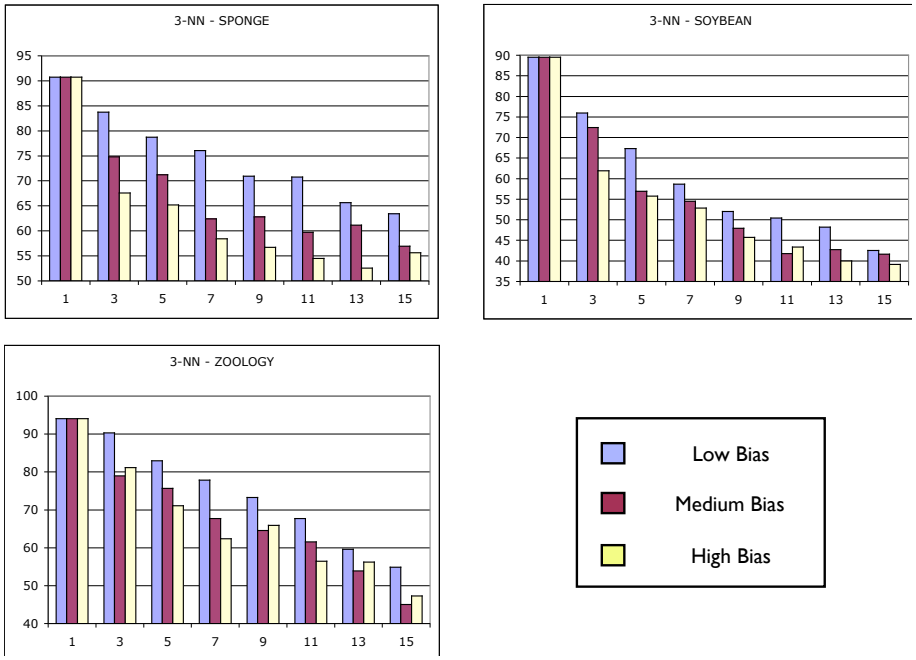


Figure 4.18: Bias effect on the Classification accuracy in agents using 3-NN to solve problems individually.

- Approval Voting (AV),
- Majority Voting (MV), where each individual agent will just vote for a single solution (the solution with more endorsing cases),
- Cases as Votes (CaV), where the cases retrieved by an individual agent will be considered as votes (i.e. each retrieved case equals to one vote, e.g. if an agent retrieves 3 cases supporting a certain solution class, then it will case 3 votes for that solution class).

Let us now define MV and CaV (since BWAV and AV have already been defined in Section 4.3). In MV, an agent only votes for that class for which more endorsing cases have been retrieved. Specifically, the vote of an agent A_i for a class S_k is:

$$MajorityVote(S_k, P, A_i) = \begin{cases} 1 & \text{If } \exists \mathbf{R} \in \mathcal{R}_{A_i} | \mathbf{R}.S = S_k \wedge \\ & \nexists \mathbf{R}' \in \mathcal{R}_{A_i} | \mathbf{R}'.E > \mathbf{R}.E, \\ 0 & \text{otherwise.} \end{cases} \quad (4.4)$$

When an agent $A_i$uses the CaV voting system, each retrieved case endorsing a class S_k will be considered a vote. Specifically:

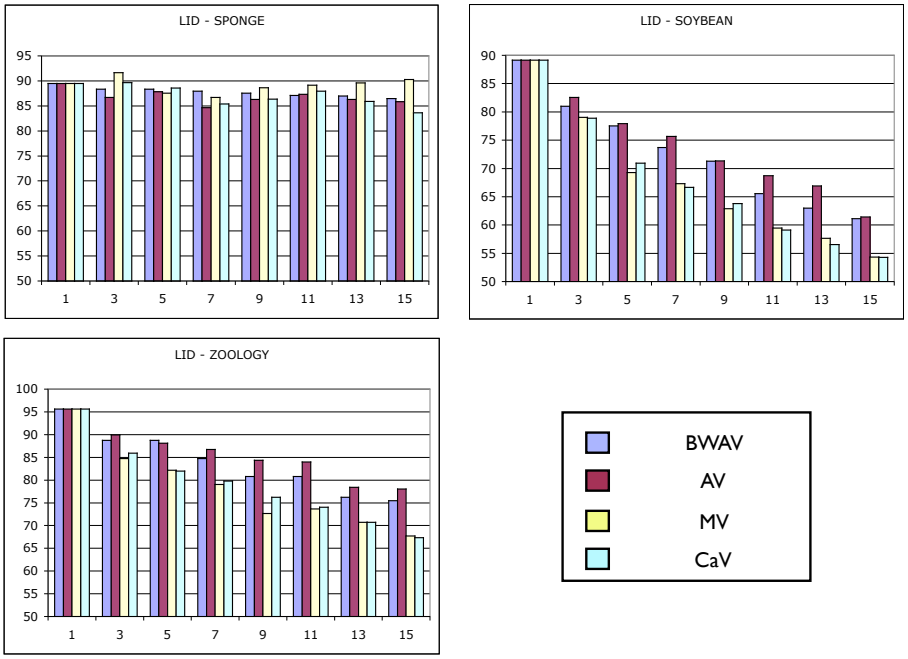


Figure 4.19: Classification accuracy comparison between agents using LID to solve problems individually and with the committee collaboration strategy.

$$CasesVote(S_k, P, A_i) = \begin{cases} \mathbf{R}.E & \text{If } \exists \mathbf{R} \in \mathcal{R}_{A_i} | \mathbf{R}.S = S_k, \\ 0 & \text{otherwise.} \end{cases} \quad (4.5)$$

In order to test the performance of each voting system we have made experiments with the three data sets (sponge, soybean and zoology). However, only results using LID and 3-NN learning methods are presented here since all the voting systems are identical when using 1-NN.

Figure 4.19 shows the results for agents that use LID as learning method. The vertical axis shows the average classification accuracy, and the horizontal axis shows the number of agents in the committee. For each committee four bars are shown, representing classification accuracy using (from left to right) BWAV, AV, MV and CaV voting systems. Figure 4.19 shows that BWAV and AV achieve clearly better results than MV and CaV both in the soybean and in the zoology data sets. Only in the sponge data set MV achieves higher accuracy values. Comparing BWAV with AV, BWAV achieves higher accuracies in the sponge data set, while AV wins in the other two data sets.

Figure 4.20 shows the same experiments than Figure 4.19 but with agents using 3-NN as learning method. BWAV and CaV obtain higher accuracies than AV or MV in most scenarios. MV performs clearly worst than the rest in all the data sets except in the sponge data set. CaV works well with a 3-Nearest

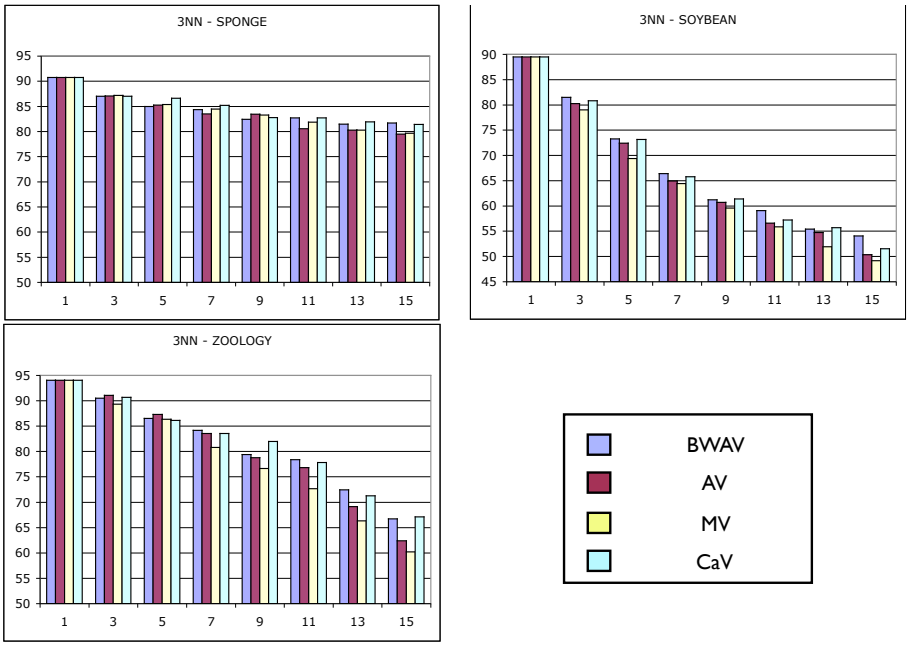


Figure 4.20: Classification accuracy comparison between agents using 3-Nearest Neighbor to solve problems individually and with the committee collaboration strategy.

Neighbor method, since as the number of retrieved cases is always three, the maximum number of votes that a solution class can have is bound to three, and therefore CaV behaves similarly to BWAV.

The interpretation of these results is the following: MV loses a lot of information that the agents can provide, since each agent only votes for a single solution class no weight is assigned to the votes. CaV uses more information than MV (since it uses the number of retrieved cases for each solution class), but is very sensible to the number of cases retrieved. It works well with methods where the number of retrieved cases is bounded (such as k-Nearest Neighbor), but has problems with methods that do not have any limit in the number of retrieved cases (such as LID). AV and BWAV use also more information than MV and are not as sensible to the number of cases retrieved as CaV is. We have shown that both BWAV and approval voting (AV) are the stronger voting systems. However, BWAV is a more informed voting system than AV since agents can weight the votes they case for each solution. As we will show in the next chapter, this information can be useful in order to learn *competence models*, so in the rest of this monograph BWAV will be used as the voting system of choice.

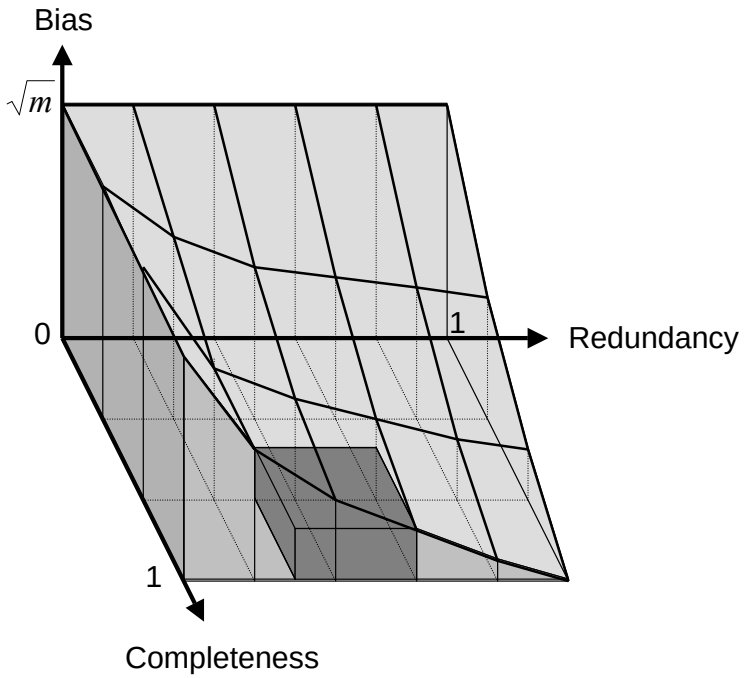


Figure 4.21: *Ensemble Space*: the space of possible committees characterized by Committee Completeness, Committee Bias and Committee Redundancy. The dark area contains the committees with higher accuracy.

4.6 Ensemble Space Redux

In this chapter we have presented a characterization of committees based on three measures, namely: Committee Completeness, Committee Redundancy and Committee Bias. Those three measures represent different aspects of committees. In the experiments section, we have performed several experiments in order to determine how each one of the measures in the committee characterization affects classification accuracy of a committee and therefore determine how individual case base characteristics affect the performance of a committee. In Section 4.5.2 we have presented experiments showing that classification accuracy increases if Committee Redundancy is moderate, even outperforming centralized single agent scenarios that work with the same data. Moreover, in Section 4.5.3 we have presented experiments showing that, as Committee Bias increases, classification accuracy of the committee decreases. Finally, although we have not performed any experiment to test if the accuracy increases or decreases with different values of Committee Completeness, it is obvious that committee (and

individual) accuracy will increase as Committee Completeness decreases.

Therefore, the area of the *Ensemble Space* marked in Figure 4.21 in dark grey contains committees with high accuracy (the area with low Committee Bias, high Committee Completeness and moderate Committee Redundancy). In the experiments section we have seen that certain committees in this area can achieve classification accuracy values higher than a centralized approach where a single agent owns all the cases. Finally, we would like to remark that these results have a direct relation with ensemble learning such as Bagging. Notice that Bagging creates individual classifiers by selecting random subsets of the training set with replication, i.e. adding some redundancy. Therefore, the resulting ensemble of a Bagging process will be the equivalent of a committee falling in the dark grey area shown in Figure 4.21.

4.7 Conclusions

This chapter has presented the Committee Collaboration Strategy (CCS), that allows groups of individual CBR agents to collaborate by forming committees in order to solve problems without compromising their autonomy. Committees allow a group of agents to solve problems taking benefit from the ensemble effect and therefore solving them with a higher classification accuracy than if the problems were solved individually. We have experimentally compared the classification accuracy of CCS with respect the classification accuracy of the individual agents, and obtained that CCS achieves always higher classification accuracy than the individual agents, specially when there is a large number of agents with small case bases.

Moreover, CCS requires a voting system in order to aggregate the predictions coming from the individual agents of the committee. In this chapter we have presented the BWAV voting system and compared it with other voting systems such as Approval Voting (AV), Majority Voting (MV) or Cases as Votes (CaV). The comparison between the classification accuracy of CCS with different voting systems yielded that BWAV and AV are the stronger voting systems. Moreover, BWAV is a more informed voting system than AV since agents can weight the votes they cast for each solution. As we will show in the next chapter, this information can be useful in order to improve the collaboration of agents and therefore BWAV will be the voting system of choice in the rest of this monograph.

The classification accuracy of CCS strongly depends on the case bases of the individual agents that compose a committee. In order to analyze this dependency, we have presented a characterization of committees based on three measures, namely: Committee Completeness, Committee Redundancy and Committee Bias. These three measures characterize the distribution of cases among the agents in a committee. Moreover, we have performed experiments to see how each measure affects classification accuracy of CCS. The conclusions extracted from those experiments is that committees that have high Committee Completeness, low Committee Bias and moderate Committee Redundancy achieve

the highest classification accuracy. It is interesting to note that a committee satisfying the previous conditions can outperform a centralized approach where single agent owns all the cases in terms of classification accuracy.

Chapter 5

Dynamic Committees

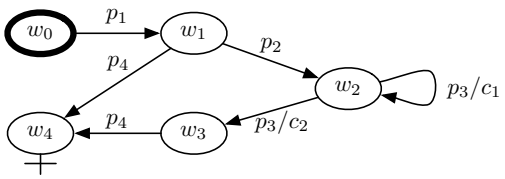
In the previous chapter we have shown that convening committees can effectively help individual agents to increase their classification accuracy. In this chapter we are going to present several collaboration strategies aimed at deciding when to convene a committee and which agents invite to join a committee. Specifically, we are going to present two collaboration strategies, namely *Peer Counsel Collaboration Strategy* and *Bounded Counsel Collaboration Strategy*, and empirically evaluate them.

5.1 Introduction

We have shown that the Committee Collaboration Strategy (CCS) can effectively improve the problem solving performance of the agents in a $\mathcal{MAC}$ system with respect to agents solving problems individually. However, when an agent uses CCS, no policy is used to select which agents are invited to join the committee, and *all* the agents in a $\mathcal{MAC}$ system are invited each time that an agent wants to use CCS. Moreover, it is not obvious that forming a committee with all the available agents is the best option for all the problems: possibly smaller committees have an accuracy comparable (or indistinguishable) to that of the complete committee. Furthermore, possibly some problems could be confidently solved by one agent while others could need a large committee to be solved with confidence.

In this chapter we will study different collaboration strategies that do not invite always all the agents to join the committee. The goal of these strategies is to study whether it is possible to achieve similar accuracies than the Committee Collaboration Strategy without convening always the complete committee. Moreover, some application domains may require smaller committees than others. We are interested in studying whether it is possible to provide agents with strategies that convene large committees only when the application domain requires it, and convene smaller ones when there is no need for large ones.

This chapter will focus on solving two main problems:



p_1 :	$Request(?User, ?A_i, ?P)$
p_2 :	$Request(!A_i, \mathcal{A}^c, !P)$
p_3/c_1 :	$Inform(?A_j, !A_i, ?\mathbf{R})/$ $\lvert!w_0w_1\mathbf{R}\rvert < \#(\mathcal{A}^c) - 2$
p_3/c_2 :	$Inform(?A_j, !A_i, ?\mathbf{R})/$ $\lvert!w_0w_1\mathbf{R}\rvert = \#(\mathcal{A}^c) - 2$
p_4 :	$Inform(!A_i, !User, ?S)$

Figure 5.1: Interaction protocol for the *Peer Counsel* collaboration strategy.

1. Deciding when an individual agent can solve a problem individually and when it is needed to convene a committee,

2. Deciding, when a committee is being convened, how many agents and which agents should be invited to join the committee.

A collaboration strategy that convenes a different committee in function of the current problem is called a *Dynamic Committee* collaboration strategy.

In this chapter we will present three dynamic committee strategies, namely the *Peer Counsel Collaboration Strategy* (P-CCS) and the *Bounded Counsel Collaboration Strategy* (B-CCS) that are intermediate points between CCS and solving problems individually.

All the strategies presented in this chapter use *competence models* (see Section 3.5.1) in order to decide the number of agents in a committee. Both P-CCS and B-CCS use predefined competence models.

The structure of the chapter is as follows. First, we will present the Peer Counsel Collaboration Strategy. Them, the Bounded Counsel Collaboration Strategy will be presented. Finally, we will experimentally evaluate both collaboration strategies.

5.2 Peer Counsel Collaboration Strategy

The intuition behind the Peer Counsel Collaboration Strategy (P-CCS) is that a committee only needs to be convened for difficult problems. Easy problems can be solved individually. To accomplish that, an agent A_i using P-CCS will

use the following strategy: each time that A_i has to solve a problem P, the problem will be first solved individually; A_i will then use a *competence model* of itself that assesses the confidence of the individually predicted solution: if this confidence is high, the individual solution is kept; otherwise a committee will be convened. In the case that a committee has to be convened, A_i will proceed exactly as in CCS.

The Peer Counsel Collaboration Strategy is composed of an interaction protocol, and a decision policy:

Definition 5.2.1. *The* Peer Committee Collaboration Strategy*(P-CCS) is a collaboration strategy* $\langle I_{P-CCS}, D_{PC}, D_V \rangle$*, where* I_{P-CCS} *is the P-CCS interaction protocol shown in Figure 5.1,* D_{PC} *is the* Peer Counsel *decision policy used to decide when to convene a committee, and* D_V *is a decision policy based on any voting system that can be used to aggregate the evidence gathered by the individual agents into a global prediction.*

Figure 5.1 shows the formal specification of the I_{P-CCS} interaction protocol. Notice that the protocol is similar to the CCS protocol (Figure 4.2), except that now there is a connection between states w_1 and w_2: the protocol will move from state w_1 to w_2 when the agent A_i decides (by using the D_{PC} decision policy) that the problem can be solved individually.

The D_{PC} decision policy is used in state w_1 of the P-CCS interaction protocol. Notice that in state w_1, the convener agent has two possible messages to send: p_2 to convene a committee or p_4 to solve the problem individually. The D_{PC} decision policy uses a *Self-Competence* model in order to decide whether the agent itself is competent enough to solve a problem P without external help. A *Self-Competence* model for an agent A_i takes as input $\mathcal{R}_{A_i}$ (the set of SERs built by an agent A_i to solve a problem P) and returns a confidence value in the interval $[0, 1]$, that represents the confidence on A_i for having individually found the correct solution for P (0 represents the minimum confidence and 1 represents the maximum confidence):

Definition 5.2.2. *(Self-Competence Model)*

$$\text{Self-Competence}(\mathcal{R}_{A_i}) = \begin{cases} \frac{1}{M} Ballot(Sol(\mathcal{S}, \{A_i\}), \{A_i\}) & \text{If } N > 1, \\ Ballot(Sol(\mathcal{S}, \{A_i\}), \{A_i\}) & \text{If } N = 1. \end{cases}$$

where $M = \sum_{S_k \in \mathcal{S}} Ballot(S_k, \mathcal{A}^c)$*, is the sum of all the votes casted by the agent. and* $N = \#(\{S_k \in \mathcal{S} | Ballot(S_k, \mathcal{A}^c) \neq 0\})$*, is the number of different classes for which the agent has voted for.*

That is to say, if agent A_i has built only one SER for a single solution ($N = 1$), the *Self-Competence* model will return the ballot for that solution (i.e. the votes for that solution). The intuition is that the higher the ballot, the larger the number of cases retrieved endorsing the predicted solution class, and therefore the higher the confidence on having predicted the correct solution class. Moreover, if agent A_i has built SERs for more than one solution (and

therefore $N > 1$), the *Self-Competence* model will return the fraction of votes that that are given to the most voted solution $Sol(\mathcal{S}, \{A_i\})$. The intuition here is that the larger the fraction of votes for the predicted solution class, the larger the fraction of retrieved cases endorsing that solution class, and therefore the higher the confidence on having predicted the correct solution class.

Using this competence model, we can now define the D_{PC} decision policy for an agent A_i as a boolean decision policy that decides whether to solve a problem individually or not. When $D_{PC}(\mathcal{R}_{A_i}) = true$ the problem will be solved individually, otherwise a committee will be convened:

Definition 5.2.3. *(Peer Counsel Decision Policy)*

$$D_{PC}(\mathcal{R}_{A_i}) = (\text{Self-Competence}(\mathcal{R}_{A_i}) \geq \eta\ )$$

where η is a threshold parameter.

The intuition behind the D_{PC} decision policy is that when the confidence on solving a specific problem individually is high enough, A_i does not need to convene a committee since the individual prediction is very likely to be correct and convening a committee is useless. However, if the individual confidence is low, it is better to convene a committee in order to confirm that the predicted solution class was correct or to predict another solution class that is more likely to be correct. Moreover, in our experiments, we have set the threshold $\eta = 0.75$.

Section 5.4 presents the experimental results evaluating P-CCS. The next section presents the *Bounded Counsel Collaboration Strategy* that, in addition of trying to convene the complete committee only when needed (as P-CCS), also tries to reduce the number of agents in the committee.

5.3 Bounded Counsel Collaboration Strategy

While the idea of P-CCS is to convene a committee only for difficult problems, the *Bounded Counsel Collaboration Strategy* (B-CCS) is designed to study if smaller committees can reach accuracy values comparable to that of the the complete committee. We want to study when smaller committees can be convened without losing classification accuracy, and how can an agent decide how many agents need to be invited to join the committee.

We propose an incremental approach to determine the size of the committee needed to solve a problem. In the incremental approach, the convener agent first solves the problem individually. Then, a competence model is used to determine whether there is enough confidence on the individually predicted solution. If there is not enough confidence, then a committee is convened in an incremental way: a new agent A_j is invited to join the committee; the committee of two agents solve the problem and a competence model is used again to determine whether there is enough confidence on the solution predicted by that committee. If there is not enough confidence a new agent is invited, and so on. When the competence model estimates that a solution has enough

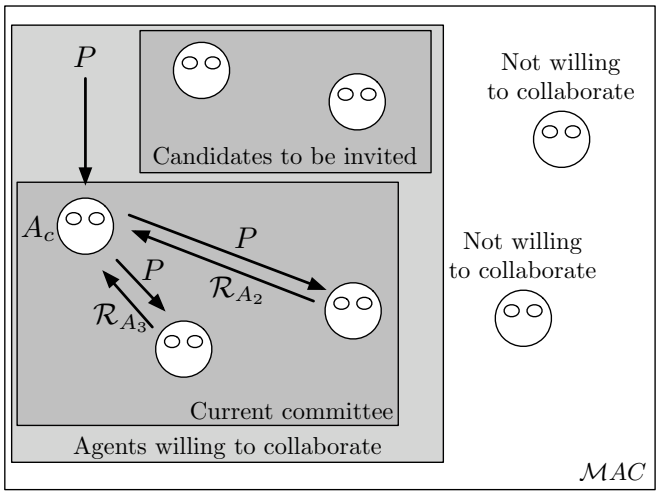


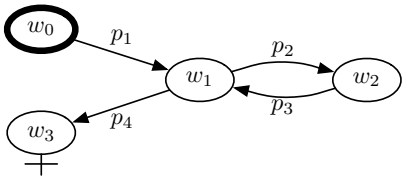
Figure 5.2: Illustration of B-CCS where 3 agents have already been invited to join the committee, forming a committee of 4 agents.

confidence, the process terminates and the solution predicted is returned. Figure 5.2 illustrates this process: from all the agents in the $\mathcal{MAC}$ system that have agreed to collaborate, some of them have already joined the committee, and some of them are candidates to be invited if the confidence in the solution predicted by the current committee is not high enough. Moreover, notice that some agents in the $\mathcal{MAC}$ system could be unwilling to participate in B-CCS, thus are not candidates to be invited to join the committee.

The Bounded Counsel collaboration strategy is composed by an interaction protocol and two decision policies:

Definition 5.3.1. *The* Bounded Committee Collaboration Strategy *(B-CCS) is a collaboration strategy* $\langle I_{B-CCS}, D_H, D_{AS}, D_V \rangle$*, where* I_{B-CCS} *is the B-CCS interaction protocol shown in Figure 5.3,* D_H *is the Bounded Counsel* Halting *decision policy (used to decide when to stop inviting agents to join the committee),* D_{AS} *is the Bounded Counsel* Agent Selection *decision policy (used to decide which will be the next agent to be invited to join the committee) and* D_V *is a decision policy based on any voting system that can be used to aggregate the evidence gathered by the individual agents into a global prediction.*

Figure 5.3 shows the formal specification of the I_{B-CCS} interaction protocol. The protocol consists of 4 states: w_0 is the initial state, and, when a user requests an agent A_i to solve a problem P, the protocol moves to state w_1. The first time the protocol is in state w_1 the convener agent decides whether convening a committee is needed or not. If A_i wants to invite another agent, then message p_2 is sent to another agent A_j containing the problem P and the protocol moves to state w_2. In state w_2 A_i waits until A_j sends back message p_3 containing its own



p_1 :	$Request(?User, ?A_i, ?P)$
p_2 :	$Request(!A_i, ?A_j, !P)$
p_3 :	$Inform(!A_j, !A_i, ?\mathbf{R})$
p_4 :	$Inform(!A_i, !User, ?S)$

Figure 5.3: Interaction protocol for the *Bounded Counsel* collaboration strategy.

prediction for the problem P, then the protocol will move back to state w_1. In state w_1 the convener agent assesses the confidence of the current solution and uses the D_H decision policy to decide whether another agent has to be invited to join the committee or not. If A_i decides to invite more agents, then message p_2 will be send to another agent, repeating the process of inviting another agent; if A_i decides that no more agents need to be invited to join the committee the voting system will be used to obtain an aggregate solution S. Finally, A_i will send the result to the user with message p_4, and the protocol will move to the final state w_3.

B-CCS requires two individual decision policies: the Bounded Counsel *Halting* decision policy D_H, that decides whether inviting more agents to join the committee is needed, and the Bounded Counsel *Agent Selection* decision policy D_{AS}, that decides which agent A_j to invite to join the committee. Notice that the order in which the agents are invited to join the committee may be important if the goal of A_i is to invite to the minimum number of agents. However, in our experiments, the D_{AS} policy randomly selects an agent from the set of agents in $\mathcal{A}$ that have not been still convened into the committee. In Chapter 6 we will present a more informed way of selecting the next agent to invite.

The D_H decision policy uses the *Committee-Competence* model that measures the confidence in a solution predicted by a committee to be correct. The *Committee-Competence* model takes as input $\mathcal{R}^c$ (the set of SERs built by a committee of agents $\mathcal{A}^c$ to solve a problem P) and returns a confidence value in the interval $[0, 1]$, that represents the confidence on the committee for having found the correct solution for P (0 represents the minimum confidence and 1 represents the maximum confidence):

Definition 5.3.2. *(Committee-Competence Model)*

$$\text{Committee-Competence}(\mathcal{R}^c) = \begin{cases} \frac{1}{M} Ballot(Sol(\mathcal{S}, \mathcal{A}^c), \mathcal{A}^c) & \text{If } N > 1, \\ min(Ballot(Sol(\mathcal{S}, \mathcal{A}^c), \mathcal{A}^c), 1) & \text{If } N = 1. \end{cases}$$

where $M = \sum_{S_k \in \mathcal{S}} Ballot(S_k, \mathcal{A}^c)$*, is the sum of all the votes casted by the agents and* $N = \#(\{S_k \in \mathcal{S} | Ballot(S_k, \mathcal{A}^c) \neq 0\})$*, is the number of different classes for which the agents have voted for.*

That is to say, if the agents in $\mathcal{A}^c$ have built SERs for a single solution ($N = 1$), the *Committee-Competence* model will return the ballot for that solution. Moreover, notice that the ballot for a solution when there are more than one agent in $\mathcal{A}^c$ can be greater than 1. Therefore we take the minimum between the ballot and 1 to ensure that the competence models output confidence values within the interval $[0, 1]$. The intuition is that the higher the ballot, the larger the number of cases retrieved by the agents endorsing the predicted solution class, and therefore the higher the confidence on having predicted the correct solution class. Moreover, if the agents in $\mathcal{A}^c$ have built SERs for more than one solution (and therefore $N > 1$), the *Committee-Competence* model will return the fraction of votes that that are given to the most voted solution $Sol(\mathcal{S}, \{A_i\})$. The larger fraction of votes for the predicted solution, the larger the number of agents that have voted for the predicted solution class or the larger the number of cases that each individual agent has retrieved endorsing the predicted solution class, and therefore the higher the confidence on having predicted the correct solution class.

Using this competence model, we can now define the D_H as a boolean decision policy that decides whether the convener agent can stop inviting agents to the committee; if $D_H(\mathcal{R}^c) = true$, no more agents will be invited to the committee.

Definition 5.3.3. *(Bounded Counsel Halting Decision Policy)*

$$D_H(\mathcal{R}^c) = (\text{Committee-Competence}(\mathcal{R}^c) \geq \eta)$$

where η *is a threshold parameter.*

The intuition behind the D_H decision policy is that if the confidence on the solution predicted by the current committee is high enough, there is no need for inviting more agents to join the committee. Notice that when A_i is alone (and can be considered as a committee of 1) this decision is equivalent to choose between solving the problem individually or convene a committee. In our experiments we have set $\eta = 0.75$.

The next section presents an experimental evaluation of P-CCS and B-CCS collaboration strategies, and compares them with the Committee Collaboration Strategy.

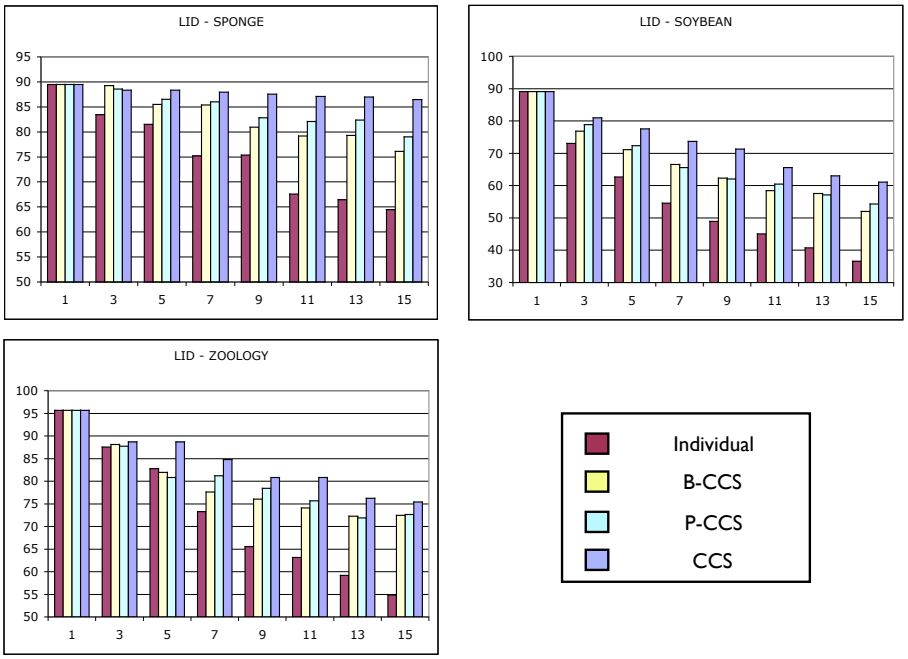


Figure 5.4: Classification accuracy comparison between agents using LID to solve problems with several collaboration strategies.

5.4 Experimental Evaluation

In this section we are going to empirically compare the performance of the Peer Counsel and Bounded Counsel collaboration strategies with the Committee Collaboration Strategy (CCS) and with agents working in isolation. In order to evaluate the performance of a collaboration strategy we will measure two features: the classification accuracy and the size of the committees convened.

We used the same scenarios than in previous chapter, i.e. $\mathcal{M}$AC systems consisting of 3, 5, 7, 9, 11, 13 and 15 agents using 1-NN, 3-NN, and LID as learning methods. The data sets used are also sponges, soybean and zoology. The results presented are also the average of 5 runs of a 10 fold cross validation.

5.4.1 Accuracy Evaluation

Figures 5.4, 5.5 and 5.6 present the classification accuracy achieved by agents in several $\mathcal{M}$AC systems using the dynamic collaboration strategies presented so far.

Figure 5.4 shows the results for agents that use LID as learning method. The Committee Collaboration Strategy (CCS) has achieved the higher accuracy values in almost all the experiments and agents solving problems individually achieve the lowest accuracy in all the experiments. The Bounded Counsel

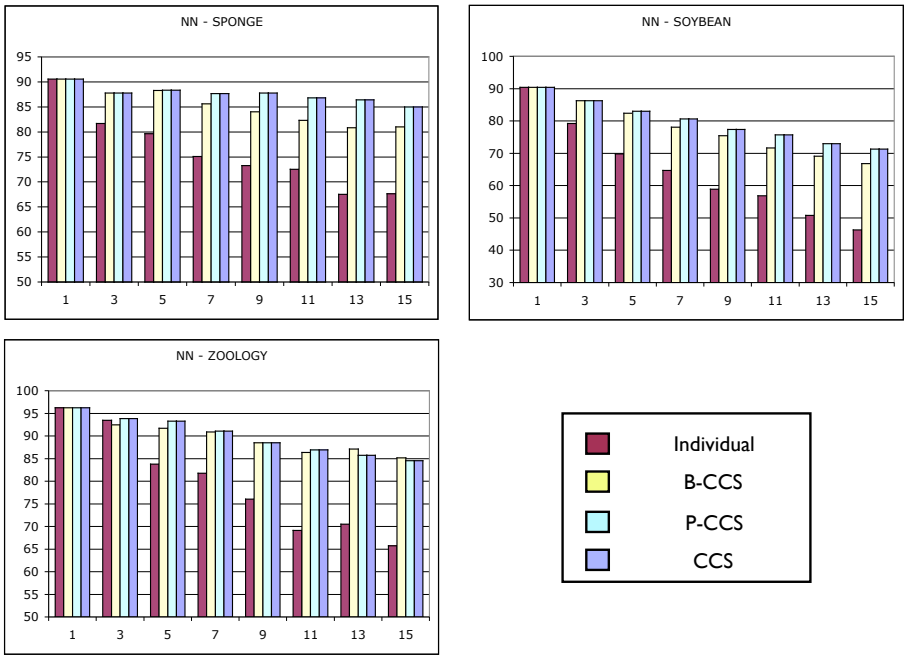


Figure 5.5: Classification accuracy comparison between agents using Nearest Neighbor to solve problems with several collaboration strategies.

Collaboration Strategy and the Peer Counsel Collaboration Strategy have an intermediate accuracy value between CCS and individual agents: they achieve noticeably higher accuracy than agents working individually but without reaching the accuracy of CCS. For instance, in a $\mathcal{M}$AC system of 9 agents in the zoology data set, agents working individually achieve an accuracy of 65.54% while CCS achieves an accuracy of 80.79%; B-CCS achieves an accuracy of 76.04% and P-CCS an accuracy of 78.41%. Comparing Bounded Counsel with Peer Counsel, Peer Counsel usually achieve higher accuracy values than Bounded Counsel, but the difference is very small.

Figure 5.5 shows the results for agents that use 1-NN as learning method. Notice that P-CCS is equivalent to CCS when the learning method is 1-NN since the Peer Counsel decision policy D_{PC} always decides to convene the complete committee if the number of retrieved cases is only 1 (as happens to be in 1-NN). Therefore, we will only consider the comparison among individual agents, B-CCS and CCS. Figure 5.5 shows that the accuracy of B-CCS is again between individual agents and CCS in the sponge and soybean data sets. However, with 1-NN, B-CCS is able to achieve the same level of accuracy than CCS in the zoology data set. Moreover, the difference between B-CCS and CCS is smaller using nearest neighbor than using LID. For instance, in a 9 agents $\mathcal{M}$AC system in the soybean data set, individual agents achieve an accuracy of 58.89%, B-CCS

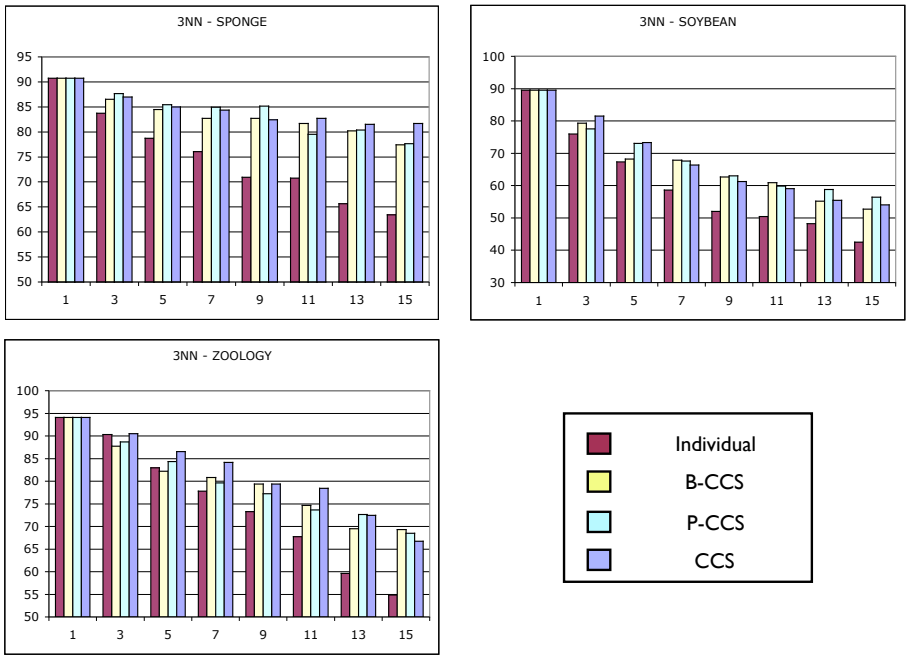


Figure 5.6: Classification accuracy comparison between agents using 3-Nearest Neighbor to solve problems with several collaboration strategies.

an accuracy of 75.44% while P-CCS and CCS has an accuracy of 77.39%.

Figure 5.6 shows the results for agents that use 3-NN as learning method. Notice that using 3-NN the accuracy of B-CCS and P-CCS is much closer to the accuracy of CCS than using LID and 1-NN. In fact, CCS achieves higher accuracy than B-CCS and P-CCS in 10 experiments, P-CCS outperforms CCS and B-CCS in 7 experiments, and B-CCS achieves the highest accuracy in 6 experiments. In the three experiments where the $\mathcal{M}$AC systems are composed of a single agent, all the strategies achieve the same accuracy since they are equivalent.

Summarizing, on average, the Committee Collaboration Strategy achieves the highest accuracy values, the Peer Counsel Collaboration Strategy and the Bounded Counsel Collaboration Strategy follow close, and agents working individually achieve very low classification accuracy when the data is very fragmented. The difference between P-CCS and B-CCS lies in the size of the convened committees. Let us now analyze the experimental results in terms of committee sizes.

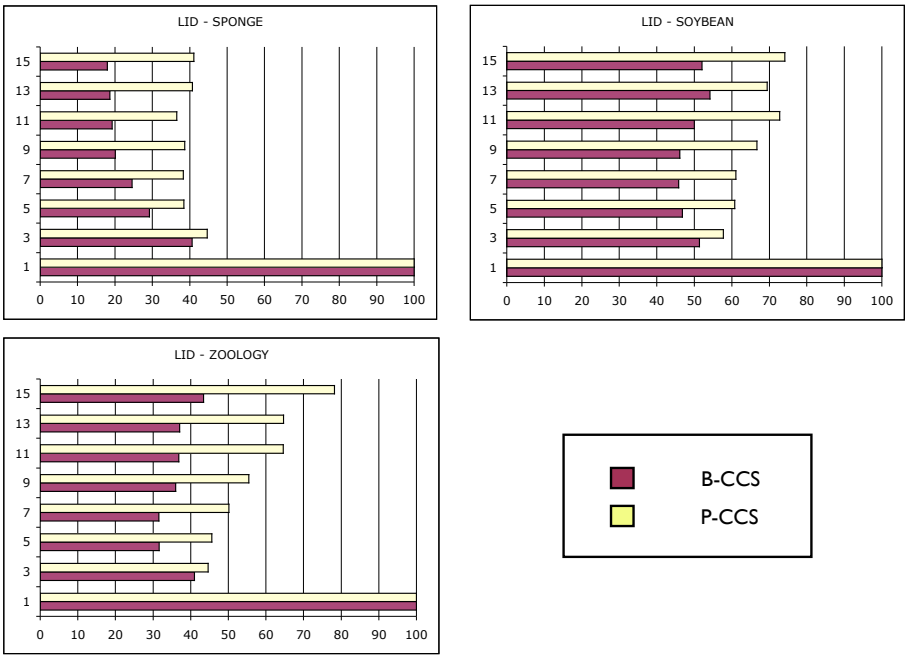


Figure 5.7: Committee size comparison between agents using LID to solve problems with several collaboration strategies.

1	3	5	7	9	11	13	15
100%	89.0%	77.0%	72.1%	69.0%	69.81%	64.25%	63.1%

Table 5.1: Percentage of problems solved individually by agents using P-CCS in the sponge data set using LID.

5.4.2 Committee Size Evaluation

Figures 5.7, 5.8, and 5.9 present the average number of agents in the committees convened for solving problems in several $\mathcal{M}$AC systems using the dynamic collaboration strategies presented so far. One of the reasons for studying the size of the committees convened by the different collaboration strategies is that in many application domains, convening committees may have a cost, and that cost is surely related to the number of agents that join the committee.

Figure 5.7 shows the average size of the committees convened by the different collaboration strategies when agents use LID as learning method. We have considered that when the agents work individually, they form a committee of only one agent. When the agents use the Committee Collaboration Strategy, all the agents in the system are invited to join the committee and, since in our experiments agents never reject to join a committee, therefore the size of the committees convened by CCS is exactly the number of agents in the $\mathcal{M}$AC

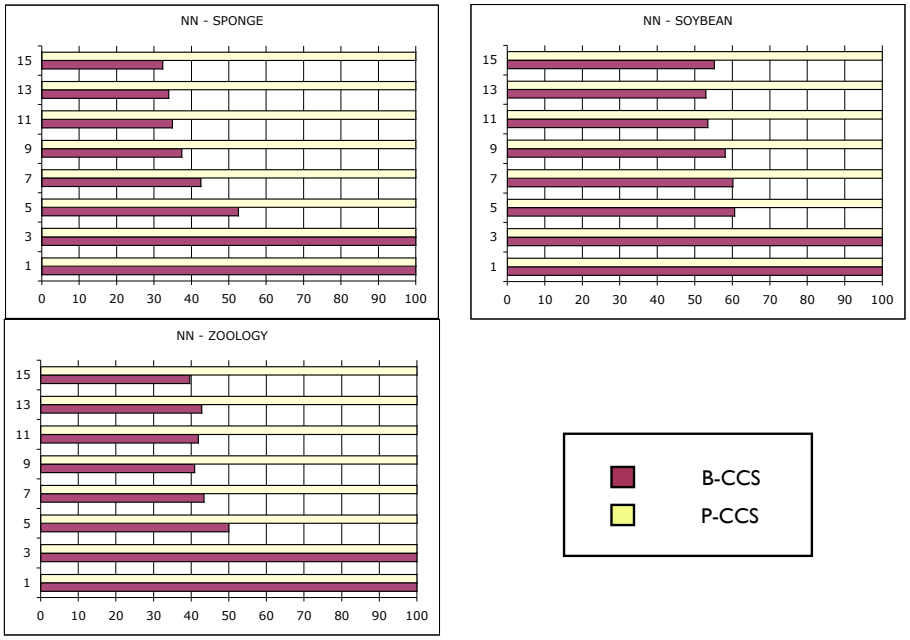


Figure 5.8: Committee size comparison between agents using Nearest Neighbor to solve problems with several collaboration strategies.

system. P-CCS and B-CCS lie between these two extremes. The average size of the committees convened by P-CCS and B-CCS grows as the number of agents increase, but it never reaches the size of the committees convened by CCS. Moreover, Figure 5.7 shows that the average committee size achieved by B-CCS is always lower than that of P-CCS. The reason is that only a reduced number of agents join the committee convened by B-CCS for each problem. Moreover, notice that the committees convened by B-CCS and P-CCS in the sponge data set are smaller the committees convened in the soybean or zoology data sets. For instance, the average size of a committee convened by B-CCS in a $\mathcal{M}$AC system composed of 15 agents in the sponge data set is 15.26% of the agents while it is 52.06% in the soybean data set. Moreover, the average size of a committee convened by P-CCS in a $\mathcal{M}$AC system composed of 15 agents in the sponge data set is 41.13% of the agent while it is 74.13% in the soybean data set. The conclusion is that both P-CCS and B-CCS are adaptive, since depending on the data set the number of agents in a committee varies because the number of agents convened that are needed to achieve a high confidence in the solution is different for each data set.

Moreover, Table 5.1 shows the percentage of times that an agent using P-CCS solves a problem individually in the sponge data set and using LID. The figure shows that when the individual agent has a large case base (in our exper-

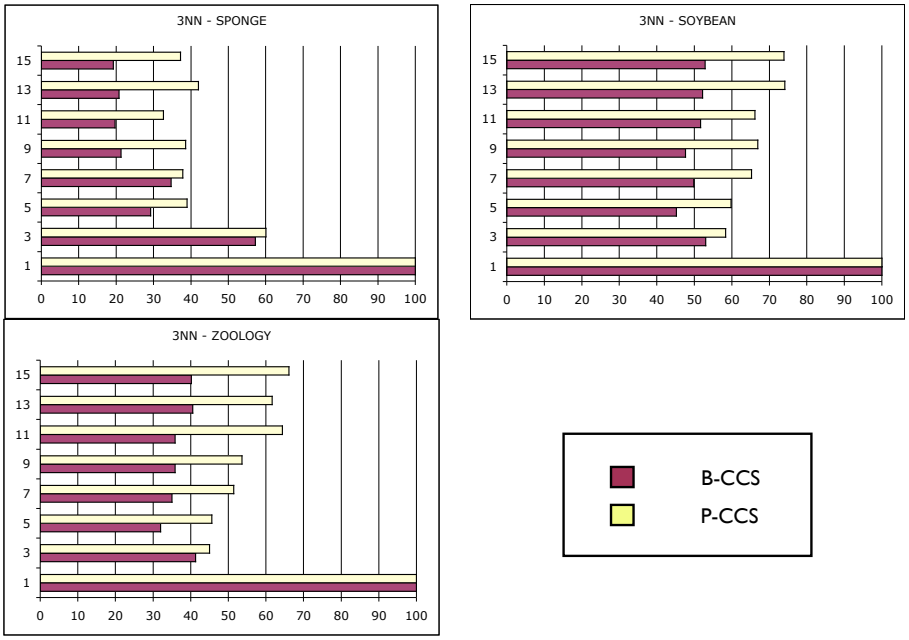


Figure 5.9: Committee size comparison between agents using 3-Nearest Neighbor to solve problems with several collaboration strategies.

iments, when the $\mathcal{M}$AC system contains few agents) the agents solve problems individually more often, e.g. a 89.0% of the problems in the 3 agents system. As the case base size of the agents decrease (in experiments with a large number of agents), the agent convenes committees more often since the confidence in the individually predicted solutions decreases.

Figure 5.8 shows the average committee size when agents use 1-NN as learning method. As we have seen in the accuracy evaluation, P-CCS is equivalent to the CCS with the Peer Counsel decision policy D_{PC} that we have used. Therefore, the average committee size convened by P-CCS is equal to that of CCS. The committees convened by B-CCS have a size between those of CCS and the individual agents. Moreover, we can also observe committees convened for solving a problem in the soybean and zoology data sets are larger than committees convened for solving a problem in the sponge data set.

Finally, Figure 5.9 shows the average committee size when agents use 3-NN as learning method. The situation is very similar to that shown in Figure 5.9: P-CCS convenes smaller committees than CCS, and B-CCS convenes even smaller committees than P-CCS. Moreover, we also observe that committees convened to solve a problem in the sponge data set are, as before, smaller than those convened to solve problems in the other two, thus showing again the adaptive behavior of P-CCS and B-CCS.

Summarizing, we can draw the following conclusions: using a collaboration strategy (CCS, P-CCS or B-CCS) is beneficial for individual agents in terms of classification accuracy. CCS is the collaboration strategy that achieves the highest classification accuracy in average, but also convenes the largest committees. Both P-CCS and B-CCS achieve much higher accuracies than individual agents and convene committees with an average size smaller than the those convened by CCS. Comparing P-CCS to B-CCS, B-CCS seems better, since it achieves a very similar accuracy (in fact, the difference in accuracy between P-CCS and B-CCS is very seldom statistically significant) and convenes smaller committees. Moreover, B-CCS and P-CCS have the advantage with respect to CCS that they can decide when an agent can solve the problem individually (by using the *Self-Competence* model or the *Committee-Competence* model). Finally, B-CCS and P-CCS are able to reach high accuracy values (close to that of CCS) and with smaller committees. However, the accuracy achieved by CCS is still higher than that of B-CCS or P-CCS.

5.5 Conclusions

In this chapter we have introduced the idea of *dynamic committee* collaboration strategies. A dynamic committee strategy is a collaboration strategy that convenes a different committee for each different problem that an agent wants to solve. Specifically, we have presented two dynamic committee collaboration strategies: *Peer Counsel Collaboration Strategy* (P-CCS) and *Bounded Counsel Collaboration Strategy* (B-CCS). The goal of the dynamic committee strategies is to reduce the number of agents involved in the solution of each problem: if an agent has a high confidence on the individually predicted solution for a problem, then it is not needed to convene a committee. Furthermore, if the solution predicted by a small committee of agents has a high confidence, then there is no need for convening a larger committee.

All the dynamic committee collaboration strategies use the information provided by *competence models* to make decisions. Specifically, competence models allow individual agents to assess when problems can be solved individually, or when convening a committee to solve them is needed. Moreover, competence models can be also used to assess the competence of a committee and to decide whether more agents have to be invited to join the committee or not. Although in this chapter we have used predefined competence models the next chapter will show that competence models can be automatically learnt by the individual agents.

Both collaboration strategies presented in this chapter, B-CCS and P-CCS, have been empirically compared against the Committee Collaboration Strategy (CCS) and against agents solving problems individually. Concerning individual accuracy, B-CCS, P-CCS, and CCS have shown to achieve higher classification accuracy values than agents solving problems individually. Moreover, concerning committee classification accuracy, CCS achieves the highest classification accuracy but convening larger committees. P-CCS and B-CCS achieve a

classification accuracy lower (but very close) than CCS and convening smaller committees.

The dynamic committees approach is also related to the MCBR approach by Leake and Sooriamurthi [53]. In the MCBR approach, when a CBR system solves a problem, it has to decide whether to retrieve cases from the local cases base or to dispatch the problem to an external case base where cases with a higher similarity to the target problem may be found. The main difference from the dynamic committee approach is that in the MCBR approach, the retrieve process of the CBR cycle is distributed while in the dynamic committee approach, the retrieve process is performed individually (thus preserving the privacy and autonomy of each individual agent) and the reuse process is distributed.

As a general conclusion, we have shown that dynamic committee collaboration strategies are useful collaboration strategies that can greatly improve the classification accuracy of an individual agent and that can be used without the high cost of using Committee Collaboration Strategy. Specifically, in a real system with a very large number of agents the dynamic committee strategies are preferable to the Committee Collaboration Strategy. Moreover, both B-CCS and P-CCS have shown that they are able to convene small committees for easy problems and convene large committees only for difficult problems. However, notice that both B-CCS and P-CCS have parameters in their decision policies that can modify their behavior. Those parameters need to be hand tuned in order to achieve the desired behavior. The next chapter presents a proactive learning strategy with which each agent will be able to acquire its individual competence models to be used in their decision policies. This proactive learning strategy avoids the need of hand-tuning the decision policy parameters in order to achieve the desired behavior.

Chapter 6

Proactive Learning for Collaboration

This chapter presents a proactive learning approach with which agents in a $\mathcal{M}$AC system can learn their own competence models. Then, the Proactive Bounded Counsel Collaboration Strategy (PB-CCS), a dynamic committee collaboration strategy that uses those learnt competence models, is presented. Finally, PB-CCS is empirically compared with CCS, B-CCS, and P-CCS.

6.1 Introduction

The previous chapter has introduced two dynamic committee collaboration strategies that use competence models in order to decide when to solve problems individually and when to convene a committee and also to decide how many agents will be invited to join the committee in the case that one has to be convened. Decision policies based on predefined *Self-Competence* and *Committee-Competence* competence models have parameters that need to be hand tuned and that can only be tuned by experimentation. Moreover, such hand tuned decision policies cannot be expected to automatically work well in every application domain or for every configuration of a $\mathcal{M}$AC system: the decision policy should be adapted to the specific application domain, to the degree of expertise that the agents in the $\mathcal{M}$AC system have, etc.

In this chapter we are going to present a proactive learning technique that will allow the individual agents to learn their own competence models to be used in the collaboration strategies, both to decide when to convene committees and to decide which agents to convene. Decision policies that use these learnt competence models do not need to be hand tuned for each application domain. Our goal is to study whether it is possible that individual agents learn their own specific competence models adapted to their specific application domain. Moreover, we will also define the *Proactive Bounded Counsel Collaboration Strategy* (PB-CCS), that uses the competence models learnt by this proactive learning

technique.

PB-CCS is a dynamic committee collaboration strategy based on the Bounded Counsel Collaboration Strategy. When an agent uses PB-CCS a committee of agents will be convened iteratively by inviting agents to join the committee one at a time. Therefore, at each iteration, an agent using PB-CCS to convene a committee must take two decisions:

1. The first decision to be made is whether it is needed to invite more agents to join the committee or not.

2. If more agents are needed, the agent has to decide which agent to invite next.

The iterative process of taking this two decisions is equivalent to deciding when to collaborate with other agents, and with which agents to collaborate. In this section we will show that, using a proactive learning technique, agents are able to learn by themselves how to take the previous two decisions. Specifically, the agents will learn competence models to be used in their individual decision policies.

Specifically, at each iteration t the set of agents $\mathcal{A}$ in the $\mathcal{MAC}$ system that have agreed to collaborate with the convener agent can be divided in two sets of agents: $\mathcal{A} = \mathcal{A}_t^c \cup \mathcal{A}_t^r$, where $\mathcal{A}_t^c$ contains the agents that have joined the committee at round t, and $\mathcal{A}_t^r$ contains the rest of the agents. At round $t = 0$, $\mathcal{A}_t^c = \{A_c\}$, i.e. only the convener agent is in the committee. The set $\mathcal{A}_t^r$ are the agents that are candidates to be invited to join the committee at a round $t + 1$. Figure 6.1 illustrate all these sets of agents, where we can see a $\mathcal{MAC}$ system composed of 10 agents: one of them is using PB-CCS to solve a problem P, 3 agents have not accepted to collaborate and 6 agents have agreed to collaborate ($\mathcal{A}$); from the 6 agents that have agreed to collaborate, 3 of them have already been invited to join the committee ($\mathcal{A}_t^c$), and 3 of them are still not in the committee ($\mathcal{A}_t^r$).

The structure of this chapter is as follows. First, Section 6.2 introduces the concepts related to competence models. Section 6.3 explains the proactive learning technique used to learn competence models; then, Section 6.4 presents the Proactive Bounded Counsel Collaboration Strategy (PB-CCS), a dynamic committee collaboration strategy that uses learnt competence models. Finally, PB-CCS is empirically evaluated at Section 6.5, comparing it with Committee Collaboration Strategy, the Peer Counsel Collaboration Strategy and the Bounded Counsel Collaboration Strategy.

6.2 Competence Models

In this section we are going to introduce the concepts needed to explain the proactive learning technique and the competence models that are to be learnt.

A competence model (see Definition 3.5.1) is a function that assesses the likelihood that the prediction made by an agent (or by a set of agents) for a

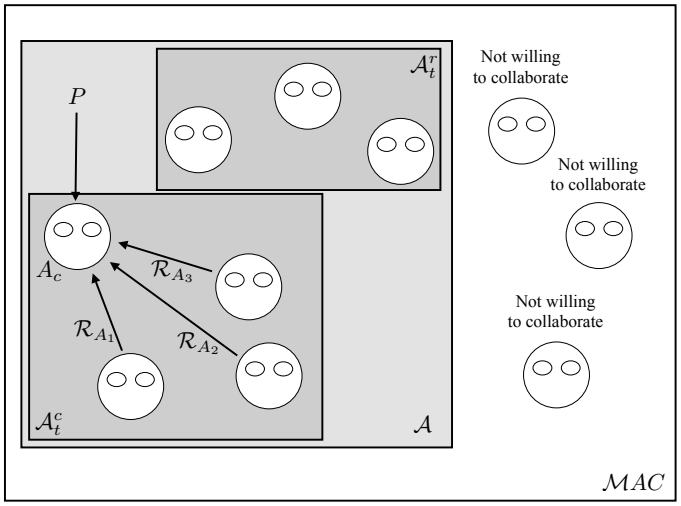


Figure 6.1: Illustration of PB-CCS at a round t.

given problem P is correct. In the dynamic committee collaboration strategies, competence models are used to decide when is better to invite new agents to join a committee, and which agents to invite. Therefore, competence models will be used for two purposes:

- To assess the competence of a given committee and decide whether inviting more agents to join the committee could improve performance.
- To assess the competence of agents that still have not been convened in order to decide which of them should be invited to join the committee.

A central issue for those decisions is to assess the competence of a set of collaborating agents $\mathcal{A}^c$, including the special case of a committee composed of a single agent (the convener agent), that corresponds to assessing the confidence of a single agent individually solving a problem.

Therefore, a competence model must assess the competence of an agent or group of agents given a voting situation, i.e. a situation in which committee has been convened and the convener agent is ready to apply a voting system to obtain a final prediction for the problem. Given the voting situation, the competence models will be used to decide the next action (invite more agents or not). Notice that the collection of SERs $\mathcal{R}_{\mathcal{A}^c}$ casted by the agent members of a committee $\mathcal{A}^c$ completely characterizes a voting situation (since from $\mathcal{R}_{\mathcal{A}^c}$ we can obtain which agents are members of the committee and which have been their votes).

Definition 6.2.1. *A* voting situation $\mathcal{R}_{\mathcal{A}^c}$ *is a set of SERs sent by a committee of agents* $\mathcal{A}^c$ *to the convener agent (including the SERs of the convener agent* A_c*) built to solve a problem* P*.*

For each voting situation we can define the candidate solution of a voting situation as the solution that the committee will predict if no more agents join the committee. Moreover, we can also define the individual candidate solution of an agent A_i in a committee as the solution that A_i individually predicts for the problem:

Definition 6.2.2. *The* candidate solution S^c *of a voting situation* $\mathcal{R}_{\mathcal{A}^c}$ *is the outcome of the voting system* $S^c = Sol(\mathcal{S}), P, \mathcal{R}_{\mathcal{A}^c},$.

Definition 6.2.3. *The* individual candidate solution $S^c_{A_i}$ *of an agent* A_i *in a voting situation* $\mathcal{R}_{\mathcal{A}^c}$ *is the outcome of the voting system using only the SERs provided by* A_i*:* $S^c_{A_i} = Sol(\mathcal{S}, P, \mathcal{R}_{A_i})$*, where* $\mathcal{R}_{A_i} = \{\mathbf{R} \in \mathcal{R}_{\mathcal{A}^c} | \mathbf{R}.A = A_i\}$.

Specifically, a competence model take as input a voting situation $\mathcal{R}_{\mathcal{A}^c}$ and outputs a confidence value in the interval $[0, 1]$ (Definition 3.5.1). The output represents the confidence that the candidate solution of the voting situation is correct. If the competence model is modelling the competence of a single agent A_i, then the output represents the confidence that the individual candidate solution of A_i is correct. Therefore, learning a competence model of an agent or a group of agents means constructing a mapping between voting situations and confidence values in $[0, 1]$.

Let us assume an agent A_i member of a $\mathcal{MAC}$ system composed of n agents, $\mathcal{A} = \{A_1, ..., A_n\}$. In order to use PB-CCS, A_i needs to learn several competence models, namely $\mathcal{M}_{A_i} = \{M_c, M_{A_1}, ..., M_{A_{i-1}}, M_{A_{i+1}}, ..., M_{A_n}\}$, where M_c is a *Committee-Competence Model* and M_{A_j} are *Agent-Competence Models*.

Definition 6.2.4. *A* Committee-Competence Model M_c *is a competence model that assesses the likelihood that a committee* $\mathcal{A}^c$ *will predict the correct solution in a given voting situation* $\mathcal{R}$.

The Committee-Competence models M_c assesses the confidence in the solution predicted by a committee in a specific voting situation $\mathcal{R}$ for a problem P, and is used to decide whether the current committee $\mathcal{A}^c_t$ is competent enough to solve the problem P or it is better to invite more agents to join the committee.

Definition 6.2.5. *An* Agent-Competence Model M_{A_j} *is a competence model that assesses the likelihood that an agent* A_j *will predict the correct solution in a given voting situation* $\mathcal{R}$.

The Agent-Competence models M_{A_j} assess the confidence of an agent A_j to correctly solve a problem P given the current voting situation $\mathcal{R}$. Thus, let us consider an agent $A_j \in \mathcal{A}^r$, i.e. an agent that has not yet been invited to join the committee and thus it is a candidate to be invited to join $\mathcal{A}^c$. M_{A_j} is useful for the convener agent to estimate to which degree A_j is a "good candidate", in other words, the degree of confidence in that inviting A_j will increase the likelihood that the enlarged committee will predict the correct solution. Therefore, the convener agent can use the Agent-Competence models to select which agent A_j is the best candidate to be invited to join the committee

by selecting the agent A_j for which its competence model predicts the highest confidence to find the correct solution for P given the current voting situation $\mathcal{R}$.

The next section presents the proactive learning process used by the agents to learn the competence models.

6.3 Proactive Learning of Competence Models

This section presents a proactive learning technique with which an agent A_i in a $\mathcal{M}$AC system can learn the competence models $\mathcal{M}_{A_i}$ to be used in PB-CCS. In order to learn these competence models, agents need to collect examples from where to learn. This section presents the way in which an agent can proactively collect those examples and how can a competence model be learnt from them.

The proactive learning technique consists of several steps: first, an agent A_i that wants to learn a competence model obtains a set of cases with known solution (that can be taken from its local case base), and those cases are transformed to problems (by removing their solutions); the agent sends then those problems to other agents and obtains their individual predictions for the problems; with the predictions made by the other agents for all the problems sent, A_i will construct a set of voting situations; finally, these voting situations will be the input of a learning algorithm from which the competence models will be learnt.

Moreover, in order to apply standard machine learning techniques, we need to characterize the voting situations by defining a collection of attributes in order to express them as attribute-value vectors.

Definition 6.3.1. *The* characterization of a voting situation $\mathcal{R}_{\mathcal{A}_t^c}$ *is a tuple* $\langle A_1, ..., A_n, S^c, V^c, V^r, \rho \rangle$*, where:*

- *The attributes* $A_1, ..., A_n$ *are boolean.* $A_i = 1$ *if* $A_i \in \mathcal{A}_t^c$ *(i.e. if* A_i *is a member of the current committee), and* $A_i = 0$ *otherwise.*
- $S^c = Sol(\mathcal{S}, P, \mathcal{R}_{\mathcal{A}_t^c})$ *is the candidate solution.*
- $V^c = Ballot(S^c, \mathcal{A}^c)$ *are the votes for the candidate solution.*
- $V^r = (\sum_{S_k \in \mathcal{S}} Ballot(S_k, \mathcal{A}^c)) - V^c$ *is the sum of votes for all the other solutions.*
- $\rho = \frac{V^c}{V^c + V^r}$ *is the ratio of votes supporting the candidate solution.*

Notice that the values of most features are computed from the votes cast by each agent. Therefore, the values of this characterization depend on the voting system used. Notice also that, as we have said in Section 4.5.4, BWAV provides more information than Approval Voting, and therefore characterizations of voting situations built using BWAV contain more information.

Induction will be used to learn the competence models $\mathcal{M}_{A_i}$ of an agent A_i from voting situations. Thus, a competence model $M \in \mathcal{M}_{A_i}$ will be learnt by

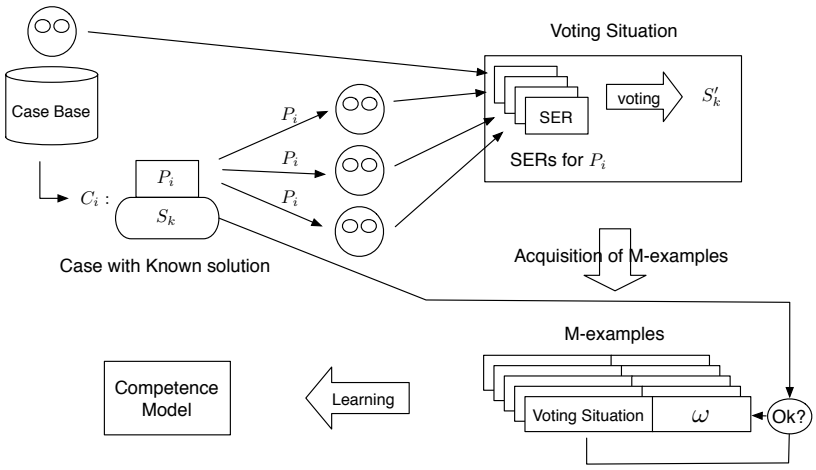


Figure 6.2: Detailed graphical representation of the proactive learning technique to learn competence models.

collecting a set of examples to form a data set and inducing the competence model from them. We will call an example collected to learn a competence model M an *M-example*.

Definition 6.3.2. *An* M-example m *derived from a case* c *is a pair* $m = \langle\langle A_1, ..., A_n, S^c, V^c, V^r, \rho\rangle, \omega\rangle$, *where* $\langle A_1, ..., A_n, S^c, V^c, V^r, \rho\rangle$ *is the characterization of a voting situation* $\mathcal{R}_{\mathcal{A}_t^c}$ *according to Definition 6.3.1 and* ω *represents the "prediction correctness" of the voting situation, such that* $\omega = 1$ *if the candidate solution of the voting situation* $\mathcal{R}_{\mathcal{A}_t^c}$ *was the correct one (i.e. if* $S^c = c.S$*) and* $\omega = 0$ *otherwise (if* $S^c \neq c.S$*).*

Competence models can be learnt by collecting sets of M-examples. Moreover, in order to learn the competence models $\mathcal{M}_{A_i}$, an agent A_i must collect a set of M-examples for each competence model $M \in \mathcal{M}_{A_i}$, i.e. a set of M_c-examples, a set of M_{A_1}-examples, and so on. In the rest of this section, we will present the proactive learning technique that the agents in a $\mathcal{M}$AC system use in order to acquire all the needed M-examples, and learn from them the required competence models.

Figure 6.2 presents a scheme of the proactive learning process that will be explained in the remaining sections of this chapter. Specifically, the steps involved in the proactive learning process are the following ones:

1. An agent that wants to learn a competence model M, obtains a set of cases with known solution by taking cases from its local case base.

2. Those cases are transformed into problems by removing their solution and are sent to other agents in the $\mathcal{M}$AC system in order to obtain their individual predictions.

3. Voting situations are then built from these individual predictions (Definition 6.3.3). And from these voting situations, M-examples are constructed (Definition 6.3.6).

4. Finally, with the collection of M-examples, a competence model is learnt using an inductive algorithm.

These four steps will be presented in detail in the rest of this section. Then, an exemplification of the process will be presented in Section 6.3.3. After that, Section 6.4 presents the Proactive Bounded Counsel Collaboration Strategy. Finally, PB-CCS is experimentally evaluated in Section 6.5.

6.3.1 Acquisition of *M-examples*

In this section we are going to present the proactive process that an agent follows in order to acquire M-examples from where to learn the competence models.

Since an agent A_i needs the competence models $\mathcal{M}_{A_i}$, a different training set T_M will be needed to learn each competence model $M \in \mathcal{M}_{A_i}$. We will call $\mathcal{T}_{A_i} = \{T_{M_c}, T_{M_{A_1}}, ...T_{M_{A_{i-1}}}, T_{M_{A_{i+1}}}, ..., T_{M_{A_n}}\}$ to the collection of training sets needed by an agent to learn the competence models.

For example, when A_i is building a competence model M_c of the committee, A_i sends a problem $c.P$ to the rest of agents in the $\mathcal{M}$AC system. After receiving their predictions, A_i builds the voting situation resulting of putting together all the SERs built by the agents. Then, A_i uses the voting system to determine whether the candidate solution of that voting situation is correct or not. If the candidate solution for the problem $c.P$ is correct, then A_i can build an M_c-example with $\omega = 1$, and if the prediction is incorrect, A_i can build an M_c-example with $\omega = 0$.

Notice that the output of a confidence model is a number in the interval $[0, 1]$, but the ω values in the M-examples are only 1 or 0. The reason is that when an M-example is built there are only two possible situations: that the agent (or set of agents) have predicted the correct solution or not.

Although the proactive learning technique can be applied to learn each individual competence model one at a time, this is not necessary since all competence models can be learnt at the same time. Specifically, an agent A_i that wants to obtain the collection of training sets needed to learn the competence models proceeds as follows.

Definition 6.3.3. Acquisition of Individual Predictions*:*

1. *A_i chooses a subset of cases $B_i \subseteq C_i$ from its individual case base.*

2. *For each case $c \in B_i$:*

   (a) *A_i uses I_{CCS} (the interaction protocol of CCS) to convene a committee of agents $\mathcal{A}^c$ to solve the problem $c.P$. After this, A_i has obtained the SERs built by all the rest of agents in $\mathcal{A}^c$ for problem $c.P$.*

(b) A_i solves c.P using a leave-one-out *method, i.e. it solves c.P using as case base $C_i - c$ and creates its own set of SERs $\mathcal{R}_{A_i}$.*

(c) With the set $\mathcal{R}_{\mathcal{A}^c}$ of SERs obtained (that includes all the SERs from the other agents obtained in step (a) and the SERs of A_i computed in (b)), A_i builds voting situations from where to construct M-examples (as explained below in Definition 6.3.6).

Notice that A_i can build more than one voting situation from the collection $\mathcal{R}_{\mathcal{A}^c}$ of SERs in Step 2.(c). For instance, the set of SERs built by A_i, $\mathcal{R}_{A_i} \subseteq \mathcal{R}_{\mathcal{A}^c}$ corresponds to a voting situation where only agent A_i has cast votes. The set of SERs built by A_i and any other agent A_j, $(\mathcal{R}_{A_i} \cup \mathcal{R}_{A_j}) \subseteq \mathcal{R}_{\mathcal{A}^c}$ corresponds to a voting situation where A_i and A_j have cast their votes. In the following, we will write $\mathcal{R}_{\mathcal{A}'}$ to refer to the set of SERs built by a set of agents $\mathcal{A}'$.

Definition 6.3.4. *A* Valid Voting Situation *$\mathcal{R}_{\mathcal{A}'}$ for an agent A_i and a problem c.P is a voting situation where A_i has casted its votes, i.e. a set of SERs built by a set of agents $\mathcal{A}'$ that at least contains A_i. Specifically, $\mathcal{R}_{\mathcal{A}'} \subseteq \mathcal{R}_{\mathcal{A}^c}$ such that $\mathcal{A}' \subseteq \mathcal{A}^c$ and $A_i \in \mathcal{A}'$.*

Intuitively, a valid voting situation for an agent A_i is one in which A_i itself is a member of the committee. In other words, a valid voting situation can be built by selecting the set of SERs built by any subset of agents $\mathcal{A}' \subseteq \mathcal{A}^c$ (such that $A_i \in \mathcal{A}'$). We can define the set of all the possible subsets of agents of $\mathcal{A}$ that contain at least A_i as $\mathbb{A}(A_i) = \{\mathcal{A}' \in \mathcal{P}(\mathcal{A}) | A_1 \in \mathcal{A}'\}$, where $\mathcal{P}(\mathcal{A})$ represents the parts of the set $\mathcal{A}$ (i.e. the set of all the possible subsets of $\mathcal{A}$). Now, it is easy to define the set of all the possible Valid Voting Situations for an agent A_i that can be constructed from $\mathcal{R}_{\mathcal{A}^c}$ as follows:

Definition 6.3.5. *The* Set of Valid Voting Situations *for an agent A_i is: $\mathbb{V}(A_i) = \{\mathcal{R}_{\mathcal{A}'}\}_{\mathcal{A}' \in \mathbb{A}(A_i)}$, where $\mathcal{R}_{\mathcal{A}'}$ represents the set of SERs built by the set of agents $\mathcal{A}'$.*

Using the previous definitions, we can decompose Step 2.(c) above in three sub-steps: first, a sample of all the possible valid voting situations that can be built from $\mathcal{R}_{\mathcal{A}^c}$ is taken, then each one of the selected valid voting situations are characterized and finally M-examples from each of them are built.

Definition 6.3.6. Acquisition of M-examples*:*

1. *A_i takes a sample of all the possible Valid Voting Situations that can be built: $\mathbb{V}' \subseteq \mathbb{V}(A_i)$ (see Section 6.3.1.2).*

2. *For every voting situation $\mathcal{R} \in \mathbb{V}'$, the agent A_i determines the characterization of the voting situation $\langle A_1, ..., A_n, S^c, V^c, V^r, \rho \rangle$ following Definition 6.3.3.*

3. *With this characterization A_i can build M-examples. Specifically, A_i will build one M -example for each competence model $M \in \mathcal{M}_{A_i}$ (see Section 6.3.1.1).*

Section 6.3.1.1 presents the process of building specific M-examples for each competence model $M \in \mathcal{M}_{A_i}$. Then, Section 6.3.1.2 presents the sampling technique used to take a sample of Valid Voting Situations.

6.3.1.1 Constructing M-examples from Voting Situations

Let us focus now on how M-examples are constructed for each specific competence model $M \in \mathcal{M}_{A_i}$. Depending on whether the M-example is built for the competence model M_c (that models the competence of the committee) or for a competence model M_{A_j} (that models the competence of an agent A_j), an agent A_i proceeds as follows in order to build an M-example from a voting situation $\mathcal{R}$:

- To build an M_c-example, A_i determines the candidate solution $S^c = Sol(\mathcal{S}, c.P, \mathcal{R}_{\mathcal{A}'},)$ obtained by applying the voting system to all the SERs in $\mathcal{R}_{\mathcal{A}'}$. If $Sol(\mathcal{S}, c.P, \mathcal{R}_{\mathcal{A}'}) = c.S$, then the following M_c-example is built

  $$m = \langle\langle A_1, ..., A_n, S^c, V^c, V^r, \rho\rangle, 1\rangle$$

  where $\omega = 1$ because the M-example characterizes a voting situation where the predicted solution is correct.

  If $S^c \neq c.S$, then the following M_c-example is built

  $$m = \langle\langle A_1, ..., A_n, S^c, V^c, V^r, \rho\rangle, 0\rangle$$

  where $\omega = 0$ because the M-example characterizes a voting situation where the predicted solution is not correct.

- To build an M_{A_j}-example, A_i determines the individual candidate solution yield by A_j, i.e. $S^c_{A_j} = Sol(\mathcal{S}, c.P, \mathcal{R}_{A_j})$. If $S^c_{A_j} = c.S$ (i.e. the prediction of A_j is correct), then the following M_{A_j}-example is built:

  $$m = \langle\langle A_1, ..., A_n, S^c, V^c, V^r, \rho\rangle, 1\rangle$$

  and if $S^c_{A_j} \neq c.S$ (i.e. the prediction of A_j is incorrect), then the following M_{A_j}-example is built

  $$m = \langle\langle A_1, ..., A_n, S^c, V^c, V^r, \rho\rangle, 0\rangle$$

6.3.1.2 Bounded Sampling of M-examples

Notice that with each voting situation $\mathcal{R} \in \mathbb{V}'$, an M-example can be constructed for each different competence model in $\mathcal{M}_{A_i}$. Therefore, the larger the size of $\mathbb{V}' \subseteq \mathbb{V}(A_i)$, the larger the number of M-examples that can be constructed. The size of $\mathbb{V}(A_i)$ (that is equivalent to the size of $\mathbb{A}(A_i)$ depends on the number of agents in the committee convened to solve each of the problems

Size	1	2	3	4	5	6	7	8	9
Total	1	8	28	56	70	56	28	8	1
Random (100)	0	3	11	22	28	22	11	3	0
Bounded (100)	1	8	17	17	16	16	16	8	1

Table 6.1: Number of different subsets of $\mathcal{A}^c$ of different sizes.

$c.P$ (where $c \in B_i \subseteq C_i$). In fact, the size of $\mathbb{V}(A_i)$ grows exponentially with the size of the set of convened agents: there are 2^{n-1} different Valid Voting Situations for a $\mathcal{MAC}$ system with n agents. Therefore, building all the M-examples that can be derived from all possible valid voting situations in $\mathbb{V}(A_i)$ may be unfeasible or impractical. Thus, an agent using the proactive learning technique to learn competence models will take a sample $\mathbb{V}' \subseteq \mathbb{V}(A_i)$ from where to build M-examples.

The number of M-examples that an agent builds for each competence model M is about $\#(B_i) \times \#(\mathbb{V}')$. In our experiments we have imposed the limit of at most 2000 M-examples for each competence model. Therefore, the agents in our experiments will take subsets $\mathbb{V}' \subseteq \mathbb{V}(A_i)$ to have at most $2000/\#(B_i)$ voting situations. Moreover, in our experiments, an agent A_i using the proactive learning technique uses all the case base C_i as the set B_i (i.e. $B_i = C_i$) (in order to maximize the diversity in the set of voting situations built), and therefore the size of $\mathbb{V}'$ will be at most $2000/\#(C_i)$.

Notice that the agents need a sampling method to compute a specific subset $\mathbb{V}' \in \mathbb{V}(A_i)$. This section presents the *Bounded Sampling* technique that we have used in our experiments.

A random selection of subsets of agents of $\mathcal{A}^c$ is a possible sampling method, but we will show that some problems arise. Let us illustrate these problems with an example: Let us assume a $\mathcal{MAC}$ system composed of 9 agents $\mathcal{A} = \{A_1, ..., A_9\}$. The agent A_1 has selected a case $c \in B_i \subseteq C_i$, and has convened a committee to solve problem $c.P$. With the SERs received from the agents convened in the committee and from the SERs built by A_1 itself using a leave-one-out method A_1 has constructed the set $\mathcal{R}_{\mathcal{A}^c}$ of SERs. Assume that all the agents in the $\mathcal{MAC}$ system have agreed to join the committee, and that $\mathcal{A}^c$ contains 9 agents. Notice that there is only 1 subset of $\mathcal{A}^c$ of size 1 containing A_1, 8 subsets of size 2, 28 of size 3, and so on. The first row on Table 6.1 shows how many subsets of each different size can be drawn from $\mathcal{A}^c$. If A_1 takes a random sample of subsets, it is very likely that a large number of subsets have size 4, 5, or 6, and very few of them have size 1, 2, 3, 7, 8, or 9. This means that the M-examples generated will mostly represent situations where 4, 5 or 6 agents have cast votes. Therefore, the competence models learnt from these M-examples will be highly biased towards these situations.

A way to overcome the previous problem is performing *Bounded Sampling* of subsets. Bounded Sampling will try to balance the number of subsets of each size taken from $\mathcal{A}^c$. In this section we will present a sampling algorithm that will take a sample of subsets of agents of $\mathcal{A}^c$. From this sample of subsets of

Function Bounded Sampling($\mathcal{A}^c$, m):
$\mathbb{A}' = \emptyset$;
While $\#(\mathbb{A}') < m$, do:
For $k = 1$ to $\#(\mathcal{A}^c)$, do:
If there are still subsets of $\mathcal{A}^c$ of size k that are not in $\mathbb{A}'$
Then select a subset $\mathcal{A}' \subseteq \mathcal{A}^c$ of size k at random and
$\mathbb{A}' := \mathbb{A}' \cup \{\mathcal{A}'\}$;
Otherwise, do nothing;
End For;
End While;
Return $\mathbb{A}'$;
End Function

Figure 6.3: Algorithm to take a bounded sample of the subsets of $\mathcal{A}^c$

agents, the sample of Valid Voting Situations will be built, i.e. the Bounded Sampling algorithm will built a sample $\mathbb{A}' \subseteq \mathbb{A}(A_i)$; then the sample of Valid Voting Situations will be built as $\mathbb{V}' = \bigcup_{\mathcal{A}' \in \mathbb{A}'} \{\mathcal{R}_{\mathcal{A}'}\}$. The Bounded Sampling algorithm that makes a bounded sample $\mathbb{A}'$ of size m is shown in Figure 6.3.

For example, let us assume that we take a random sample of size $m = 100$ in the previous example of 9 agents. The most probable distribution of subsets of different sizes is shown in the second row of Table 6.1, where we can see the high number of subsets with 4, 5, and 6 agents. Third row in Table 6.1 shows the distribution of subsets of different sizes obtained with the bounded sampling algorithm with $m = 100$. As the table shows, the number of subsets of each size is now more balanced.

From each subset of $\mathcal{A}^c$ selected, A_1 can build a Valid Voting Situation from where to create an M-example for each competence model in $\mathcal{M}_{A_i}$.

The result of the process explained in this and the previous sections is a collection of M-examples for each competence model $M \in \mathcal{M}$, i.e. a collection of training sets. The next section explains how to learn a competence model from these training sets. After that, an exemplification of the process of collecting M-examples and learning competence models is presented.

6.3.2 Induction of the Competence Models

Once an agent A_i has collected enough *M-examples*, good competence models can be learnt. In our experiments we have used an induction algorithm based on decision trees but with several considerations:

1. Numerical attributes are discretized. Each numeric attribute a is discretized to have just 2 possible values. The discretization is performed by computing a cutpoint κ. Left branch of the decision tree will have the M-examples with $value(a) \leq \kappa$ and in the right branch all the M-examples whith $value(a) > \kappa$.

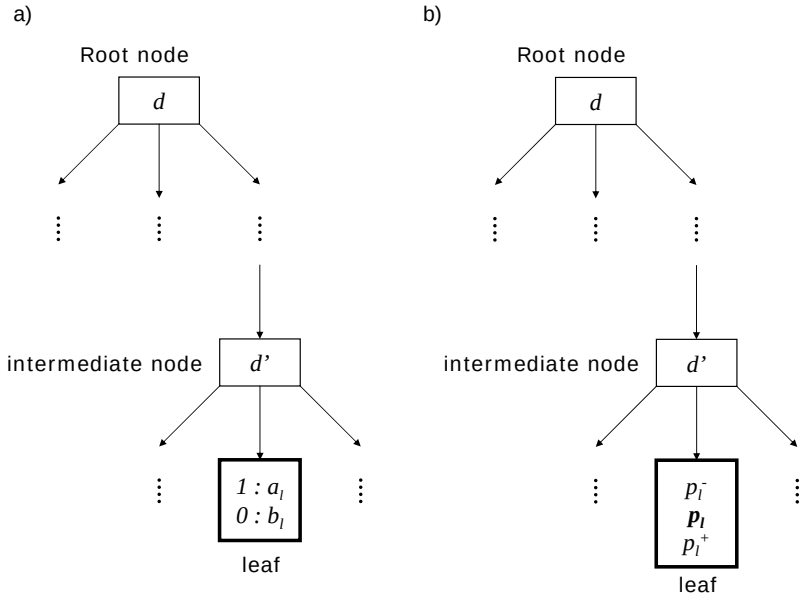


Figure 6.4: a) A decision tree holding the number of M-examples with each confidence value in each leaf. b) Confidence tree computed from the decision tree shown in a).

2. Error-based pruning of the tree is used to avoid overfitting.

3. As M-examples do not have many attributes, each leaf of the tree there will likely have a mix of examples with different confidence values. Figure 6.4.a shows a schematic view of a decision tree such that in each leaf l, the number of M-examples with confidence 1 and with confidence 0 is shown: a_l represents the number of M-examples with confidence 1 and b_l represents the number of M-examples with confidence 0.

Since the decision tree learnt has to be used to assess confidence values (i.e. real numbers in the interval $[0, 1]$), we will generate a *confidence tree* from the decision tree learnt. This confidence tree will be the one able to assess the confidence values.

Definition 6.3.7. *A* confidence tree *is a structure consisting on two types of nodes:*

- *A* decision node*, that contains a decision d. For each possible answer to the decision d, the decision node points towards another confidence tree (The top decision node is called the* root node*, and the rest are called the* intermediate nodes*),*

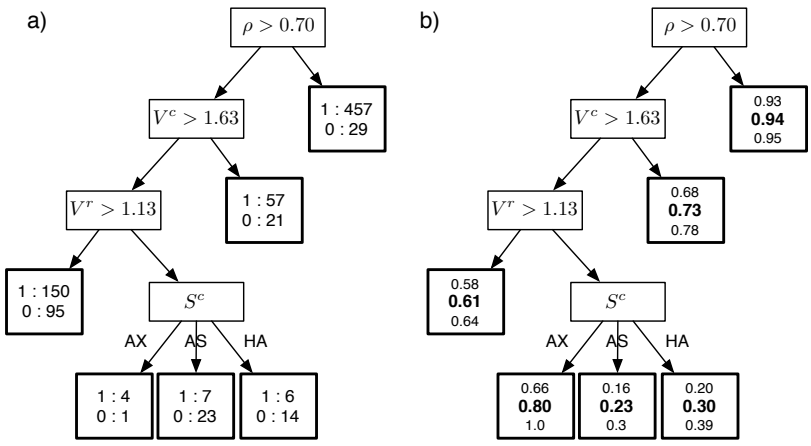


Figure 6.5: a) Decision tree learnt as the competence model M_c in a $\mathcal{M}$AC system composed of 5 agents. b) Confidence tree computed from the decision tree shown in a). For the numerical attributes, the right branches of each node contain the M-examples that match the condition in the node. AS, AS and HA are the possible solution classes in $\mathcal{S}$. The left figure shows the number of M-examples with each confidence value that have fallen in each tree, and the right figure shows the estimation of the confidence in each leaf.

- *or a* leaf node l*, that contains three real numbers:* p_l^-*,* p_l*, and* p_l^+ *(such that* $p_l^- \leq p_l \leq p_l^+$*). Where* p_l *is the expected confidence of a voting situation that is classified in leaf* l*, and* p_l^- *and* p_l^+ *are respectively, the pessimistic and optimistic estimations of that confidence.*

Confidence trees are generated from decision trees in the following way (Figure 6.4.b illustrates this process):

- Decision nodes are preserved without changes.
- Each leaf node l in a confidence tree contains 3 values, computed from the values a_l and b_l of the corresponding leaf node of the decision tree:
  - $p_l = (1/(a_l + b_l)) * (1 * a_l + 0 * b_l)$ is the expected confidence of an M-example classified in leaf l.
  - p_l^-: the pessimistic estimation of the confidence of the confidence of an M-example classified in that leaf l (see below).
  - p_l^+: the optimistic estimation of the confidence of the confidence of an M-example classified in that leaf l (see below).

Figure 6.5 shows an example of the conversion from a decision tree (on the left) to a confidence tree (on the right). On each leaf l of the confidence tree, the three values p_l^-, p_l, and p_l^+ (such that $p_l^- \leq p_l \leq p_l^+$) are shown. Notice

that p_l is the expected confidence of a voting situation that is classified in leaf l. However, since p_l is just an estimation of the confidence, if the number of M-examples in the leaf node l is small then p_l may be a poor estimator of the confidence of the voting situations classified on the leaf l. The greater the number of M-examples in leaf l, the better the estimation of the confidence. To solve this problem, instead of computing a single value as the confidence estimation, the agents will compute an interval, $[p_l^-, p_l^+]$, that ensures with 66% certainty that the real confidence value is in that interval. This interval depends on the number of examples in leaf l: the greater the number of M-examples, the narrower the interval will be (see Appendix C for a detailed explanation on how to compute this interval). In Figure 6.5.b, p_l^- and p_l^+ are shown above and below p_l respectively. For instance, if we look at the right most leaf in Figure 6.5 (the one with 457 M-examples with confidence 1 and 29 M-examples with confidence 0), we can see that the estimated p_l is 0.94 and the interval is $[0.93, 0.95]$, a very narrow interval since there are a lot of M-examples to compute the estimation of the confidence.

Moreover, since the tree shown in Figure 6.5 corresponds to the M_c competence model, the confidence p_l in each leaf has to be interpreted as follows: if a voting situation $\mathcal{R}_{\mathcal{A}^c}$ is classified in a leaf l by the competence model M_c, the expected confidence of the committee prediction for a problem in the voting situation represented by $\mathcal{R}_{\mathcal{A}^c}$ is p_l. p_l^- and p_l^+ are respectively the pessimistic and optimistic confidence estimation. The interpretation of the expected confidence p_l of a prediction is that the $p_l \times 100\%$ of the times that prediction should be correct.

For the purposes that competence models will have in the dynamic committee collaboration strategies, pessimistic estimation is more safe than any other estimation (expected p_l or optimistic p_l^+). Using pessimistic estimations of the confidence the convener agent will always underestimate the confidence on the predictions of the committee, and therefore will tend to decide to invite more agents to the committee more often than if the estimation is optimistic. Therefore, the worst that can happen is that the committee convened to solve a problem is larger than in should be. However, if we make a more optimistic estimation of the confidence, (using the expected p_l or optimistic p_l^+ estimations) the convener agent may stop inviting agents too early, thus failing to correctly solve a problem more often. Therefore, since confidence trees will be used as competence models the output of a confidence tree can be defined as:

Definition 6.3.8. *The* output of a confidence tree M *for a given voting situation* $\mathcal{R}_{\mathcal{A}^c}$ *is* $M(\mathcal{R}_{\mathcal{A}^c}) = p_l^-$*, where* l *is the leaf of the confidence tree in which* $\mathcal{R}_{\mathcal{A}^c}$ *has been classified.*

For instance, imagine that an agent is using the confidence tree shown in Figure 6.5.b to assess the confidence on the prediction made by a committee in a voting situation characterized by $\mathcal{R}_{\mathcal{A}^c}$. If the number of votes for the candidate solution is larger than 70% of the total number of votes in $\mathcal{R}_{\mathcal{A}^c}$ will be classified in the right most leaf, containing the confidence values $(0.93, 0.94, 0.95)$. Thus,

the confidence value assessed for the prediction of the committee characterized by $\mathcal{R}_{\mathcal{A}^c}$ will be 0.93.

The next section presents an exemplification of the proactive learning technique used to learn the confidence trees that will be used as the competence models in the Proactive Bounded Counsel Collaboration Strategy.

6.3.3 Exemplification

In order to clarify the M-example acquisition process, we will describe an exemplification with a system composed of 3 agents $\mathcal{A} = \{A_1, A_2, A_3\}$. The agent A_1 is collecting M-examples to learn the competence models needed in the Proactive Bounded Counsel Collaboration Strategy. A_1 should learn three competence models: $\mathcal{M}_{A_1} = \{M_c, M_{A_2}, M_{A_3}\}$.

For that purpose, A_1 has selected a subset $B_1 \subseteq C_1$ of cases from its individual case base C_1. All the cases in B_1 will be used to acquire M-examples. For instance, A_1 selects one of these cases $c \in B_1$ with solution $c.S = S_1$, and convenes a committee to solve the problem $c.P$. Both A_2 and A_3 accept to join the committee, and send the following SERs to A_1: A_2 sends $\mathbf{R}_2 = \langle S_1, 3, c.P, A_2\rangle$ and A_3 sends $\mathbf{R}_3 = \langle S_2, 1, c.P, A_3\rangle$. Finally, A_1 has built the SER $\mathbf{R}_1 = \langle S_1, 2, c.P, A_1\rangle$ using a leave-one-out method. Therefore, A_1 has collected the set of SERs $\mathcal{R}_{\mathcal{A}^c} = \{\mathbf{R}_1, \mathbf{R}_2, \mathbf{R}_3\}$ from the set of agents $\mathcal{A}^c = \{A_1, A_2, A_3\}$.

There are 4 possible subsets of $\mathcal{A}^c$ that contain A_1, namely $\mathbb{A}(A_1) = \{\{A_1\}$, $\{A_1, A_2\}$, $\{A_1, A_3\}, \{A_1, A_2, A_3\}\}$. Assume that the agent A_1 chooses the collection $\mathbb{A}' = \{\{A_1\}, \{A_1, A_2\}, \{A_1, A_3\}\}$ of subsets of agents to build voting situations from where to construct M-examples.

From the first subset of agents $\mathcal{A}' = \{A_1\}$, the following voting situation $\mathcal{R}' = \{R_1\}$ is built. A_1 computes the attributes that characterize the voting situation $\mathcal{R}'$: $(1, 0, 0, S_1, 0.66, 0.00, 1.00)$. From this voting situation, the three following M-examples can be built:

- An M_c-example: $\langle(1, 0, 0, S_1, 0.66, 0.00, 1.00), 1\rangle$, since the candidate solution S_1 is the correct one.

- An M_{A_2}-example: $\langle(1, 0, 0, S_1, 0.66, 0.00, 1.00), 1\rangle$, since the SER of agent A_2 endorses the correct solution class S_1. It is important to understand that this M_{A_2}-example characterizes a situation where A_1 has voted, the candidate solution of the current committee (containing only A_1) is S_1 and A_2 has still not joined the committee. A confidence value $\omega = 1$ means that in this situation A_2 has predicted the correct solution class S_1.

- An M_{A_3}-example: $\langle(1, 0, 0, S_1, 0.66, 0.00, 1.00), 0\rangle$, since the SER of agent A_3 endorses an incorrect solution class S_2. As in the previous situation, it is important to understand that this M_{A_3}-example characterizes a situation where A_1 has voted, the candidate solution of the current committee (containing only A_1) is S_1 and A_3 has still not joined the committee. A

confidence value $\omega = 0$ means that in this situation A_3 has predicted an incorrect solution class.

From the second subset of agents $\mathcal{A}' = \{A_1, A_2\}$, the following voting situation $\mathcal{R}' = \{R_1, R_2\}$ is built. The characterization is $(1, 1, 0, S_1, 1.41, 0.00, 1.00)$, and the M-examples that can be built are:

- An M_c-example: $\langle(1, 1, 0, S_1, 1.41, 0.00, 1.00), 1\rangle$, since the candidate solution S_1 is the correct one.
- An M_{A_3}-example: $\langle(1, 1, 0, S_1, 1.41, 0.00, 1.00), 0\rangle$, since the SER of agent A_3 endorses an incorrect solution class S_2.

Notice that no M_{A_2}-example is built from this voting situation, since A_2 is a member of the committee corresponding to the characterized voting situation.

Finally, from the third subset of agents $\mathcal{A}' = \{A_1, A_2\}$, the following voting situation $\mathcal{R}' = \{R_1, R_2\}$ is built. The characterization is $(1, 0, 1, S_1, 0.66, 0.50, 0.57)$, and the M-examples that can be built are:

- An M_c-example: $\langle(1, 0, 1, S_1, 0.66, 0.50, 0.57), 1\rangle$, since the candidate solution S_1 is the correct one.
- An M_{A_2}-example: $\langle(1, 0, 1, S_1, 0.66, 0.50, 0.57), 1\rangle$, since the SER of agent A_2 endorses the correct solution class S_1.

Therefore, with just a single case $c \in B_i$, the agent A_i has built 3 M_c-examples, 2 M_{A_2}-examples and 2 M_{A_3}-examples. After A_1 has collected M-examples using all the cases in B_i, 3 training sets will be built: T_{M_c}, $T_{M_{A_2}}$, and $T_{M_{A_3}}$. From these 3 training sets, A_1 can now induce the corresponding confidence trees to be used as the competence models M_c, M_{A_2}, and M_{A_3}. Similarly, agents A_2 and A_3 can also use the same technique to acquire their respective competence models if they need them. Notice that each agent in a $\mathcal{M}$AC system is free to use the collaboration strategies and decision policies that it prefers. Therefore, if A_1 uses the proactive learning technique to learn its own competence models, A_2 and A_3 are not forced to use it. Each agent will acquire its competence models as it prefers.

6.4 Proactive Bounded Counsel

In this section we are going to define the Proactive Bounded Counsel Collaboration Strategy (PB-CCS), that uses the competence models learnt by the proactive learning technique presented in Section 6.3.

Using PB-CCS, an agent tries to convene a committee that maximizes the confidence of correctly solving a concrete problem P. Moreover, selecting such a committee is a difficult task, and we are going to present an iterative approach where the convener agent will select agents one by one (as in B-CCS) trying to convene a committee that is competent enough to solve P. Notice that all the

agents in the $\mathcal{M}$AC system could have used the proactive learning technique to learn the competence models. Therefore, any agent in the $\mathcal{M}$AC system can act as the convener agent. Moreover, we must bear in mind that, as in the other collaboration strategies, the convener agent has polled the agents in the $\mathcal{M}$AC system to know the set of agents $\mathcal{A}$ that are willing to collaborate with him before starting PB-CCS.

In order to use PB-CCS the convener agent must be able make two decisions: when to stop inviting agents to join the committee, and which agent to invite at each round of PB-CCS. Therefore, the convener agent needs two individual decision policies, namely a *Halting* decision policy and an *Agent Selection* decision policy. Moreover, the convener agent also needs a voting decision policy in order to aggregate all the individual predictions.

Definition 6.4.1. *The* Proactive Bounded Committee Collaboration Strategy *(PB-CCS) is a collaboration strategy* $\langle I_{PB-CCS}, D_H, D_{AS}, D_{BWAV} \rangle$*, where* I_{PB-CCS} *is the PB-CCS interaction protocol shown in Figure 6.7,* D_H *is the Proactive Bounded Counsel* Halting *decision policy,* D_{AS} *is the Proactive Bounded Counsel* Agent Selection *decision policy and* D_{BWAV} *is the voting decision policy based on BWAV (see Section 4.3).*

When an agent A_i wants to solve a problem P using PB-CCS proceeds as follows: first A_c solves the problem individually; then, A_c uses its Halting decision policy D_H to decide whether the confidence in the individually found solution is high enough and no committee needs to be convened, or if the confidence is not high enough, and a committee must be convened to solve P. If a committee has to be convened, A_i uses its Agent Selection decision policy D_{AS} to decide which is going to be the first agent A_{j_1} to invite. When A_{j_1} joins the committee, it solves the problem P, and sends its prediction to A_i, who uses BWAV in order to aggregate both predictions. Then, D_H is used again to decide whether more agents have to join the committee or not. This process is repeated until D_H decides that the confidence on the prediction found by the current committee is high enough.

Since PB-CCS is an iterative collaboration strategy consisting in a series of rounds, we will use t to note the current round of the protocol, $\mathcal{A}_t^c$ to note the subset of agents of $\mathcal{A}$ that have joined the committee and $\mathcal{A}_t^r$ to note the subset of agents of $\mathcal{A}$ that have still not been invited to join the committee. Finally, we will note $\mathcal{R}_{\mathcal{A}_t^c}$ the set of all the SERs submitted to the convener agent by all the agents in $\mathcal{A}_t^c$ (included the SERs built by the convener agent A_c itself) for the problem P, i.e. $\mathcal{R}_{\mathcal{A}_t^c}$ represents the voting situation at round t.

In the following sections we will present both the decision policies and the interaction protocol used in the Proactive Bounded Counsel Collaboration Strategy.

6.4.1 Proactive Bounded Counsel Policies

This section presents the two decision policies D_H and D_{AS}, since D_{BWAV} is the Bounded Weighted Approval Voting voting system already presented in

Section 4.3. In order to implement those two decision policies an agent A_i uses the competence models $\mathcal{M}_{A_i} = \{M_c, M_{A_1}, ..., M_{A_{i-1}}, M_{A_{i+1}}, ..., M_{A_n}\}$, that are exactly those learnt during the proactive learning process.

Using those competence models, we can define the Proactive Bounded Counsel Halting decision policy D_H as a boolean decision policy that decides whether the convener agent can stop inviting agents to the committee at a round t; i.e. if $D_H(\mathcal{R}_{\mathcal{A}_t^c}) = true$, no more agents will be invited to join the committee.

Definition 6.4.2. *(Proactive Bounded Counsel Halting Decision Policy)*

$$D_H(\mathcal{R}_{A_i}) = \big(M_c(\mathcal{R}_{\mathcal{A}_t^c}) \geq \eta_1\big) \vee \big(max_{A_j \in \mathcal{A}_t^r}(M_{A_j}(\mathcal{R}_{\mathcal{A}_t^c})) < \eta_2\big)$$

where η_1 and η_2 are threshold parameters.

The D_H decision policy checks if one of two conditions is met: that the confidence in the solution predicted by the current committee is high enough or that there is a low confidence on the agents that have not yet joined the committee. If the confidence in the solution predicted by the current committee is high enough, i.e. $M_c(\mathcal{R}_{\mathcal{A}_t^c}) \geq \eta_1$, there is no need to invite more agents since the current solution has a very high likelihood to be the correct one. Moreover, if the confidence on an agent $A_j \in \mathcal{A}^r$ that is not in the committee is very low, i.e. $M_{A_j}(\mathcal{R}_{\mathcal{A}_t^c}) < \eta_2$, inviting A_j to join the committee is not advisable (since the prediction of that agent will very likely be incorrect and thus increases the chances of the committee prediction of being also incorrect). Therefore, if the maximum confidence of every agents in $\mathcal{A}_t^r$ is very low, i.e. $max_{A_j \in \mathcal{A}_t^r}(M_{A_j}(\mathcal{R}_{\mathcal{A}_t^c})) < \eta_2$, inviting any of these agents to join the committee is not advisable.

The two threshold parameters η_1 and η_2 have the following interpretation:

- η_1 represents the minimum confidence required for the candidate solution of the current voting situation, i.e. the convener agent will stop inviting agents when the confidence on the candidate solution of the current committee is above η_1.

- η_2 represents the minimum confidence required in the prediction of an individual agent to allow that agent to join the committee. Thus, those agents whose predictions have a confidence value lower than η_2 will not be invited to join the committee.

Notice that by varying η_1 and η_2, the behavior of PB-CCS can be changed. If we set a high value for η_1, the convener agent will tend to convene larger committees (since a high confidence is required to stop inviting agents), and if we set a low value for η_1, the convener agent will stop inviting agents earlier, since a lower confidence is required. Moreover, by setting a high value for η_2, the convener agent will be very selective with the agents allowed to join the committee, and only those agents with a confidence higher than η_2 will be allowed to join. On the other hand, a low value of η_2 will make the convener

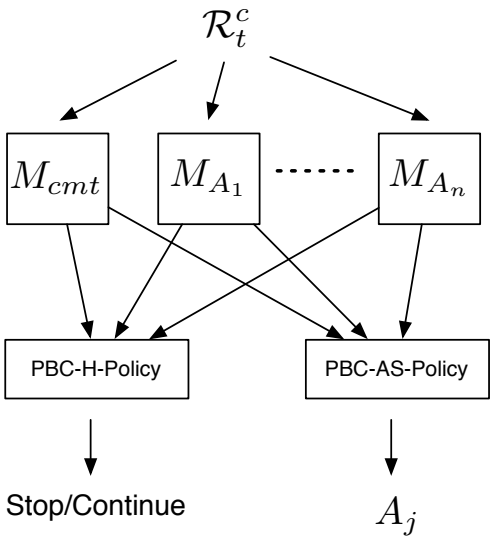


Figure 6.6: Relation among the competence models and the Proactive Bounded Counsel decision policies.

agent to be very permissive, and any agent can potentially be invited to join the committee.

In fact, if $\eta_1 = 0.0$, an agent will always solve problems individually, and if the parameters are set to $\eta_1 = 1.0$ and $\eta_2 = 0.0$ the resulting collaboration strategy will convene always all the available agents in the $\mathcal{M}$AC system, and therefore achieve the same results than the Committee Collaboration Strategy. Furthermore, by increasing η_2 (leaving $\eta_1 = 1.0$) we can obtain a collaboration strategy that invites all the agents to join the committee except those that have a confidence level lower than η_2. Therefore, η_1 and η_2 allow us to define a range of different strategies to build committees.

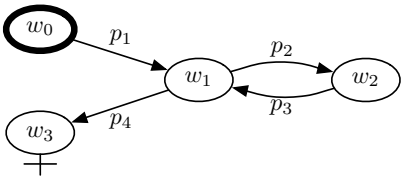
The second decision policy is the Proactive Bounded Counsel Agent Selection decision policy D_{AS}, that is defined as a function that takes as input a voting situation $\mathcal{R}_{A_i}$ and a set of candidate agents to be invited to the committee and returns the name of the agent that has the highest confidence on finding the correct solution for a given problem:

Definition 6.4.3. *(Proactive Bounded Counsel Agent Selection Decision Policy)*

$$D_{AS}(\mathcal{R}_{A_i}, \mathcal{A}_t^r) = argmax_{A \in \mathcal{A}_t^r}(M_A(\mathcal{R}_{\mathcal{A}_t^c}))$$

.

That is to say, D_{AS} selects to invite the agent $A_j \in \mathcal{A}_t^r$ that has the highest confidence $M_{A_j}(\mathcal{R}_{\mathcal{A}_t^c})$ on predicting the correct solution.



p_1 :	$Request(?User, ?A_i, ?P)$
p_2 :	$Request(!A_i, ?A_j, !P)$
p_3 :	$Inform(!A_j, !A_i, ?\mathbf{R})$
p_4 :	$Inform(!A_i, !User, ?S)$

Figure 6.7: Interaction protocol for the *Proactive Bounded Counsel* collaboration strategy.

Figure 6.6 shows the relations among the competence models and the decision policies in PB-CCS. The figure shows that in each voting situation where the policies have to be used, the confidence assessments given by the competence models are used by the decision policies. Moreover, notice that the competence models are used in each round of PB-CCS, since at each round there is a new agent in the committee and therefore the voting situation is different.

6.4.2 Proactive Bounded Counsel Protocol

Figure 6.7 shows the I_{PB-CCS} interaction protocol used in PB-CCS. Notice that I_{PB-CCS} is exactly the same as the protocol of the Bounded Counsel Collaboration Strategy (see Section 5.3).

Figure 6.7 shows that the states w_1 and w_2 of the protocol form the main loop of the protocol. Thus, PB-CCS is composed of a series of *rounds*. We will use the letter t to note the current round. The first round $t = 0$ starts the first time that the protocol reaches state w_1, and each time that the protocol changes from state w_2 to w_1 a new round $t + 1$ starts.

The protocol works as follows: the protocol is initiated when an agent receives a problem P from the user in message p_1. This agent will act as the convener agent A_c and the protocol moves to state w_1. The first time that the protocol reaches state w_1, the convener agent A_c solves the problem P individually. Then, the decision policy D_H is used to assess the confidence of the individually predicted solution. If D_H determines that the solution has not enough confidence, a new agent must be invited to join the committee. D_{AS} is used to select which agent $A_j \in \mathcal{A}_t^r$ will be invited, and A_c sends an invitation message p_2 to A_j. A_j individually solves problem P and replies to A_c with

message p_3 containing its individual prediction for problem P. At this point, A_c uses again the decision policy D_H to assess the confidence of the current committee prediction. If D_H determines that the candidate solution predicted by the current committee has enough confidence, message p_4 is sent to the user containing the final solution predicted by the committee, and the protocol ends.

Next section presents an experimental evaluation of PB-CCS, comparing it with CCS, P-CCS, and B-CCS.

6.5 Experimental Evaluation

This section presents the experimental evaluation of the performance of PB-CCS. To evaluate the behavior of the PB-CCS using the learnt competence models, we have compared the behavior of agents that use PB-CCS with agents that use the Committee Collaboration Strategy (CCS) and with agents that use the Bounded Counsel Collaboration Strategy (B-CCS). We have made experiments with $\mathcal{MAC}$ systems composed of 3, 5, 7, 9, 11, 13, and 15 agents. In these experiments, we have only used agents using 3-NN as learning method in the sponges data set. Moreover, in order to investigate whether the PB-CCS can adapt to different circumstances thanks to the proactive learning of competence models, we have performed experiments in three different scenarios: the *uniform* scenario, the *redundancy* scenario, and the *untruthful agents* scenario.

In the *uniform* scenario each individual agent receives a random sample of the data set with Committee Redundancy $\mathbb{R} \backsimeq 0.0$ and with Committee Bias $\mathbb{B} = 0.0$ (see Section 4.4). That is to say, in the uniform scenario the case bases of the individual agents are disjunct (i.e. there is no case shared by two agents' case bases), and the individual case bases have an average Case Base Bias of 0 (i.e. the ratio of cases of each class in the individual case bases is nearly the same than in the complete data set).

In the *redundancy* scenario each individual agent receives a random sample of the data set with redundancy $\mathbb{R} = 0.1$ and with Committee Bias $\mathbb{B} = 0.0$. That is to say, in the redundancy scenario, there is a number of cases that are present in more than one agents' case bases. For instance, with a Committee Redundancy $\mathbb{R} = 0.1$ in a $\mathcal{MAC}$ system composed of 5 agents working in the sponges domain each agent will have an average of 71.12 cases while with Committee Redundancy $\mathbb{R} = 0.0$ they will have only about 54.00 cases. The goal of performing experiments in the redundancy scenario is to test the behavior of PB-CCS in different areas of the *ensemble space* (see Section 4.4).

Finally, in the *untruthful agents* scenario some of the individual agents in the $\mathcal{MAC}$ system are untruthful. When an untruthful agent A_i joins a committee sometimes lies, i.e. it will vote for a solution different from the individual prediction A_i itself makes. Nevertheless, untruthful agents only lie when they are not the convener agents of a committee. In our experiments we have used 1, 2, 3, 4, 5, 6, and 7 untruthful agents for the 3, 5, 7, 9, 11, 13, and 15 agents $\mathcal{MAC}$ systems respectively. Untruthful agents lie 50% of the times in the experiments we report hereafter. Performing experiments in the untruthful agents scenario

has two goals: the first is to test the robustness of PB-CCS in the presence of malicious agents, and the second one is to verify that the learnt competence models are adequate in the sense that they are able to learn that there are some agents that should have a low confidence (the untruthful agents) and that those agents shouldn't be invited to join a committee.

The goal of performing experiments in these scenarios is to test whether the individually learnt competence models are useful to decide when to stop inviting agents to join the committee and which agents to invite under different conditions. The uniform scenario is the basic scenario, where each individual agent has a different sample of the training set. The redundancy scenario has been designed to test how PB-CCS performs in a different area of the ensemble space (where the performance of the individual agents is different and the ensemble effect changes) and to test whether it is possible to learn competence models adequate to work in different points of the ensemble space. Since each agent has more cases in the redundancy scenario than in the uniform scenario, it is expected that each individual agent has a greater individual accuracy. Therefore, we expect that the number of times an agent solves a problem individually without needing to convene a committee increases in the redundancy scenario. Moreover, the average number of agents needed to solve a problem should decrease for the same reason.

Finally, the untruthful agents scenario models a situation in which not all the agents of the system can be trusted. We have designed this scenario to test whether the learnt competence models can detect which agents in the system can be trusted and which cannot. In this scenario, we expect that the performance of the committee decreases with respect to the uniform scenario. Moreover, by using competence models, the proactive bounded counsel collaboration strategy should be able to detect untruthful agents and very seldom invite them to join the committee; consequently we expect the performance of PB-CCS not to decrease as much as the performance of CCS, thus showing a more robust behavior.

6.5.1 PB-CCS Evaluation in the Uniform Scenario

Figure 6.8 shows the classification accuracy and average committee size for agents using 3-NN as learning method to solve problems in the sponge data set. The left hand plot of Figure 6.8 shows the classification accuracy and the right hand plot shows the average committee size. $\mathcal{M}$AC systems with 1, 3, 5, 7, 9, 11, 13 and 15 agents are tested. For each $\mathcal{M}$AC system results for agents using CCS, B-CCS, and PB-CCS are presented. Moreover, two different parameter settings have been evaluated for PB-CCS: the first one with $\eta_1 = 0.9$ and $\eta_2 = 0.5$ and the second one with $\eta_1 = 0.95$ and $\eta_2 = 0.5$. In the first parameter settings the convener agent will request a confidence of at least 0.9 in order to stop inviting agents to join the committee, and in the second parameter settings, the convener agent will request a confidence of at least 0.95. Therefore, the expected behavior is that in the second parameter settings both the convened committees and the classification accuracy would be larger. Moreover, both

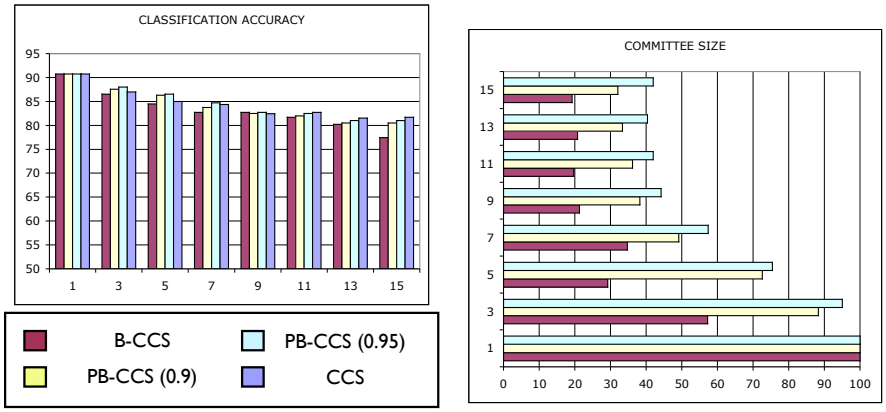


Figure 6.8: Classification accuracy and average committee size for agents using CCS, B-CCS, and PB-CCS in the sponges data set and using 3-NN in the uniform scenario.

parameter settings request that all invited agents have at least a confidence of 0.5 of predicting the correct solution for the current problem.

Figure 6.8 shows that the classification accuracy of PB-CCS is very close to that of CCS. In fact, with $\eta_1 = 0.95$ the difference in classification accuracy between PB-CCS and CCS is not statistically significant. Moreover, the classification accuracy of PB-CCS (both with $\eta_1 = 0.9$ and $\eta_1 = 0.95$) is higher than the classification accuracy of B-CCS in all of the $\mathcal{MAC}$ systems except in the 9 agents system (where the difference is not statistically significant).

The right hand plot of Figure 6.8 shows the average size of the committees convened by PB-CCS and B-CCS expressed as the percentage of the agents in the $\mathcal{MAC}$ system convened in average (we do not show the size of the committees convened by CCS that is 100% always since CCS invites all the agents to join the committee). The figure shows that the average size of the committees convened by PB-CCS is smaller than the committees convened by CCS and specially in $\mathcal{MAC}$ systems with a large number of agents. The figure also shows that the average size of the committees convened by PB-CCS is larger than in B-CCS. Furthermore, PB-CCS convenes larger committees than B-CCS, but we can conclude that PB-CCS invites more agents to join the committee when needed (since PB-CCS has a higher classification accuracy than B-CCS). Moreover, the threshold parameter η_1 affects the average size of the committee: if $\eta_1 = 0.95$ the size of the committees tends to be larger than with $\eta_1 = 0.9$, as expected.

Both the left and right hand plots of Figure 6.8 show that there is a tradeoff between committee size and classification accuracy. If η_1 takes lower values, the committee size is smaller and the classification accuracy approaches that of the individual agents; if η_1 takes large values, the committee size is larger and the classification accuracy is higher.

Figure 6.8 shows that PB-CCS achieves a better tradeoff of accuracy and

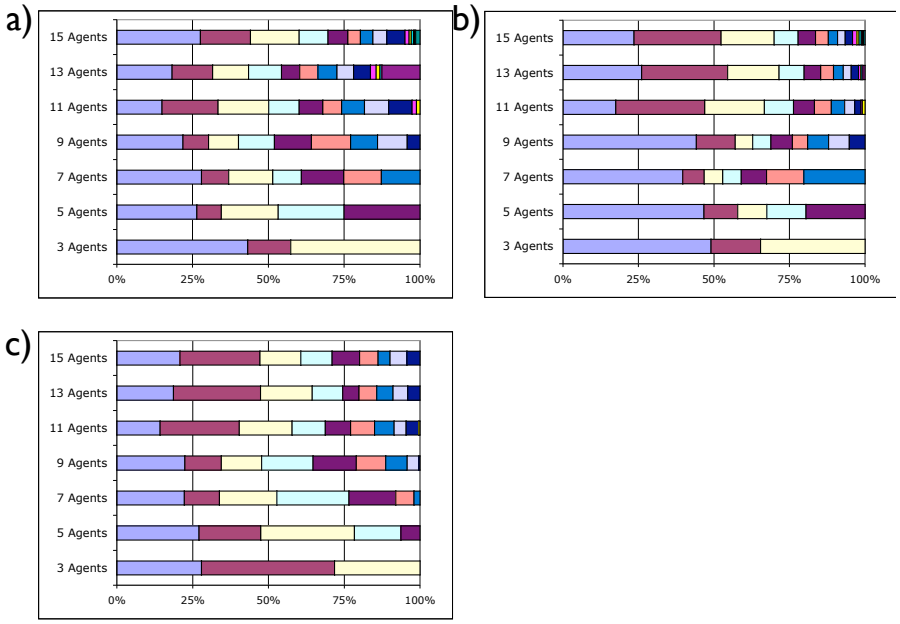


Figure 6.9: Percentage of times that the convener agent has convened committees of different sizes in the uniform scenario (a), the redundancy scenario (b) and in the untruthful agents scenario (c).

committee size than CCS since the classification accuracy achieved by PB-CCS with $\eta_1 = 0.95$ is undistinguishable of the accuracy of CCS while the average size of a committee convened by PB-CCS is much smaller than 100% (the size of a committee convened by CCS).

Figure 6.9.a shows the percentage of times that the convener agent has convened committees of different sizes with $\eta_1 = 0.9$. An horizontal bar is shown for each $\mathcal{M}$AC system (one for the 3 agents system, another for the 5 agent system, and so on). Each bar is divided in several intervals: the leftmost interval represents the percentage of times that the convener agent has solved the problem individually; the second interval represents the percentage of times that a committee of 2 agents has been convened, and so on. The right most interval represents the percentage of times that a committee containing all the agents in the system has been convened. Figure 6.9.a shows that in the 3 agents system, about 40% of the times the convener agent solves the problem individually without the need of convening a committee. However, this percentage is reduced in the $\mathcal{M}$AC systems with more agents; this is an expected result since in systems with more agents individual case bases are smaller and thus individual accuracy is lower (as we have seen in the experiments of Section 4.5.1); consequently, the Proactive Bounded Counsel Halting decision policy D_H decides more often to

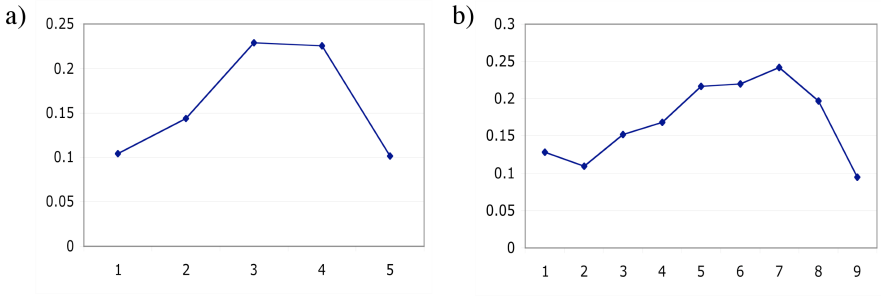


Figure 6.10: Distribution of the error among different convened committees for a 5 agents system (a) and a 9 agents system (b).

convene a committee. However, even for a 15 agents system, more than 25% percent of the times an agent can solve problems individually without compromising the overall $\mathcal{M}$AC performance. Moreover, for all the $\mathcal{M}$AC systems in the experiments, more than 30% of the problems are solved with at most 2 agents. This shows that even with a large number of agents (where each agent has a small case base) the decision policies are able to detect that there are problems that can be solved individually without reducing the classification accuracy.

In order to get a better insight of the behavior of PB-CCS Figure 6.10 shows the error distribution among different sizes of dynamic committees for two $\mathcal{M}$AC systems: a system composed of 5 agents and a system composed of 9 agents with $\eta_1 = 0.9$. The horizontal axis shows the different committee sizes: from 1 to 5 in the 5 agents system and from 1 to 9 in the 9 agents system. A committee size of 1 corresponds to the convener agent solving problems alone. The vertical axis shows the average error made when a committee of a given size has been convened. Both the left and right hand plots of Figure 6.10 show the same behavior: the agents make very few errors when solving problems alone and when all the agents have joined the committee; most of the errors are made when the convened committee contains most but not all of the agents in the committee. Notice that this is an expected behavior since the D_H decision policy only decides to solve problems with a committee of one agent (i.e. alone) when the confidence on the individually predicted solution is very high. Notice also that this expected behavior only takes place if the learnt competence models are adequate. Therefore, we can conclude that the learning process produces adequate competence models since they have the expected behavior. Moreover, a large number of agents is invited to join the committee for problems for which there is a low confidence on the predicted solution and it is in these problems (when the confidence on the predicted solution is low) where a large number of errors are expected. Finally, when the convener agent invites all the agents to join the committee, the classification accuracy is that of CCS.

Summarizing, PB-CCS in the uniform scenario can achieve a classification accuracy undistinguishable to that of CCS but convening smaller committees.

Moreover, the decision policies (using the learnt competence models) can effectively detect when the convener agent can solve the problem individually as Figure 6.10 shows (since the error when the committee has only one agent is very low). Consequently we can conclude that the proactive learning process is producing adequate competence models (since they exhibit the expected behavior). Moreover, we have seen that varying parameters η_1 and η_2 have the expected result in the behavior of PB-CCS. Specifically, we have seen that by setting $\eta_1 = 0.95$ the accuracy achieved is higher than setting $\eta_1 = 0.9$ as expected. The next section analyzes the behavior of PB-CCS in a different scenario.

6.5.2 PB-CCS Evaluation in the Redundancy Scenario

In the previous section we have seen that PB-CCS performs well in the uniform scenario. We are interested on performing experiments in the redundancy scenario to test the performance of PB-CCS in other areas of the *ensemble space* with different features than the uniform scenario. Specifically, in the redundancy scenario the case bases of the agents are not disjunct as in the uniform scenario, but have some overlapping, i.e. there are cases that are present in more than one agents' case base. This can potentially interfere in the proactive learning process, since if two agents have a large intersection between their case bases the competence models that they learn about each other could be overestimating their real confidence.

In our experiments we have used a degree of redundancy $\mathbb{R} = 0.1$ (See Section 4.4.2). Therefore, agents will have larger individual case bases than in the uniform scenario: for instance, using the sponges data set in a $\mathcal{M}$AC system composed of 5 agents, each agent will have a case base with an average of 50.8 cases if $\mathbb{R} = 0.0$ while for $\mathbb{R} = 0.1$ each agent will have a case base with an average of 71.12 cases. Moreover, we have used $\eta_1 = 0.9$ and $\eta_2 = 0.5$ for all the experiments in the redundancy scenario.

Figure 6.11 shows the classification accuracy and average committee size for agents using 3-NN as learning method to solve problems in the sponge data set. The left hand plot of Figure 6.11 shows the classification accuracy and the right hand plot shows the average committee size. $\mathcal{M}$AC systems with 1, 3, 5, 7, 9, 11, 13, and 15 agents are tested, and for each $\mathcal{M}$AC system results for agents using CCS, B-CCS, and PB-CCS are presented. The left hand plot in Figure 6.11 shows that the classification accuracy of PB-CCS, B-CCS, and CCS are very similar, and their accuracy values are higher than those achieved in the uniform scenario. In fact, the difference in classification accuracy is only statistically significant in the 11 and 13 agents systems where B-CCS achieves a lower classification accuracy than PB-CCS and CCS. Therefore, PB-CCS is as proficient as CCS.

In terms of committee size, PB-CCS convenes much smaller committees than the 100% committee of CCS as the right hand plot of Figure 6.11 shows. Again, this is specially noticeable in $\mathcal{M}$AC systems with a large number of agents. For instance, in a $\mathcal{M}$AC system with 13 agents, less than the 30% of the agents are convened in average, while CCS always convenes the 100% of the agents.

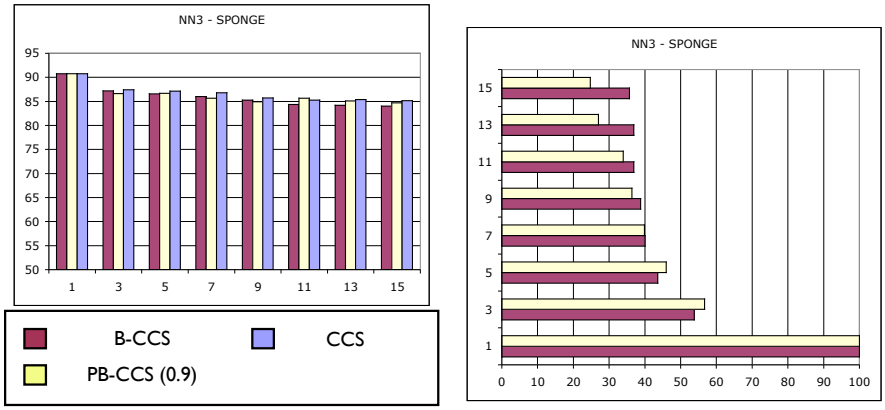


Figure 6.11: Classification accuracy and average committee size for agents using CCS, B-CCS, and PB-CCS in the sponges data set and using 3-NN in the redundancy scenario.

Moreover B-CCS also convenes larger committees in average than PB-CCS this time while having a lower accuracy as well.

Comparing the behavior of the three collaboration strategies in the redundancy scenario with their behavior in the uniform scenario, it would be desirable that the collaboration strategies convene smaller committees in the redundancy scenario since individual agents have higher classification accuracy; PB-CCS shows exactly this behavior, i.e. it convenes smaller committees in the redundancy scenario. However, B-CCS convenes larger committees in the redundancy scenario than in the uniform scenario. This happens because the competence models used by B-CCS are fixed, and do not change from one scenario to the other. Therefore those fixed competence models do not behave well in the redundancy scenario, where the agents have higher individual classification accuracy. This shows that learning competence models, as PB-CCS does, instead of using predefined ones, as B-CCS does, is a clear advantage. Moreover, another effect that we expect is that the classification accuracy of the collaboration strategies is higher in the redundancy scenario since the accuracy of the individual agents is higher. Comparing figures 6.8 and 6.11 we can observe that the three collaboration strategies show this behavior and their accuracy in the redundancy scenario is higher than in the uniform scenario.

Finally, Figure 6.9.b shows the percentage of times that the convener agent has convened committees of different sizes in the redundancy scenario. Figure 6.9.b shows that in the redundancy scenario, agents using PB-CCS solve problems individually more often than in the uniform scenario (shown in Figure 6.9.a). This shows that the proactive learning process has obtained good competence models, since the behavior of PB-CCS is the expected one, i.e. convene smaller committees in the redundancy scenario since since if the individual accuracy is higher, the agents will individually solve problems correctly more

often, and therefore, a committee has to be convened less often (and if there is the need to convene one, it can be convened with a smaller number of agents). Specifically, in $\mathcal{M}$AC systems composed of 9 agents or less, agents solve problems individually between a 40% and a 50% of the times and in systems with 11 agents or more, in the 50% of the times no more than 2 agents are invited to join the committee, while in the uniform scenario more agents were needed in average.

Concluding, from the experiments in the redundancy and uniform scenarios we can say that PB-CCS achieves a classification accuracy undistinguishable to that of CCS both in the uniform and redundancy scenario while B-CCS achieves lower accuracy values in some experiments. Moreover, PB-CCS convenes committees much smaller than those convened by CCS. Therefore, PB-CCS is better than CCS since PB-CCS achieves the same classification accuracy and convening a smaller number of agents. Moreover, PB-CCS is also better than B-CCS for two reasons: first of all, PB-CCS achieves higher classification accuracy than B-CCS, and second PB-CCS uses learnt competence models that can be more adequate to the specific $\mathcal{M}$AC system in which the convener agent is working.

6.5.3 PB-CCS Evaluation in the Untruthful Agents Scenario

The untruthful agents scenario has two goals: the first one is to evaluate the robustness of PB-CCS to the presence of malicious agents (that is equivalent to evaluate the robustness of PB-CCS to noise) i.e. that the performance of PB-CCS is independent (or is "more" independent than CCS) with respect to the *perfect cooperation* assumption (i.e. that all the agent can be trusted and will do their best to cooperate); the second goal is to evaluate whether the proactive learning process produces adequate competence models, i.e. competence models that can detect that there are some agents that have a very low confidence (the untruthful agents) and that shouldn't be invited to join the committee.

Specifically, we have prepared a scenario where some agents in the $\mathcal{M}$AC system will lie in their predictions when forming part of a committee. These *untruthful* agents will tell their individually predicted solution truthfully when they are convener agents, but will sometimes lie when they are not the convener agents. In our experiments we have set a probability of the 50% that an untruthful agent lies about its individual prediction. Specifically, there will be 1, 2, 3, 4, 5, 6 and 7 untruthful agents in the 3, 5, 7, 9, 11, 13 and 15 agents systems respectively. Moreover, in this scenario we expect that the Proactive Bounded Counsel Agent Selection decision policy, D_{AS}, is able to effectively decide which agents have a high confidence and which ones have a low confidence, so that untruthful agents are very seldom invited to join a committee.

In the presence of the untruthful agents, it is expected that the classification accuracy of all the collaboration strategies is lower than in the uniform or redundancy scenarios since there are less agents with high confidence in the system that can be invited to join the committee. Figure 6.12 shows the classification accuracy and average committee size for agents using 3-NN as learning method

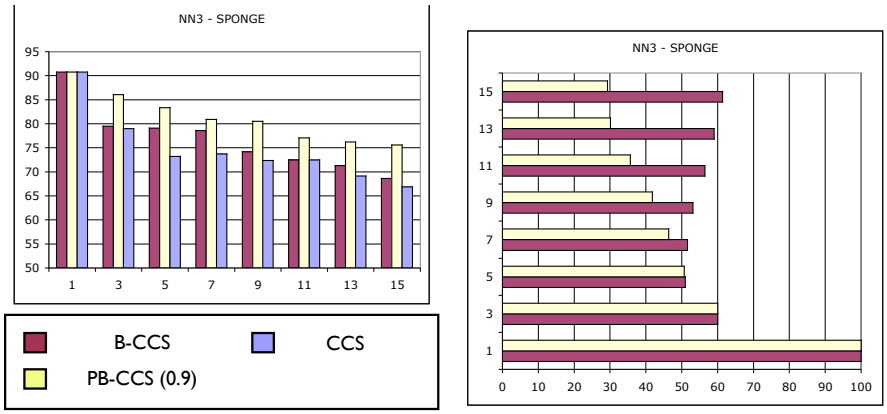


Figure 6.12: Classification accuracy and average committee size for agents using CCS, B-CCS, and PB-CCS in the sponges data set and using 3-NN in the redundancy scenario.

to solve problems in the sponge data set. The threshold parameters are $\eta_1 = 0.9$ and $\eta_2 = 0.5$. The figure shows that in this scenario the classification accuracy achieved by CCS and B-CCS is lower than the accuracy achieved by PB-CCS (in fact, the accuracy of CCS is even lower than the accuracy achieved by B-CCS). Moreover, comparing the accuracy achieved by the three collaboration strategies in the untruthful agents scenario with that achieved in the uniform scenario (shown in Figure 6.8) we see that they all achieve lower classification accuracy in the untruthful agents scenario (as expected). This reduction in classification accuracy is specially severe in CCS. For instance, the classification accuracy of CCS drops from 81.71% to 66.80% in the 15 agents scenario. Thus, we can conclude that CCS is not roust when there are agents that cannot be trusted. The accuracy of B-CCS also drops severely, but the classification of B-CCS in the untruthful agents scenario is slightly higher than that of CCS. Finally, Figure 6.12 shows that the classification accuracy of PB-CCS drops much less than B-CCS and CCS. For instance, in the 15 agents scenario, the classification accuracy drops from 81.00% to 75.62%: PB-CCS drops 4.38 points in the 15 agents system, while CCS drops 14.91 points and B-CCS drops 9.23 points. This shows that PB-CCS is much more robust in the presence of untruthful agents than CCS and B-CCS.

Concerning the committee sizes, Figure 6.12 shows on the right hand side that the average committee size convened by PB-CCS is clearly smaller than the size of the committees convened by CCS(100%), specially in systems with many agents. Moreover, in this scenario, the average size of the committees convened by PB-CCS is also smaller than those convened by B-CCS. As the number of agents increase, the difference in size of the committees convened by B-CCS and PB-CCS increases. The explanation is that B-CCS uses a random decision policy to determine which agents are invited to join the committee, and therefore,

Agents	3	5	7	9	11	13	15
Standard	47.57%	44.14%	43.63%	31.0%	32.0%	32.75%	31.8%
Untruthful	5.07%	9.03%	6.82%	7.14%	9.14%	11.57%	11.08%

Table 6.2: Average number of times that standard and untruthful agents are invited to join a committee.

untruthful agents are regularly invited to the committee. An untruthful agent that joins a committee will not likely contribute to increase the confidence of the predicted solution, and more agents will need to be invited, thus increasing the average committee size. In contrast, PB-CCS uses learnt competence models in the D_{AS} decision policy to select which of the other agents is the best one to be invited to join the committee. Results concerning classification accuracy and average committee size in Figure 6.12 prove that this decision policy is useful and that effectively helps to convene a better committee than those convened using B-CCS or CCS. Consequently, this proves that the proactive learning process produces adequate competence models since the decision policy that uses them behaves as we would expect.

Figure 6.9.c shows the percentage of times that the convener agent has convened committees of different sizes in the untruthful agents scenario. Specifically, we see that agents using PB-CCS in the untruthful agents scenario tend to convene smaller committees than in the uniform scenario (Figure 6.9.a). Notice that in the redundancy scenario smaller committees were convened due to an increased individual classification accuracy, and therefore agents solved problems individually more often. This is not the case of the untruthful agents scenario, since the committees convened in the untruthful agents scenario are smaller not because the individual agents have higher classification accuracy but because there are less agents with a high confidence that can be invited to join the committee. In fact, agents in the untruthful agents scenario should solve problems individually (without convening a committee) with the same frequency than agents in the uniform scenario, but convene smaller committees because there is a subset of agents (the untruthful agent, that are almost half of the agents in our experiments) that will be detected as having low confidence and will very seldom be invited to join the committee. For instance, in the 15 agents scenario, never more than 10 agents are convened in a committee. Moreover, Figure 6.9 shows that agents in the untruthful agents scenario solve problems individually with more or less the same probability than in the uniform scenario except in the 3 agents system. Therefore, in the untruthful agents scenario agents solve problems individually with about the same probability than in the uniform scenario, but they tend to stop inviting agents earlier (this is the cause of the small accuracy reduction of PB-CCS from the uniform scenario to the untruthful agents scenario).

For the purpose of assessing the degree in which the Proactive Bounded Agent Selection decision policy D_{AS} is able to detect the untruthful agents, the number of times that each agent has been invited to join a committee has been

counted, and Table 6.2 summarizes these results. For each $\mathcal{M}$AC system, two values are shown: the average number of times that a standard agent has been convened to a committee and the average number of times that an untruthful agent has been convened to a committee. For instance, in the 3 agents $\mathcal{M}$AC system, each one of the two standard agents is invited to join a committee a 47.57% of the times that a problem is solved while the only untruthful agent is only invited to join a committee a 5.07% of the times. This clearly shows that D_{AS} selects a truthful agent in most of the cases. In fact, the degree to which D_{AS} is able to detect the untruthful agents depends of the threshold parameter η_2. In these experiments we have set $\eta_2 = 0.5$, but if we set a higher value (e.g. $\eta_2 = 0.6$) untruthful agents will be invited even less often. Notice that the decision of choosing $\eta_2 = 0.5$ is made in order to match one the preconditions of the ensemble effect, namely that the individual error of the classifiers must be lower than 0.5 (see Section 2.1).

Moreover, another effect we can observe in Table 6.2 is that untruthful agents are invited more often in the systems with many agents than in systems with a smaller number of agents. There are two main reasons of this: the first one is that in systems with many agents the individual case bases are small and therefore, an agent has a small amount of cases that can be used for collecting M-examples, leading to less accurate competence models. However, even with the small amount of cases of the 15 agents system (where each agent has an average of only 16.9 cases), untruthful agents are invited less often than truthful agents. The second reason is that in $\mathcal{M}$AC systems with many agents, where each individual agent has a very low classification accuracy and where the estimated confidence of the current committee is very low, having an agent that may help you 50% of the times (such as the untruthful agents) is not that bad. Therefore, in systems where the individual agents have a very low classification accuracy, an agent that lies the 50% of the times may sometimes (when there are no other agents that can be invited) help to increase the confidence of the current committee.

The conclusion that we can draw form the experiments in the untruthful agents scenario is that PB-CCS is more robust than CCS and that B-CCS when the perfect collaboration assumption does not hold, i.e. when not all the agents can be trusted. The result is that PB-CCS achieves a higher classification accuracy than both CCS and B-CCS and also convening smaller committees.

6.6 Conclusions

In this chapter we have presented a proactive learning technique with which agents in a $\mathcal{M}$AC system can learn their own competence models. We have also presented the Proactive Bounded Counsel Collaboration Strategy (PB-CCS), a dynamic committee collaboration strategy that allows an agent to use the learnt competence models to decide when to invite more agents to join a committee and which agents to invite. Basically, PB-CCS uses learnt competence models (instead of predefined ones) and uses individual decision policies that are able

to use all the information that the learnt competence models can provide.

In order to evaluate the performance of PB-CCS we have compared it against CCS and B-CCS in three different scenarios: the uniform scenario, the redundancy scenario and the untruthful agents scenario. The results of the experiments show that PB-CCS achieves always accuracy values that are undistinguishable or higher than those of CCS. Specifically, PB-CCS achieves undistinguishable accuracy than CCS in the uniform and redundancy scenarios and higher accuracy values in the untruthful agents scenario. Moreover, the experimental results also show that PB-CCS always convenes smaller committees than those convened by CCS and that the agent using PB-CCS solves problems individually without needing any other agent in the $\mathcal{MAC}$ system for a considerable percentage of the problems.

PB-CCS also has been compared with respect to B-CCS in all the three scenarios. The results show that PB-CCS achieves higher classification accuracy values than B-CCS in all three scenarios. About the size of the committees convened, B-CCS convened smaller committees in the uniform scenario and PB-CCS convened smaller committees in the other two scenarios. The reason is that the fixed competence models of B-CCS are not adequate for the redundancy or the untruthful agents scenario.

We have seen that although the PB-CCS decision policies have two parameters (η_1 and η_2), their meaning is clear (η_1 is the required confidence in the solution of the committee and η_2 is the required confidence of an individual agent to be invited to join a committee). Moreover, these parameters allow us to define a range of different desired behaviors. For instance, $\eta_1 = 0$ makes the convener agent to work always individually; $\eta_1 = 1$ and $\eta_2 = 0$ prescribes the exact behavior of CCS, $\eta_1 = 1$ and $\eta_2 = 0.5$ defines a committee strategy that convenes only those agents that have a confidence higher than 0.5 (i.e., convenes a committee with *all* the minimally competent agents in the system). Therefore, PB-CCS gives us freedom to define a range of behaviors we want a multi-agent system to have.

The conclusions that we can draw from the previous results are the following ones:

- First of all, PB-CCS has shown to be more robust than CCS and B-CCS since it can achieve high classification accuracy values in a wider range of scenarios than CCS and B-CCS. Specifically, PB-CCS has shown to be robust to the presence of untruthful agents (that is equivalent to the presence of noise in the system) while CCS and B-CCS experiment a huge reduction of classification accuracy in this scenario.

- Second, PB-CCS convenes in average smaller committees than CCS while achieving the same levels of accuracy. As a consequence, we can say that PB-CCS correctly decides when to stop inviting agents, since the classification accuracy is not affected by the reduced number of agents. Moreover, since the convener agent solves the problem individually a significant number of times, we can also conclude that PB-CCS correctly decides when an

agent can solve problems individually without compromising classification accuracy.

- Third, PB-CCS behaves as expected in all of the three scenarios, i.e. it achieves a high classification accuracy, convenes smaller committees in the redundancy scenario than in the uniform scenario and very seldom convenes untruthful agents to join the committee. In fact, PB-CCS should perform well given that the following conditions are met: a) the agents have a minimum amount of cases (needed to collect M-examples), b) the agents do no change their behavior radically (otherwise the competence models will not be useful), and c) there are at least some minimally competent agents in the system (if all the agents in the system have a very low confidence, no collaboration strategy can perform well since the ensemble effect would fail to take place).

- Fourth, since PB-CCS has learnt adequate competence models in all three scenarios we can conclude that the proactive learning process is successful, in the sense that the learnt competence models are good predictors of the confidence of the committee and of the individual agents. The experimental section provides evidence for this fact:

  1. PB-CCS can adequately decide when the convener agent can solve a problem individually (and this cannot be done unless the competence models are adequate);
  2. PB-CCS convenes less agents in the redundancy scenario than in the uniform scenario (thus, the competence models are able to exploit the fact that the individual agents have a higher classification accuracy in the redundancy scenario and therefore less agents are needed in a committee to achieve a comparably high accuracy);
  3. PB-CCS is able to reduce the number of times that an untruthful agent is convened into the committee (and this would have been impossible unless the competence models of the individual agents capture which agents have a low confidence and which a high confidence);
  4. finally PB-CCS as shown to achieve the same classification accuracy (or higher) than CCS with a reduced number of agents (thus the competence model of the committee effectively models the confidence of the committee and helps PB-CCS to decide when to stop inviting agents).

Finally, we can compare the proactive learning technique with the meta learning approach [19] in Machine Learning. Some of the competence models predict the confidence of an individual agent to predict the correct solution. These competence models can be compared with meta learning, an approach that builds building models of the individual classifiers in an ensemble in order to select which is the best classifier to send a problem to. However, the meta

learning approach is a centralized one, where a set of individual classifiers have been built, and a centralized algorithm computes a competence model of each classifier. In our framework there is no such a centralized algorithm, and each agent is responsible of building its own competence models. Moreover, in the meta learning approach, each problem is solved only by one classifier and the models are used to decide which classifier is going to solve the problem. In our approach we do not force a single agent to solve each problem, but the agents use the competence models to build committees whose solution has a high confidence.

Chapter 7

Justification Endorsed Collaboration

This chapter presents the *Justification Endorsed Collaboration* (JEC). Specifically, this chapter explores how the ability of the agents to provide *justifications* (Section 3.5.2) of their predictions can be exploited for different purposes. For instance, we will present the *Justification Endorsed Committee Collaboration Strategy* (JE-CS) in which agents members of a committee will use justifications of individual predictions in order to improve the voting system used to reach an aggregated prediction.

7.1 Introduction

Chapter 4 presented a collaboration strategy called the Committee Collaboration Strategy (CCS) that allows a group of agents solve a problem in a collaborative way and achieve higher classification accuracy than when the problem is solved individually by an agent in the system. Using the Committee Collaboration Strategy the individual predictions made by each individual agents are aggregated by means of a voting system. The voting system presented in Chapter 4 assumes that each agent in the committee is equally important, i.e. every agent has the same weight in the voting. However, this may not always be the best way to aggregate the individual predictions, since not all agents are equally good solving all the different kinds of problems. Therefore, depending on the specific problem, some agents are more likely to predict the correct solution than others. Ideally, those agents should have a higher influence in the voting.

This chapter presents a new kind of collaboration called *Justification Endorsed Collaboration* (JEC), that tries to exploit the ability of the individual agents to provide justifications (Section 3.5.2) for their individual predictions. In JEC, when individual agents are requested to solve a problem, they provide a *justified prediction* (that consists of the individual prediction of the agent plus

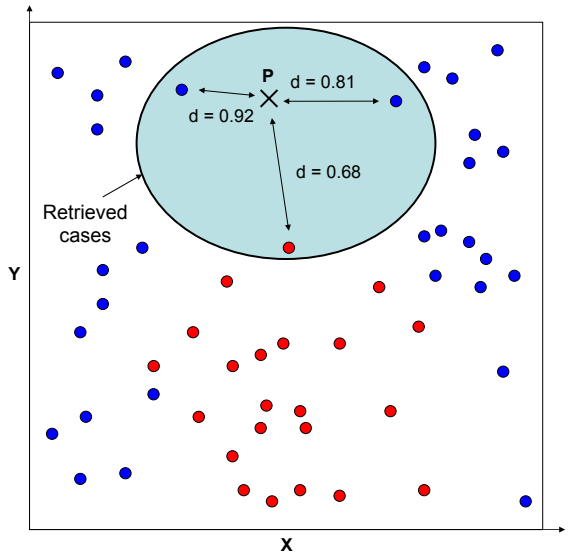


Figure 7.1: Illustration of the retrieval process using a numerical similarity measure for a two dimensional problem space.

the justification of the prediction) instead of a simple prediction. Specifically, we will present the *Justification Endorsed Committee Collaboration Strategy* (JE-CS), a specific collaboration strategy that allows the agents to determine which agents should have a higher influence in the voting system by analyzing the justified predictions provided by each agent. In this way, each individual justified prediction can be weighted properly in the voting system assigning lower weights to the weakly-assessed justified predictions and higher weights to the strongly-assessed justified predictions.

The chapter is structured as follows. Section 7.2 presents and analyzes the notion of justifications. After that, Section 7.3 presents the JE-CS collaboration strategy. Then, sections 7.4 and 7.5 explain the two decision policies needed in JE-CS, namely the *Justification Examination* and the *Justification Endorsed Voting* decision policies. After that, JE-CS is illustrated with an exemplification. Section 7.7 empirically evaluates JE-CS in three scenarios. The chapter ends with a conclusions section.

7.2 Justifications in CBR Systems

In this section we are going to refine the notion of *justifications* presented in Section 3.5.2, relating it to the notion of *symbolic similarity*. Therefore, symbolic similarity is first presented and then justifications are formally defined.

Case	Traffic_light	Cars_crossing	**Action**
c1	Green	Yes	**Wait**
c2	Green	No	**Cross**
c3	Red	Yes	**Wait**
c4	Red	No	**Wait**

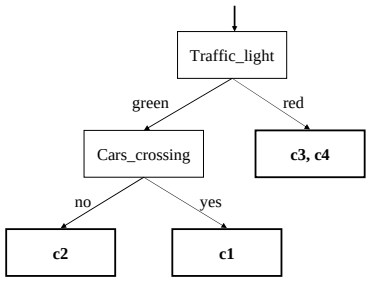


Figure 7.2: Case base and decision tree built to index the cases for a simple problem.

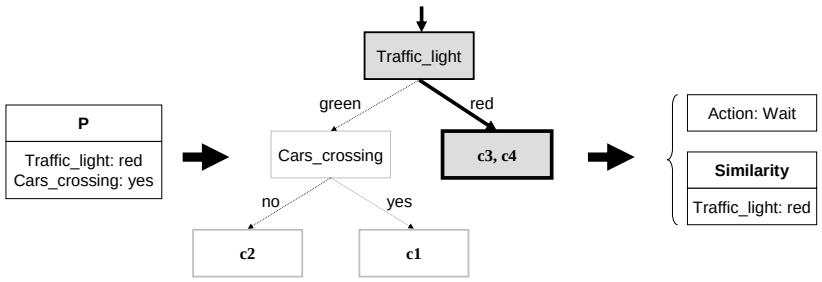


Figure 7.3: The problem P is solved using the cases in the corresponding leaf of the tree. The solution and the symbolic similarity built are shown.

7.2.1 Symbolic Similarity Measures

Case Based Reasoning systems use similarity measures to retrieve cases from the case base to solve a problem. Typically these similarity measures are numerical, i.e. the system computes the similarity between the current problem and the cases in the case base and the resulting values are real numbers (in the interval $[0, 1]$ if the similarity is normalized); then, the case (or cases) that have obtained the highest similarity value(s) are retrieved following the specific criteria defined by the CBR system. An example of a retrieval process using numerical similarity is illustrated in Figure 7.1, where the three most similar cases are selected for retrieval. However, some CBR systems can use *symbolic similarity* measures.

Definition 7.2.1. *A* symbolic similarity *among two or more cases $c_1, ..., c_n$ is a set of conditions J that all the cases $c_1, ..., c_n$ satisfy.*

Let us illustrate this with an example. Figure 7.2 shows a case base for a toy problem with two symbolic features: *Traffic_light* and *Cars_crossing*. There are two possible solution classes: *Wait* or *Cross*. In order to index the four cases in the case base some CBR systems use a decision tree, as shown in Figure 7.2. Imagine now that the problem $P = \langle Traffic\_light = red, Cars\_crossing =$

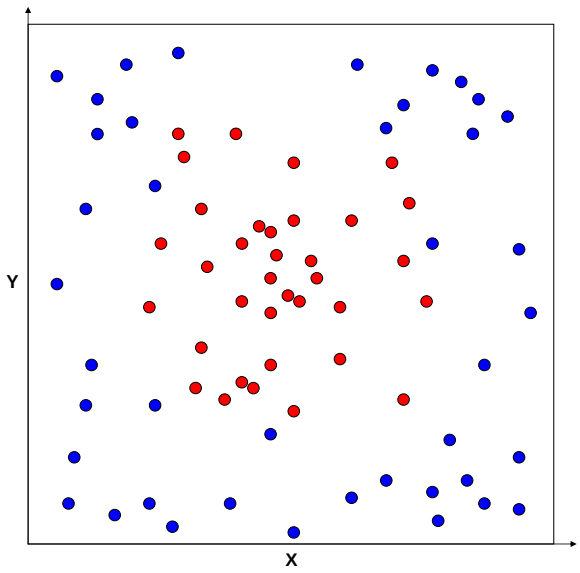


Figure 7.4: Simple problem with two numerical attributes and two classes.

$yes\rangle$ needs to be solved. The decision tree is used to retrieve cases by classifying the problem in one of the leafs of the decision tree as shown in Figure 7.3. All the cases in that leaf are the retrieved cases (in this example $c3$ and $c4$). The symbolic similarity among the problem and these two cases is built by using the branch of the decision tree that has classified the problem: $J = \{Traffic\_light = red\}$. Notice that not all the features in the problem are contained in the symbolic similarity. Using a numerical similarity measure, the case $c3$ would be the most similar case with a similarity value of 1.0 (since it is identical to the problem). However, using a symbolic similarity measure, the CBR system (by using the decision tree) has determined that if the feature *Traffic_light* has value *red*, the value of the feature *Cars_crossing* is irrelevant. Therefore, since *Cars_crossing* does not help on finding the correct solution for the problem, it does not appear in the symbolic similarity.

Moreover, symbolic similarities are not only applicable to problems with symbolic features. Problems with numerical attributes can also be solved by CBR methods using symbolic similarity measures. Let us illustrate this with an example where a decision tree is used to generate a symbolic similarity measure. Figure 7.4 shows a classification problem where there are two classes and where instances have just two numeric attributes, X and Y. Using the instances shown in Figure 7.4 the decision tree shown in Figure 7.5 is learnt. Notice that the decision tree is used to index the cases in the case base, i.e. each leaf contains a list of cases.

The decision tree in Figure 7.5 induces the partition over the problem space shown in Figure 7.6. Now imagine that we want to solve the problem $P =$

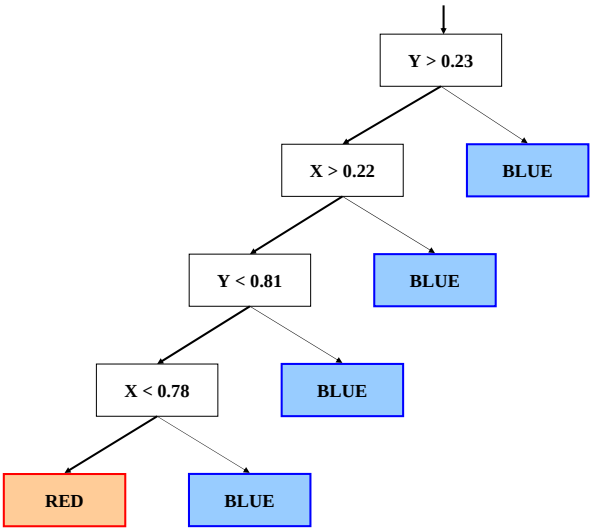


Figure 7.5: Decision tree learnt for the problem shown in Figure 7.4.

$\langle X : 0.44, Y : 0.83 \rangle$. In order to solve the problem P the decision tree is used to retrieve cases from the case base. First, the problem P is classified into a leaf of the tree as shown in Figure 7.7. All the cases in that leaf are the set of retrieved cases, and the symbolic similarity measure among the problem and the retrieved cases is built by taking the branch of the tree used to solve P in Figure 7.7. In this example, the symbolic similarity contains the conditions that appear in the nodes of the tree: $J = \{Y > 0.23, X > 0.22, Y > 0.81\}$. However, since $Y > 0.81$ is more specific than $Y > 0.23$, the symbolic similarity measure is simply $J = \{X > 0.22, Y > 0.81\}$ as shown in Figure 7.7. Notice that the similarity description contains conditions and not the concrete values of the problem: P has not been classified into the *blue* class because the value of its features are specifically $X = 0.44$ and $Y = 0.83$ but because those values satisfy the conditions in the nodes of the branch of the tree used to solve the problem. Therefore, the similarity description contains the conditions satisfied by the problem and the retrieved cases, and not the concrete values that the problem has in its features. Figure 7.8 shows graphically the set of cases retrieved for problem P.

7.2.2 Symbolic Local Approximation and Symbolic Similarity

Learning methods can be classified in two groups: eager methods and lazy methods. Eager learning methods have the property of building a global approximation (or model) of the target function; in contrast, lazy learning methods build local approximations of the target function during problem solving time

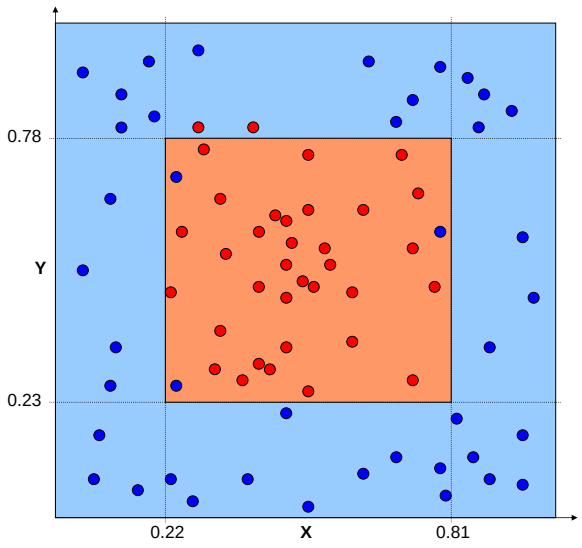


Figure 7.6: Partition induced over the problem space by the decision tree shown in Figure 7.5.

centered on the problem at hand.

A *symbolic local approximation* built by a lazy learning method is a generalization of the cases located in the neighborhood of the problem that we want to solve (i.e. the retrieved, most similar cases) and of the problem at hand. Moreover, such local approximation is a description of the information that the retrieved cases and the problem have in common (since it is a generalization of them).

The symbolic description of the similarity (*symbolic similarity*) contains information that is shared among the problem P and the retrieved cases. Moreover, it only contains the shared information that has been considered relevant by the CBR system. Therefore, the CBR system concludes that the problem P should have a solution equal to that of the retrieved cases (since they share relevant information with P). The symbolic similarity contains the information considered relevant by the CBR system to classify P into a specific solution class. Therefore, the symbolic similarity can be seen as the *justification* of why a problem P has been classified into a specific solution class, as required by the Definition 3.5.2.

Intuitively, when an agent makes a justified prediction saying that a problem P belongs to a solution class S_k and gives a justification J, the meaning is "P belongs to the class S_k, and the reason is that P satisfies J and my case base has n cases that also satisfy J and belong to the class S_k".

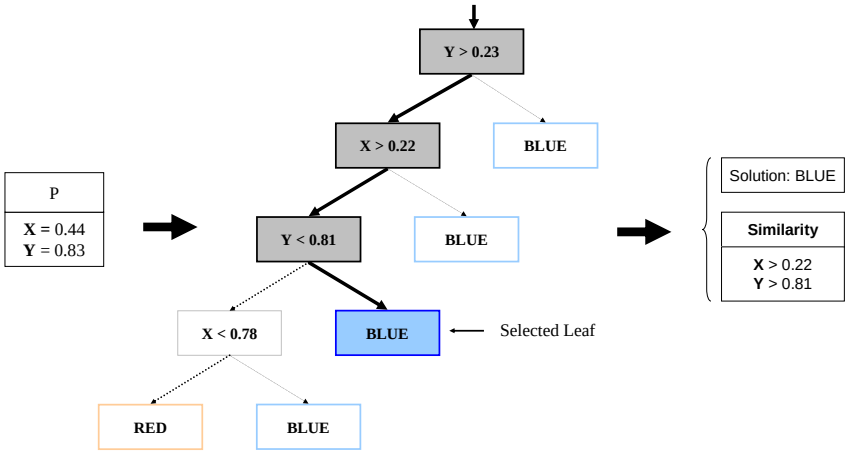


Figure 7.7: The problem P is solved using the cases in the corresponding leaf of the tree. The solution and the symbolic similarity built are shown.

7.2.3 Counterexamples and Endorsing Cases

Since a symbolic similarity is a term that generalizes the retrieved cases and the problem at hand, a justification (that is a local approximation) subsumes both the problem and the retrieved cases (since it is a generalization of them). This is a very important property of justifications and will allow us to define the concept of *counter examples* and of *endorsing cases*.

To define counter examples and endorsing cases it is important to remember that when a CBR agent A_i is requested to solve a problem P providing a justification (i.e. it is requested to provide a *justified prediction*), a *justification endorsement record* (JER) (Definition 3.5.3) $\mathbf{J} = \langle S_k, J, P, A_i \rangle$ is built. A JER contains the predicted solution class $\mathbf{J}.S$, the justification $\mathbf{J}.J$, the problem solved $\mathbf{J}.P$ and the agent who has solved the problem $\mathbf{J}.A$. Moreover, a JER $\mathbf{J} = \langle S_k, J, P, A_i \rangle$ induces a partition of any set of cases C in two sets:

- The cases *subsumed* by the justification $\mathbf{J}.J$, i.e. $\{c \in C | \mathbf{J}.J \sqsubseteq c.P\}$.
- The rest of the cases, i.e. $\{c \in C | \mathbf{J}.J \not\sqsubseteq c.P\}$.

The set of subsumed cases by the justification $\mathbf{J}.J$ are those cases that are similar to the problem $\mathbf{J}.P$ since the justification $\mathbf{J}.J$ is a symbolic similarity description. Moreover, we can further divide the set of subsumed cases into two subsets of cases: *counterexamples* and *endorsing cases*.

Definition 7.2.2. *The* set of counterexamples *of a JER* $\mathbf{J}$ *from a case base* C *is* $CE(\mathbf{J}, C) = \{c \in C | \mathbf{J}.J \sqsubseteq c.P \wedge c.S \neq \mathbf{J}.S\}$, *i.e. the set of cases* $c \in C$ *subsumed by the justification* $\mathbf{J}.J$ *that do not belong to the predicted solution class* $\mathbf{J}.S$.

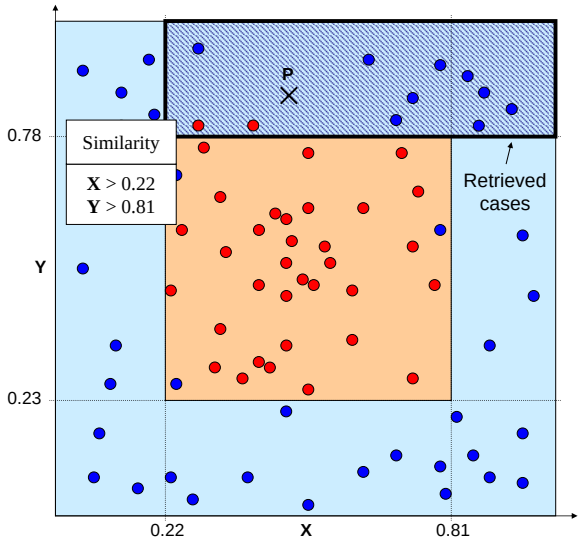


Figure 7.8: Set of retrieved cases by the similarity description presented in Figure 7.7.

Definition 7.2.3. *The* set of endorsing cases *of a JER* $\mathbf{J}$ *from a case base* C *is* $EC(\mathbf{J}, C) = \{c \in C | \mathbf{J}.J \sqsubseteq c.P \wedge c.S = \mathbf{J}.S\}$, *i.e. the set of cases* $c \in C$ *subsumed by the justification* $\mathbf{J}.J$ *that do belong to the predicted solution class* $\mathbf{J}.S$.

The set of endorsing cases are those subsumed by the justification (and therefore are those that are considered to be similar enough to the problem $\mathbf{J}.P$ to be retrieved) that belong to the predicted solution class $\mathbf{J}.S$, i.e. those cases endorse that $\mathbf{J}.S$ is the correct solution for the problem because they share the same solution. Moreover, the counterexamples are those cases that are considered similar to the problem $\mathbf{J}.P$ because are subsumed by the justification $\mathbf{J}.J$, but that do not endorse the solution $\mathbf{J}.S$ as the correct one (since they belong to a different solution class). In summary, a counterexample is a case that is not consistent with an agent's justified prediction and an endorsing case is a case that is consistent with an agent justified prediction.

Figure 7.9 illustrates the concepts of counterexamples and endorsing cases. The figure shows a problem space consisting of two numerical features. An agent A_i has learnt the decision tree shown in Figure 7.5. When the problem $P = \langle X : 0.44, Y : 0.83 \rangle$ is solved by A_i using that decision tree to retrieve cases from the case base, the solution found is *blue* and the justification provided is $J_1 = \{X > 0.22, Y > 0.81\}$. Therefore, the JER built by the agent A_i is $\mathbf{J} = \langle blue, J_1, P, A_i \rangle$. Figure 7.9 shows cases subsumed by the justification J_1 as a dashed area, in which we can find a set of counterexamples and a set of endorsing cases: two cases (shown in red) are counterexamples of problem P

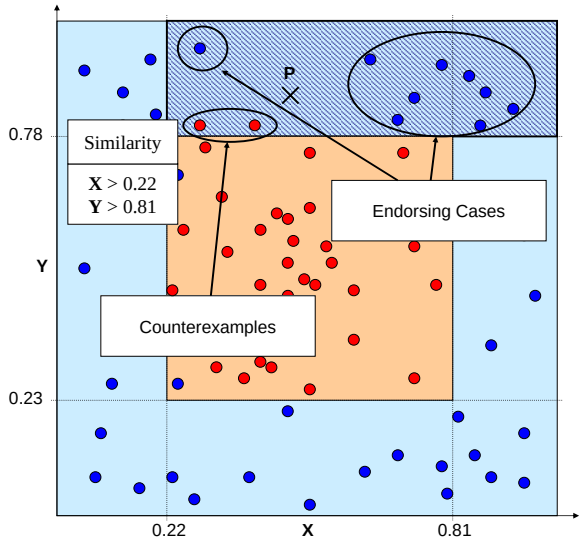


Figure 7.9: Counterexamples and endorsing cases for a given justification.

because they are subsumed by J_1, but they belong to the *red* class instead to the predicted one (*blue*); the other 9 cases (shown in blue) are endorsing cases since they endorse the justified prediction made by A_i (that the problem belongs to the *blue* solution class).

The next section presents the *Justification Endorsed Committee Collaboration Strategy*, that uses justifications in order to improve the performance of a committee of agents in terms of classification accuracy.

7.3 Justification Endorsed Committee Collaboration Strategy

This section presents the *Justification Endorsed Committee Collaboration Strategy* (JE-CS), that allows a committee of agents $\mathcal{A}^c$ to improve the classification accuracy (as we will show in the experiments section). JE-CS benefits from the ability of the individual agents to provide justifications of their individual predictions. The basic idea of JE-CS is that every agent in the committee provides a *justified prediction* of the problem at hand, that contains the predictions plus a justification of the predictions. Those justified predictions are distributed among the rest of agents in the committee in the form of Justification Endorsement Records (JERs). Then, each agent individually examines each justified prediction and those examinations are used to compute an *overall confidence estimation* of each justified prediction. Finally, the justified predictions are aggregated by the convener agent A_c by means of a weighted voting system that

uses the overall confidence estimations as weights for the votes.

Definition 7.3.1. *The* Justification Endorsed Collaboration Collaboration Strategy *(JE-CS) is a collaboration strategy* $\langle I_{JE-CS}, D_{JE}, D_{JEV} \rangle$*, where* I_{JE-CS} *is the JE-CS interaction protocol shown in Figure 7.12.* D_{JE} *is the* Justification Examination *decision policy used to compute the confidence values of the Justification Endorsement Records built by the agents, and* D_{JEV} *is the* Justification Endorsed Voting *used to aggregate the individual predictions using the confidence values computed with* D_{JE}*.*

The next sections present the two decision policies used in JE-CS (D_{JE} and D_{JEV}) and the interaction protocol (I_{JE-CS}).

7.4 Examination of Justifications

This section presents the D_{JE} policy used by the individual agents to assess the confidence of an agent concerning a justified prediction made by another agent.

Justification examination always involves two CBR agents: the *justifier*, the agent that justifies a prediction, and the *examiner*, the agent that examines that justification. The process of examination is basically performed in the following way: the examiner tests whether the justification is consistent with its case base and as a result estimates the degree of confidence that the examiner has in the justified prediction provided by the justifier.

When an agent A_i solves a problem P, A_i may predict one or more possible solutions for P. Therefore, an agent may provide one or more justified predictions, and each one of them is expressed as a JER (see Section 3.5.2.3). Thus, when solving a problem, an agent may provide one or more JERs. Moreover, each JER $\mathbf{J}$ contains the justification $\mathbf{J}.J$ that A_i has found that endorses $\mathbf{J}.S$ as the solution for P. Therefore, if an agent builds more than one JER, each JER will be examined separately by the examiner.

Let A_e be the examiner agent intending to examine a justified prediction expressed as a JER $\mathbf{J}$ built by the justifier agent A_j. The justification $\mathbf{J}.J$ contains the relevant information that A_j has used for predicting $\mathbf{J}.S$ as the solution for P. In order to assess the confidence on the justification $\mathbf{J}.J$, A_e retrieves from its individual case base C_e all the cases that are subsumed by $\mathbf{J}.J$, i.e. all the cases that satisfy that justification. Let $C_e^{\mathbf{J}.J} = \{c \in C_e | \mathbf{J}.J \sqsubseteq c.P\}$ be the set of cases retrieved from C_e such that they satisfy $\mathbf{J}.J$.

In general, the set of retrieved cases $C_e^{\mathbf{J}.J}$ will contain both counterexamples and endorsing cases. The set of counterexamples $CE(\mathbf{J}, C_e) \subseteq C_e^{\mathbf{J}.J}$ are the cases that are not consistent with the justification, since they satisfy the justification but do not have $\mathbf{J}.S$ as solution class. Therefore, the more counterexamples in the set of retrieved cases, the less consistent is the justified prediction with respect to the case base C_e. The set of endorsing cases $EC(\mathbf{J}, C_e) \subseteq C_e^{\mathbf{J}.J}$ are the set of cases that are consistent with the justification and, the more endorsing cases the more consistent is the justifier prediction with respect to C_e.

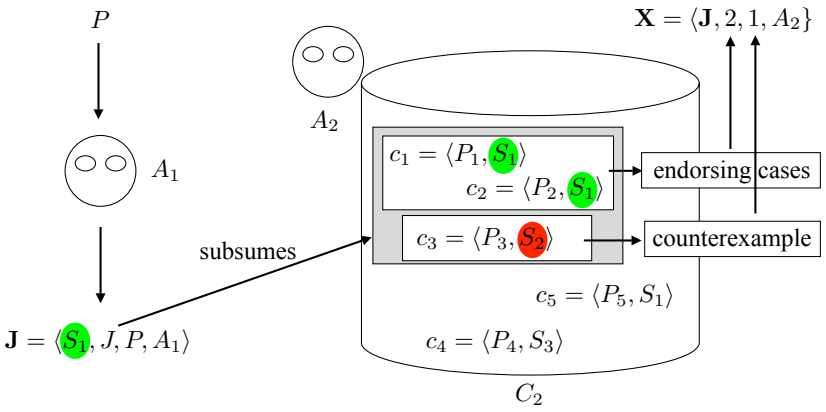


Figure 7.10: Agent A_2 examines the justification endorsement record provided by A_1 after solving the problem P.

In order to assess the confidence of a JER $\mathbf{J}$, the examiner agent will count the number of counterexamples and of endorsing cases in the set of cases retrieved by the justification $\mathbf{J}.J$ from its case base C_e. Specifically, let $Y = \#(EC(\mathbf{J}, C_e))$ be the number of endorsing cases in C_e (those consistent with the justified prediction) and let $N = \#(CE(\mathbf{J}, C_e))$ be the number of counterexamples in C_e (those not consistent with the justified prediction).

With the numbers N and Y computed, an agent A_j can compute the confidence of a JER $\mathbf{J}$ as follows:

$$\mathsf{C}(\mathbf{J}) = \frac{Y}{Y + N}$$

Notice that when all the cases are consistent with the justification ($N = 0$), the confidence takes value 1, and when all the cases are counterexamples ($Y = 0$ and $N > 0$), the confidence takes value 0.

Finally, all the information obtained during the examination process of a justified prediction expressed as a JER $\mathbf{J}$ is stored in an *Examination Record*:

Definition 7.4.1. *A* Examination Record *(XER) is a tuple* $\mathbf{X} = \langle \mathbf{J}, Y, N, A \rangle$ *where an agent* $\mathbf{J}$ *is a JER,* $Y = \#(EC(\mathbf{J}, C_e))$ *is the number of endorsing cases found by* A *for the justification* $\mathbf{J}.J$*, and* $N = \#(CE(\mathbf{J}, C_e))$ *is the number of counterexamples cases found by* A *for the justification* $\mathbf{J}.J$*.*

Examination Records allow agents to communicate the results of examining a JER against a case base, as we will explain in next section. Figure 7.10 shows an agent A_2 examining a Justification Endorsement Record built by an agent A_1. Specifically, the agent A_1 has build a JER $\mathbf{J} = \langle S_1, J, P, A_1 \rangle$ after solving a problem P. To examine $\mathbf{J}$, the agent A_2 retrieves all the cases from its case base C_2 that are subsumed by $\mathbf{J}.J$, namely c_1, c_2 and c_3. From these cases, c_1 and c_2 are endorsing cases, since their solutions are $c_1.S = c_2.S = S_1$, exactly the

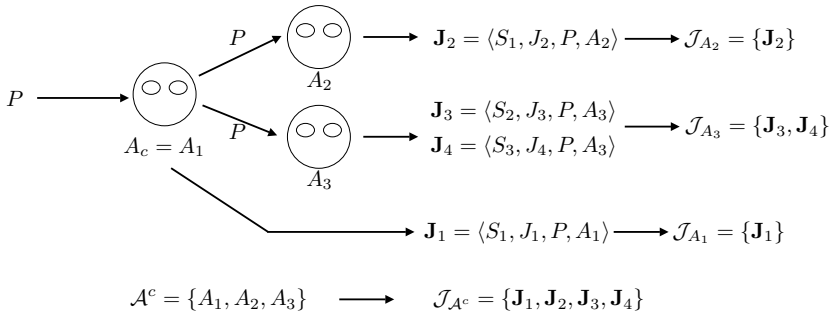


Figure 7.11: Each agent can generate one or more JERs for a problem P.

solution predicted by A_1 in the JER $\mathbf{J}$; c_3 is a counterexample since its solution $c_3.S = S_2$ is different from the one predicted by A_1. Therefore, in the example shown in Figure 7.10, A_2 builds the examination record $\mathbf{X} = \langle \mathbf{J}, 2, 1, A_2 \rangle$, since there are two endorsing cases and one counterexample.

During JE-CS each agent will examine all the examinations provided by the agents in a committee $\mathcal{A}^c$. Let us call $\mathcal{J}_{\mathcal{A}^c}$ to the set of JERs provided by a committee of agents $\mathcal{A}^c$ for a problem P.

With the previous definition of an Examination Record, we will now define the *Justification Examination* decision policy used to examine the Justification Endorsement Records in JE-CS by an examiner agent A_e:

Definition 7.4.2. *Given a set $\mathcal{J}_{\mathcal{A}^c}$ of JERs submitted by the agents in a committee $\mathcal{A}^c$, the* Justification Examination *decision policy used by an agent A_e consists in constructing an XER $\mathbf{X}_\mathbf{J}$ for each JER $\mathbf{J} \in \mathcal{J}_{\mathcal{A}^c}$:*

$$D_{JE}(\mathcal{J}_{\mathcal{A}^c}) = \{\mathbf{X}_\mathbf{J} = \langle \mathbf{J}, Y, N, A_e \rangle\}_{\mathbf{J} \in \mathcal{J}_{\mathcal{A}^c}}$$

The next section presents the voting system used by the convener agent to aggregate the individual predictions using the confidence values computed using D_{JE}.

7.5 Justification Endorsed Voting System

This section presents the *Justification Endorsed Voting* decision policy (D_{JEV}) used by the convener agent to aggregate the individual justified predictions of all the agents in the committee to obtain a global prediction. D_{JEV} is based on the *Justification Endorsed Voting* system (JEV).

The JEV voting system assumes that each individual agent may provide several JERs (as long as they contain different solution classes, i.e. an agent cannot provide two JERs with the same solution class). Figure 7.11 shows that each agent in a committee can provide one or more JERs for a given problem

P. Let $\mathcal{J}_{A_i}$ be the collection of JERs built by an agent A_i and let $\mathcal{J}_{\mathcal{A}^c}$ be the collection of JERs built by a committee of agents $\mathcal{A}^c$. Figure 7.11 illustrates these concepts with a committee formed by three agents.

Each JER contains a solution class endorsed by a justification, and is considered as a vote. Each one of these votes has a different weight assigned in function of the examinations built for that JER by the agents in the committee. The weight of each vote is a real number in the interval [0,1], that corresponds to the confidence of that JER. The *overall confidence* of $\mathbf{J}$ can be computed as follows:

Definition 7.5.1. *The* Overall Confidence *of a Justification Endorsement Record* $\mathbf{J}$ *computed by aggregating the Examination Records computed by a committee of agents* $\mathcal{A}^c$ *is:*

$$\mathsf{C}_{\mathcal{A}^c}(\mathbf{J}) = \frac{\sum_{A_i \in \mathcal{A}^c} \mathbf{X}^{A_i}.Y}{\sum_{A_i \in \mathcal{A}^c} \mathbf{X}^{A_i}.Y + \mathbf{X}^{A_i}.N}$$

where $\mathbf{X}^{A_i}$ *is the examination record build by* A_i *for the justification endorsement record* $\mathbf{J}$.

Notice that in order to compute the overall confidence an agent has also to provide a self-examination, i.e. an agent also constructs examination records for its own justification endorsement records.

Using the overall confidence definition, we can define the *Justification Endorsed Vote* (JVote) of an agent A_i for a given solution class S_k as the sum of the weight of the JER provided by A_i with solution S_k:

$$JVote(S_k, P, A_i) = \begin{cases} \mathsf{C}_{\mathcal{A}^c}(\mathbf{J}) & \text{if } \exists \mathbf{J} \in \mathcal{J}_{A_i} : \mathbf{J}.S = S_k \ , \\ 0 & \text{otherwise.} \end{cases} \tag{7.1}$$

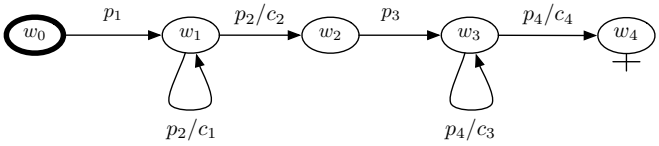
We can aggregate the votes of all the agents in $\mathcal{A}^c$ for one class by computing the ballot for that class:

$$JBallot(S_k, P, \mathcal{A}^c) = \sum_{A_i \in \mathcal{A}^c} JVote(S_k, P, A_i) \tag{7.2}$$

and therefore, the winning solution class is the class with more votes in total, i.e.:

$$JSol(\mathcal{S}, P, \mathcal{A}^c) = \arg\max_{S_k \in \mathcal{S}} JBallot(S_k, P, \mathcal{A}^c) \tag{7.3}$$

Notice that using this voting system, the number of votes that an agent is able to cast is only limited to the number of different JERs it can provide (since an agent can provide a JER for each different solution class, the number of votes an agent can cast is limited by the number of solution classes). However, if those JERs have poor justifications, they will have low confidence and therefore will have a low influence in the outcome of the voting.



p_1 :	$Request(?A^c, ?\mathcal{A}^c, ?P)$
p_2/c_1 :	$Inform(?A_j, !A^c, ?\mathbf{J})/$ $\lvert !w_0w_1\mathbf{J}\rvert = \#(\mathcal{A}^c) - 2$
p_2/c_2 :	$Inform(?A_j, !A^c, ?\mathbf{J})/$ $\lvert !w_0w_1\mathbf{J}\rvert < \#(\mathcal{A}^c) - 2$
p_3 :	$Request(!A^c, !\mathcal{A}^c, ?\mathbf{J}(P))$
p_4/c_3 :	$Inform(?A_j, !A^c, ?X_{A_j}(P))/$ $\lvert w_2w_3\mathbf{X}\rvert < \#(\mathcal{A}^c) - 2$
p_4/c_4 :	$Inform(?A_j, !A^c, ?X_{A_j}(P))/$ $\lvert w_2w_3\mathbf{X}\rvert = \#(\mathcal{A}^c) - 2$

Figure 7.12: Interaction protocol for the JE-CS collaboration strategy.

Using the JEV voting system we can define the *Justification Endorsement Voting* decision policy used to aggregate the individual agents' predictions into an overall prediction:

Definition 7.5.2. *The* Justification Endorsement Voting *decision policy used by the convener agent in order to aggregate the individual predictions into an overall prediction is:*

$$D_{JEV}(\mathcal{S}, P, \mathcal{A}^c) = JSol(\mathcal{S}, P, \mathcal{A}^c)$$

where $\mathcal{S}$ is the set of all the possible solution classes.

The next section presents the Justification Endorsed Committee Collaboration Strategy (JE-CS) that can be used by a committee of agents to improve their classification accuracy.

7.6 Justification Endorsed Committee Interaction Protocol

Figure 7.12 presents the interaction protocol used in the JE-CS collaboration strategy. Let A_c an agent that has convened a committee of agents $\mathcal{A}^c$ to solve a problem P. The protocol works as follows:

1. The initial state of the protocol is w_0. The convener agent A_c broadcasts the problem P (with message p_1) to the rest of agents in $\mathcal{A}^c$ and the protocol moves to w_1.

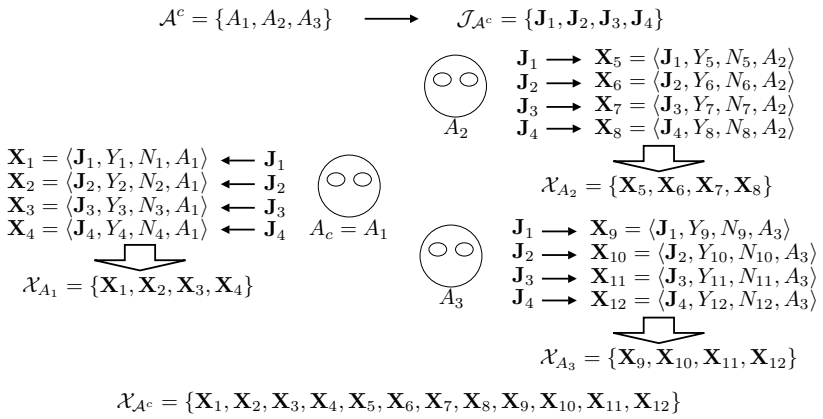


Figure 7.13: Each agent in the committee generates an examination record for every justification endorsement record in $\mathcal{J}_{\mathcal{A}^c}$.

2. In the state w_1, every agent $A_i \in \mathcal{A}^c$ in the committee solves the problem individually and builds $\mathcal{J}_{A_i}$ (as shown in Figure 7.11 for a system composed of 3 agents), and sends $\mathcal{J}_{A_i}$ to the convener agent A_c with message p_2. The convener agent also solves the problem individually and locally stores its own set of JERs.

3. When the convener agent has received the JERs from all the agents in the committee, the protocol moves to state w_2. The convener agent has collected the JERs $\mathcal{J}_{\mathcal{A}^c} = \cup_{A \in \mathcal{A}^c} \mathcal{J}_A$ (that includes also its own JERs). Then the convener agent broadcasts $\mathcal{J}_{\mathcal{A}^c}$ to all the other agents with message p_3 and the protocol moves to state w_3.

4. In state w_3, every agent in the committee has $\mathcal{J}_{\mathcal{A}^c}$, the collection of all JERs . Each agent A_i in the committee uses their individual D_{JE} decision policy to generate the examination endorsement records for the JERs in $\mathbf{J}_{\mathcal{A}^c}$:

   (a) For each JER $\mathbf{J} \in \mathcal{J}_{\mathcal{A}^c}$, agent A_i examines the justification $\mathbf{J}.J$ of the JER against its case base,

   (b) after contrasting the justification against the local case base, A_i builds a XER $\mathbf{X} = \langle \mathbf{J}, Y, N, A_i \rangle$ (Figure 7.13 illustrates this step for a system composed of 3 agents),

   (c) Agent A_i builds a XER for each justification in $\mathbf{J}_{\mathcal{A}^c}$ and the result is a collection of XERs $\mathcal{X}^{A_i}$.

   (d) A_i sends the collection of XERs $\mathcal{X}^{A_i}$ to the convener agent A_c (except if A_i is the convener agent itself).

5. When the convener agent receives the examination records of all the agents in the committee, the protocol moves to step w_4 (a final state). We will call $\mathcal{X}^{\mathcal{A}^c} = \cup_{A_i \in \mathcal{A}^c} \mathcal{X}^{A_i}$ the set of all the XERs built by all the agents in $\mathcal{A}^c$ (as illustrated in Figure 7.13). The convener agent A_c can now perform a weighted voting system on the solution classes using its D_{JEV} decision policy, i.e. the convener agent uses the JEV voting system presented in section 7.5 to compute an aggregate solution.

We have just presented the JE-CS collaboration strategy, including its two individual decision policies and its interaction protocol. Before presenting the experimental results, the next section will show that the examination process of the JERs is not sensitive to the distribution of cases among the agents.

7.6.1 Confidence Estimation Robustness

We will now show that the examination process of the JERs is not sensitive to the distribution of cases among the agents (i.e. the examination process is not affected by Committee Bias, see Section 4.4) or to the number of agents in the system while maintaining the same collection of cases. In other words, given a JER $\mathbf{J}$, the overall confidence value $\mathsf{C}(\mathbf{J})$ computed will not change if we completely redistribute the cases among the agents. $\mathsf{C}(\mathbf{J})$ also remains unchanged if we change the number of agents while maintaining the same collection of cases. This strong result ensures that the confidence measures computed are robust and that they are always computed taking advantage of all the information available in the system. However, the confidence values estimated indeed depend on the degree of Committee Redundancy of the system.

To prove the previous statements, we just have to prove two things:

1. That moving one case from one agent's case base to another agent's case base the overall confidence estimation $\mathsf{C}_{\mathcal{A}^c}(\mathbf{J})$ doesn't change (since we can and obtain any case distribution by transferring cases one by one), and

2. that adding/removing an agent with an empty case base the overall confidence estimation $\mathsf{C}_{\mathcal{A}^c}(\mathbf{J})$ does not change either.

Let us assume that in a committee consisting of n agents: $\mathcal{A}^c = \{A_1, ..., A_n\}$, with case bases $C_1, ..., C_n$, the JER to be examined is $\mathbf{J}$. After the examination process, every agent $A_j \in \mathcal{A}^c$ has computed the number of endorsing cases and of counterexamples, $\mathbf{X}^{A_j}.Y$ and $\mathbf{X}^{A_j}.N$, where $\mathbf{X}^{A_j}$ is the examination record build by A_j for the justification endorsement record $\mathbf{J}$. Now, if we transfer a case $c \in C_i$ to C_j we obtain the following case base $C_i^* = C_i - \{c\}$ and $C_j^* = C_j + \{c\}$, and three situations may arise:

1. $\mathbf{J}.J \not\sqsubseteq c.P$: In this situation, since the case c is not subsumed by the justification, the new values $\mathbf{X}_{\mathbf{J}}^{A_i}.Y^*$, $\mathbf{X}_{\mathbf{J}}^{A_i}.N^*$, $\mathbf{X}_{\mathbf{J}}^{A_j}.Y^*$ and $\mathbf{X}_{\mathbf{J}}^{A_j}.N^*$ computed for agents A_i and A_j are the same than the previous $\mathbf{X}_{\mathbf{J}}^{A_i}.Y$, $\mathbf{X}_{\mathbf{J}}^{A_i}.N$,

$\mathbf{X}_{\mathbf{J}}^{A_j}.Y$ and $\mathbf{X}_{\mathbf{J}}^{A_j}.N$ (The asterisk is notes the values computed using case bases C_i^* and C_j^*). Thus, $\mathsf{C}_{\mathcal{A}^c}(\mathbf{J})$ does not change.

2. $\mathbf{J}.J \sqsubseteq c.P$ and $\mathbf{J}.S = c.S$: In this situation, the new value $\mathbf{X}_{\mathbf{J}}^{A_i}.Y^*$ computed by A_i is equal to $\mathbf{X}_{\mathbf{J}}^{A_i}.Y - 1$ and $\mathbf{X}_{\mathbf{J}}^{A_j}.Y^*$ computed by A_j is equal to $Y\mathbf{X}_{\mathbf{J}}^{A_j}.Y + 1$. The confidence will be then:

$$\mathsf{C}_{\mathcal{A}^c}(\mathbf{J}) = \frac{\sum_{A_j \in \mathcal{A}^c} \mathbf{X}_{\mathbf{J}}^{A_j}.Y^*}{\sum_{A_j \in \mathcal{A}^c} \mathbf{X}_{\mathbf{J}}^{A_j}.Y^* + \mathbf{X}_{\mathbf{J}}^{A_j}.N^*} =$$

$$\frac{(\sum_{A_j \in \mathcal{A}^c} \mathbf{X}_{\mathbf{J}}^{A_j}.Y) + 1 - 1}{(\sum_{A_j \in \mathcal{A}^c} \mathbf{X}_{\mathbf{J}}^{A_j}.Y + \mathbf{X}_{\mathbf{J}^{A_j}.N}) + 1 - 1} =$$

$$\mathsf{C}_{\mathcal{A}^c}(\mathbf{J}) = \frac{\sum_{A_j \in \mathcal{A}^c} \mathbf{X}_{\mathbf{J}}^{A_j}.Y}{\sum_{A_j \in \mathcal{A}^c} \mathbf{X}_{\mathbf{J}}^{A_j}.Y + \mathbf{X}_{\mathbf{J}}^{A_j}.N}$$

Therefore, the value of $\mathsf{C}(\mathbf{J})$ remains unchanged.

3. $\mathbf{J}.J \sqsubseteq c$ and $\mathbf{J}.S \neq c.S$: In this case, the proof is the same than in the previous situation, but the changed values are $\mathbf{X}_{\mathbf{J}}^{A_i}.N$ and $\mathbf{X}_{\mathbf{J}}^{A_j}.N$.

Therefore the confidence estimation does not depend on the distribution of cases among the agents.

The second proof we need is showing that the estimation of the confidence remains also unchanged if we change the number of agents while maintaining the same collection of cases. For this purpose, we just have to consider what happens if we add/remove an agent A_i to/from the system with an empty case base: as the numbers $\mathbf{X}^{A_i}.Y$ and $\mathbf{X}^{A_i}.N$ computed by this agent will equal zero, adding or removing this agent from the system will not modify the value of $\mathsf{C}(\mathbf{J})$. Therefore, as redistributing cases among the agents doesn't modify the confidence estimate, removing an agent by first transferring all its cases to other agents, and then removing it also do not modify the confidence estimate. Similarly, adding an agent with an empty case base and transferring some cases to it from another agent's case base does not modify the overall confidence estimation $\mathsf{C}_{\mathcal{A}^c}(\mathbf{J})$ of any JER $\mathbf{J}$. Therefore we have proven that the overall confidence estimations are not dependent on the case distribution among the agents nor the number of agents in the system.

Notice, however, the overall confidence estimations indeed depend on the degree of redundancy, i.e. if we duplicate one case, the confidence estimate will change. Nonetheless, for a fixed degree of redundancy, changing the distribution or the number of agents, the overall confidence estimation will remain unchanged.

The next section presents an exemplification of the JE-CS collaboration strategy.

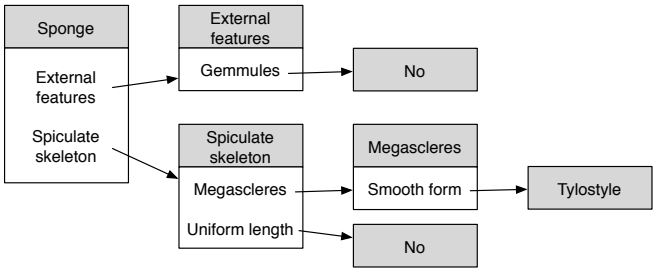


Figure 7.14: Justification J_1 generated by agent A_1 after solving the problem P.

7.6.2 Exemplification

Let us show how the JE-CS collaboration strategy works by means of an example. Let $\mathcal{M}$ be a $\mathcal{M}$AC system composed of three agents, namely $\mathcal{A} = \{A_1, A_2, A_3\}$. The agent A_1 wants to solve the problem P (thus, A_1 will play the role of *convener* agent). P is a problem of marine sponge classification, and therefore the possible solution classes are $\mathcal{S} = \{Hadromerida, Axinellida, Astrophorida\}$.

First, A_1 convenes a committee composed of agents $\mathcal{A}^c = \{A_1, A_2, A_3\}$, then A_1 sends the problem P to the other members of the committee, namely to A_2 and to A_3. All three agents try to solve the problem individually using LID [9] as the CBR method, and build their justified predictions expressed as JERs as explained in Section 3.5.2.1.

After solving the problem P using LID, A_1 has found that the solution class for the problem P is *Hadromerida* and the justification is the one shown in Figure 7.14. Therefore, A_1 builds a justified endorsement record $\mathbf{J}_1 = \langle Hadromerida, J_1, P, A_1\rangle$, where J_1 is the justification shown in Figure 7.14. Then, $\mathbf{J}_1$ is sent to A_2 and A_3. Analogously, A_2 builds $\mathbf{J}_2 = \langle Axinellida, J_2, P, A_2\rangle$ and A_3 builds $\mathbf{J}_3 = \langle Hadromerida, J_3, P, A_3\rangle$, and sends them to the other agents (notice that each agent could generate more than one JER if J_i covers cases in more than one solution class).

At this point, each agent (A_1, A_2, and A_3) has the set $\mathcal{J}_{\mathcal{A}^c} = \{\mathbf{J}_1, \mathbf{J}_2, \mathbf{J}_3\}$, containing all the JERs built by all the agents in the committee. It's time to build the examination records.

When A_2 starts examining $\mathbf{J}_1$ coming from the agent A_1, all the cases in the case base of A_2 that are subsumed by the justification $\mathbf{J}_1.J$ are retrieved: 13 cases, 8 cases belonging the *Hadromerida* solution class, and 5 cases belonging to *Astrophorida*. That means that A_2 knows 5 counterexamples (i.e. cases that completely satisfy the justification $\mathbf{J}_1.J$, but that do not belong to the *Hadromerida* class). Therefore, A_2 builds the examination record $\mathbf{X}_1 = \langle \mathbf{J}_1, 8, 5, A_2\rangle$. Then, A_2 continues by examining the next JER, $\mathbf{J}_2$ (its own JER), and finds only 3 cases belonging the *Axinellida* class. Therefore, A_2

builds the following XER: $\mathbf{X}_2 = \langle \mathbf{J}_2, 3, 0, A_2 \rangle$ (in fact, when an agent examines its own JERs, it is not needed to perform the full examination process, since the agent already knows which are the endorsing cases and counterexamples for that JER). Finally, A_2 also builds the XER for $\mathbf{J}_3$: $\mathbf{X}_3 = \langle \mathbf{J}_3, 5, 1, A_2 \rangle$. Those three XERs are sent to the convener agent A_1. In the same way, A_3 also builds its own XERs and sends them to A_1. The convener agent A_1 also builds its own XERs.

At this point of the protocol, the convener agent has the set of XERs built by itself and the ones that has received from the other agents. Specifically, the convener A_1 has the XERs $\mathbf{X}_1$, $\mathbf{X}_2$, and $\mathbf{X}_3$ (previously mentioned), and also the rest of the XERs built by the agents: $\mathbf{X}_4 = \langle \mathbf{J}_1, 7, 0, A_3 \rangle$, $\mathbf{X}_5 = \langle \mathbf{J}_2, 2, 5, A_3 \rangle$, $\mathbf{X}_6 = \langle \mathbf{J}_3, 10, 0, A_3 \rangle$, $\mathbf{X}_7 = \langle \mathbf{J}_1, 15, 0, A_1 \rangle$, $\mathbf{X}_8 = \langle \mathbf{J}_2, 1, 4, A_1 \rangle$, $\mathbf{X}_9 = \langle \mathbf{J}_3, 6, 1, A_1 \rangle$. With all those examination records, A_1 builds the set $\mathcal{X}^{\mathcal{A}^c} = \{\mathbf{X}_1, ..., \mathbf{X}_2, \mathbf{X}_3, \mathbf{X}_4, \mathbf{X}_5, \mathbf{X}_6, \mathbf{X}_7, \mathbf{X}_8, \mathbf{X}_9\}$. Then, A_1 computes the overall confidence measures for each justification endorsement record in the set $\mathcal{J}_{\mathcal{A}^c}$ using Definition 7.5.1. For instance, for the JER $\mathbf{J}_1$, the confidence measure will be obtained from the XERs $\mathbf{X}_1$, $\mathbf{X}_4$ and $\mathbf{X}_7$ (the XERs that refer to $\mathbf{J}_1$):

$$\mathsf{C}_{\mathcal{A}^c}(\mathbf{J}_1) = \frac{8 + 7 + 15}{8 + 5 + 7 + 0 + 15 + 0} = 0.85$$

In the same way, the confidence $\mathsf{C}(\mathbf{J}_2) = 0.40$ will be computed from $\mathbf{X}_2$, $\mathbf{X}_5$ and $\mathbf{X}_8$ and the confidence $\mathsf{C}(\mathbf{J}_3) = 0.91$ from $\mathbf{X}_3$, $\mathbf{X}_6$ and $\mathbf{X}_9$. Notice how the weakest JER ($\mathbf{J}_2$) has obtained the lowest confidence, while stronger justifications obtain higher confidence values.

Once all the confidence measures of the JERs have been computed the Justification Endorsing Voting can be used in order to obtain an overall solution. For instance, the votes for agent A_1 are:

- $JVote(Hadromerida, A_1) = 0.85$,
- $JVote(Axinellida, A_1) = 0.0$,
- and $JVote(Astrophorida, A_1) = 0.0$.

Notice that A_1 has provided only one JER, that endorses the solution class $Hadromerida$ as the solution for the problem P, therefore its votes for the rest of the classes are zero.

The JVotes of A_2 and A_3 are computed similarly and then the votes of all the agents are aggregated as ballots:

- $JBallot(Hadromerida, P, \mathcal{A}^c) = 0.85 + 0.91 = 1.76$,
- $JBallot(Axinellida, P, \mathcal{A}^c) = 0.40$,
- $JBallot(Astrophorida, P, \mathcal{A}^c) = 0.0$.

Finally, the outcome of the voting system is the class with the highest ballot, namely $JSol(\mathcal{S}, P, \mathcal{A}^c) = Hadromerida$.

7.7 Experimental Evaluation

In this section the performance of the Justification Endorsed Collaboration Collaboration Strategy (JE-CS) will be empirically evaluated and compared with the performance of the Committee Collaboration Strategy (CCS) and with the performance of individual agents. We have made experiments with $\mathcal{M}$AC systems consisting in 3, 5, 7, 9, 11, 13, and 15 agents. The data sets used are sponges, soybean and zoology, and the results presented are the average of 5 runs of a 10 fold cross validation. In order to test the generality of the JE-CS strategy, we present experiments using LID and decision trees as the learning methods. Moreover, decision trees have only been tested with the soybean and zoology data sets since the sponge data set is not a propositional data set and decision trees cannot directly manage it. Moreover, we have made experiments in three different scenarios (as in the previous chapters): the *uniform* scenario, the *redundancy* scenario, and the *biased* scenario.

In the *uniform* scenario each individual agent receives a random sample of the data set with Committee Redundancy $\mathbb{R} \backsimeq 0.0$ and with Committee Bias $\mathbb{B} = 0.0$ (see Section 4.4). That is to say, in the uniform scenario the case bases of the individual agents are disjunct (i.e. there is no case shared by two agents' case bases), and the individual case bases have an average Case Base Bias of 0 (i.e. the ratio of cases of each class in the individual case bases is nearly the same than in the complete data set).

In the *redundancy* scenario each individual agent receives a random sample of the data set with redundancy $\mathbb{R} = 0.1$ and with Committee Bias $\mathbb{B} = 0.0$. That is to say, in the redundancy scenario, there is a number of cases that are present in more than one agents' case bases. For instance, with a Committee Redundancy $\mathbb{R} = 0.1$ in a $\mathcal{M}$AC system composed of 5 agents working in the sponges domain each agent will have an average of 71.12 cases while with Committee Redundancy $\mathbb{R} = 0.0$ they will have only about 54.00 cases. The goal of performing experiments in the redundancy scenario is to test if the justification examination process is sensible to committee redundancy.

In the *biased* scenario each individual agent receives a random sample of the data set with Committee Redundancy $\mathbb{R} \backsimeq 0.0$ and with Committee Bias $\mathbb{B} > 0.0$. Specifically, we have set $\mathbb{B} = 0.45$ for the sponge data set, $\mathbb{B} = 0.17$ for the soybean data set, and $\mathbb{B} = 0.32$ for the zoology data set. In this way, each agent will receive case bases that are not representative of the training set, and therefore, each individual agent has a low classification accuracy (as we showed in Section 4.5.3). In this scenario, we want to evaluate whether JE-CS can compensate the reduced classification accuracy of the individual agents and achieve a higher classification accuracy than CCS.

7.7.1 JE-CS evaluation in the uniform scenario

Figure 7.15 shows the classification accuracy for agents using LID as learning method in the three data sets (sponge, soybean and zoology). The figure shows classification accuracy for agents using JE-CS, CCS and for agents solving prob-

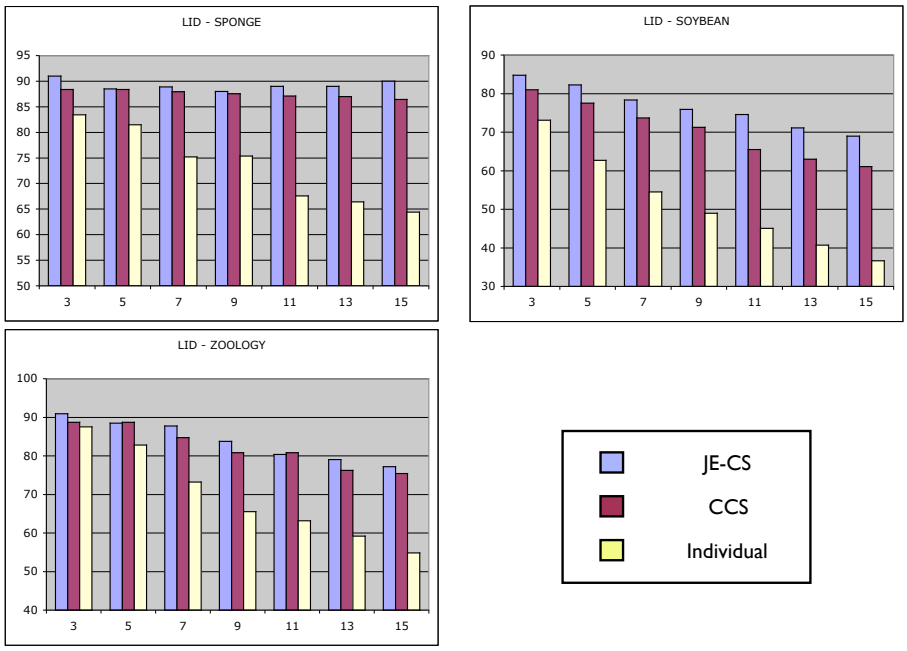


Figure 7.15: Classification accuracy comparison between agents using LID to solve problems with JE-CS, CCS, and individually in the uniform scenario.

lems individually. The first thing that Figure 7.15 shows is that agents using JE-CS achieve higher classification accuracy values in almost all the systems than agents using CCS, and specially in the soybean data set.

The difference in classification accuracy between JE-CS and CCS is specially significative in systems with many agents. For instance, in the 15 agents system for the sponge data set, the classification accuracy achieved with CCS is 86.45% while with JE-CS it is 90.00% . This can be explained by the fact that in systems with many agents, each individual agent has a lower classification accuracy (as shown in Figure 7.15), and therefore there are more chances to detect low confidence predictions. The fact that JE-CS increases the accuracy more in these situations proves that indeed the predictions with low confidence are detected and their influence in the voting system is diminished accordingly. Moreover, this result also reinforces the fact that the overall confidence estimations are not affected by the fragmentation of data.

Notice however that the benefits of JE-CS over CCS are small in the zoology data set. The explanation is that the zoology data set is very small (consists on just 101 cases) and the cases have only a few attributes. In fact, the cases have just 16 attributes: 15 of them are binary and the other one is numerical and can take the values 2, 4, 6 and 8. Moreover, there are 7 possible solution classes. Thus, in a 15 agents system, each agent will have an average of 6.06, that is less

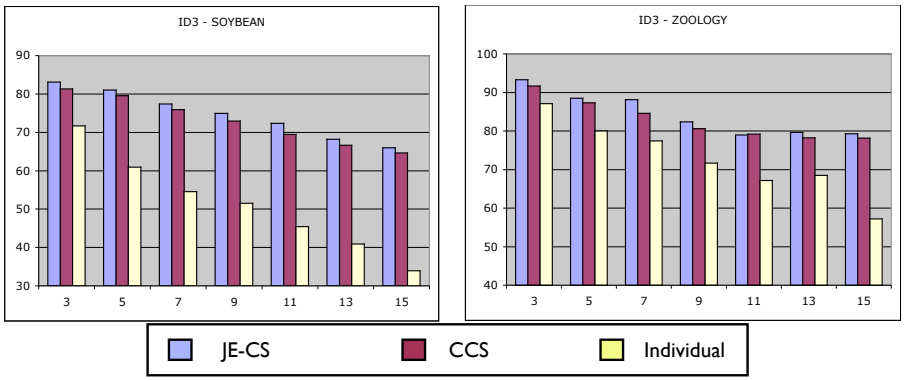


Figure 7.16: Classification accuracy comparison between agents using decision trees to solve problems with JE-CS, CCS, and individually in the uniform scenario.

than 0.87 cases in average per solution class. Therefore the individual agents have very few information with which to build interesting justifications. Notice that in the soybean and sponge data sets the larger benefits (comparing to CCS) where achieved in systems with many agents, where each agent has a small case base. That is because in a system with many agents, many agents will provide a weakly justified prediction, but some of them will provide a strongly justified prediction. However, in the zoology data set, as each agent has so few cases, the most probable situation is that all of them provide a weakly justified prediction, and in this case JE-CS can do little to improve classification accuracy. Thus, a requisite of JE-CS to work is that there are at least some strongly justified predictions among the predictions provided by the agents in a committee (that will be the ones that will be assigned higher overall confidence estimations). If all the justified predictions are weak, then JE-CS converges to CCS.

Nevertheless, notice that even in this data set where the agents are not able to build very interesting justifications JE-CS is able to achieve most of the times slightly better results than CCS.

Figure 7.16 shows the classification accuracy for agents using decision trees as learning method in the soybean and zoology data sets. The figure shows that, although the differences are not as large as using LID, agents using JE-CS achieve higher classification accuracy values than agents using CCS. In the soybean data set, agents using JE-CS achieve an accuracy that is always about a 2% above of that achieved by agents using CCS. Specifically, in the 9 agents system the accuracy of JE-CS is 74.92% while the accuracy of CCS is 72.96%. Notice also that the gains achieved in the zoology data set are smaller than those achieved in the soybean data set. The explanation is the same than for the LID method, and again we see that although the gains are small, JE-CS achieves usually higher accuracy than CCS.

Summarizing, JE-CS achieves higher classification accuracy values than CCS

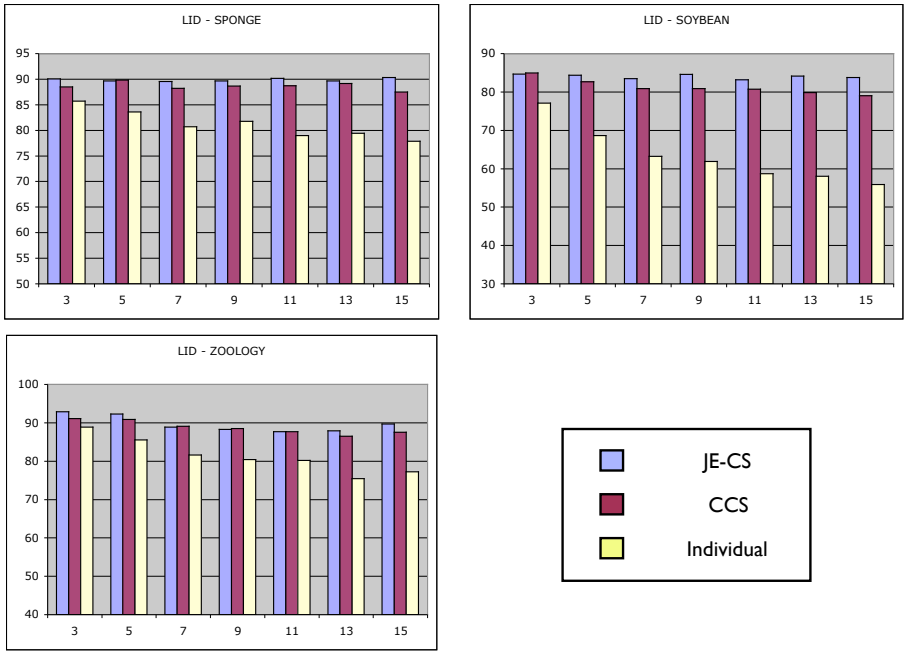


Figure 7.17: Classification accuracy comparison between agents using LID to solve problems with JE-CS, CCS, and individually in the redundancy scenario.

in all the data sets both using LID and decision trees. In those data sets where justifications are richer (sponge and soybean) the difference in classification accuracy between JE-CS and CCS is larger, and in those data sets where justifications contain less information (such as the zoology data set where the individual agents have very few cases) JE-CS approaches CCS. Finally, we have seen agents using LID obtain more benefits using JE-CS than agents using decision trees. The explanation is that LID is a lazy learning method that explicitly uses a symbolic description of the similarity among cases, decision trees however are learnt using an eager learning algorithm and do not have any explicit representation of the similarity among cases; we use the tree path that classifiers a problem as its justification but, since the tree has been generated using an eager learning method, the justification is not specific to the problem at hand as it is in LID.

7.7.2 JE-CS evaluation in the redundancy scenario

In the redundancy scenario there are cases that are present in more than one agent's case base. As we have shown in Chapter 4, redundancy is expected to increase the individual and CCS classification accuracy. However, as we explained in Section 7.6.1 the overall confidence estimation process is sensible to case redundancy. Therefore, it is not obvious that JE-CS will also increase

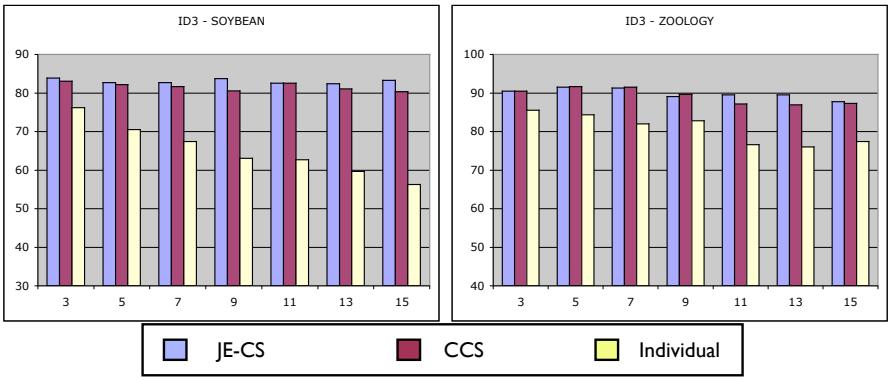


Figure 7.18: Classification accuracy comparison between agents using decision trees to solve problems with JE-CS, CCS, and individually in the redundancy scenario.

its classification accuracy values. The goal of performing experiments in this scenario is to test if redundancy affects JE-CS positively or negatively.

Figure 7.17 shows that the classification accuracy values achieved by JE-CS are higher than those achieved by CCS. Again, the increase of classification accuracy is more significant in systems with many agents. Comparing the results with the uniform scenario, we see that CCS has increased its classification accuracy values (as expected), and that JE-CS has also increased its classification accuracy values. This shows that although the overall confidence estimation process may be affected by redundancy, JE-CS is robust and experiments an increase in classification accuracy with redundancy. The larger increase has taken place in the soybean data set for the 15 agents system, where JE-CS and CCS had achieved an accuracy of 68.99% and 61.1% in the uniform scenario, while they have achieved an accuracy of 83.77% and 79.02% respectively in the redundancy scenario.

Figure 7.18 shows the classification accuracy for agents using decision trees as learning method in the soybean and zoology data sets in the redundancy scenario. Figure 7.18 confirms that JE-CS is affected positively by redundancy, and thus the classification accuracy of JE-CS in the redundancy scenario is higher than in the uniform scenario. Moreover, the classification accuracy achieved by JE-CS is higher than that achieved by CCS for both data sets.

Again, we can observe that the gains achieved in the zoology data set are not al large as in the other data sets, but that JE-CS is still able to achieve usually higher classification accuracy values than CCS. Moreover, in the situations where JE-CS is not able to improve CCS if converges to CCS.

Summarizing, we have seen that CCS improves its classification accuracy with redundancy. This could cancel the improvement achieved by JE-CS in principle. However, experiments have shown that this is not the case and JE-CS still achieves higher classification accuracy values than CCS given that the

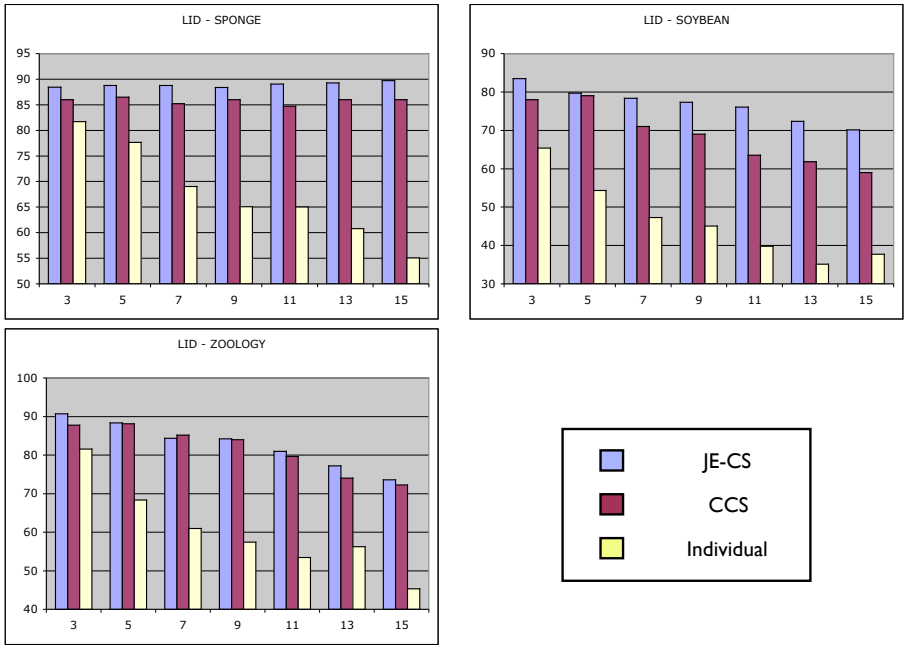


Figure 7.19: Classification accuracy comparison between agents using LID to solve problems with JE-CS, CCS, and individually in the biased scenario.

agents can provide interesting justifications.

7.7.3 JE-CS evaluation in the biased scenario

Chapter 4 has shown that the accuracy achieved by CCS is reduced in the presence of case base bias. In a committee of agents where each individual agent has a biased case base, each agent may provide individual predictions that are less reliable since they are made using a biased case base. Using the overall confidence estimation process, JE-CS tries to solve this problem by computing a confidence estimation of each individual justified prediction made by the agents in a committee. The goal of making experiments in the biased scenario is to confirm that JE-CS is more robust in the presence of bias than CCS.

Figure 7.19 shows the classification accuracy for agents using LID as learning method in the three data sets (sponge, soybean and zoology) in the biased scenario. Figure 7.19 shows that the accuracy achieved by agents using JE-CS is clearly higher than the accuracy of agents using CCS in the soybean and sponge data sets. Moreover, this difference is more noticeable in the experiments with many agents and specially large in the soybean data set. For instance, in the 15 agents system in the soybean data set, JE-CS achieves an accuracy of 70.13% while CCSonly a 59.00%. In the zoology data set, the accuracy gain is

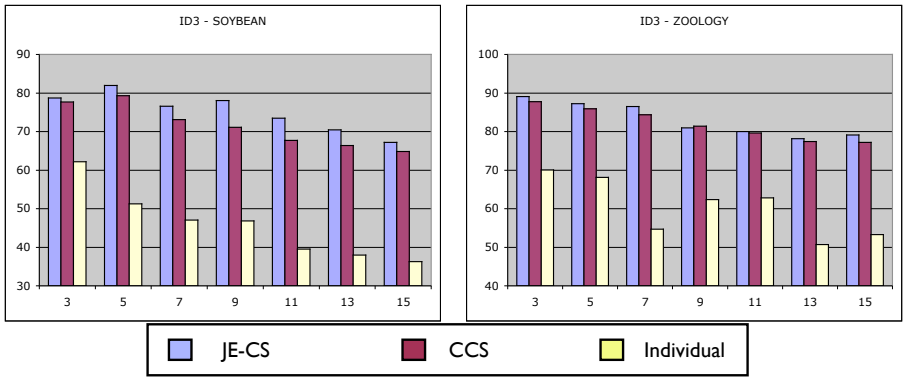


Figure 7.20: Classification accuracy comparison between agents using decision trees to solve problems with JE-CS, CCS, and individually in the biased scenario.

not so significant.

Moreover, comparing the accuracy achieved in the biased scenario (Figure 7.20) with the accuracy achieved in the uniform scenario (Figure 7.16), we can see that the accuracy of CCS is lower in the biased scenario (as expected). However, in the sponge and soybean data sets the classification accuracy of JE-CS has remained practically unchanged from the uniform scenario to the biased scenario (this effect cannot be seen in the zoology data set for the same reasons explained in the previous scenarios). Therefore, the experiment shows that the accuracy achieved by JE-CS is not affected by case base bias.

Figure 7.20 shows the classification accuracy for agents using decision trees as learning method in the soybean and zoology data sets in the redundancy scenario. The figure shows that JE-CS achieves clearly higher classification accuracy values than CCS in the soybean data set. Classification accuracy values achieved using CCS decrease with respect to the uniform scenario but accuracy values achieved using JE-CS do not in the soybean data set. In fact, classification accuracy of agents using JE-CS is higher for some systems in the biased scenario than in the uniform scenario in the soybean data set. However, agents working with the zoology data set again do not obtain such a great improvements as agents working with the soybean data set and they perform just slightly better than using CCS.

We can conclude that JE-CS is robust with respect to case base bias provided that the agents can construct interesting justifications (as in the sponge and soybean data sets).

7.8 Conclusions

In this chapter we have developed the notion of justifications presented in chapter 3. A justification is the explanation that an agent gives for having made a specific prediction for a problem. Moreover, the notion of justification has allowed us to define *justified predictions*, with which we have defined a new kind of committees where the agents can justify their individual predictions. Moreover, we have presented a method with which a committee of agents can assess the confidence of a justified prediction made by an agent by means of aggregating individual examinations made by the agents of a committee. Finally, we have presented a collaboration strategy named JE-CS, that uses justifications in order to increase the classification accuracy achieved by the Committee Collaboration Strategy presented in Chapter 4.

In order to evaluate JE-CS, we have compared it with CCS in several scenarios and using different data sets and learning methods. The results show that JE-CS achieves higher classification accuracy than CCS in all the scenarios and data sets (although in one data set, zoology, the accuracy gain is very small). Specifically, we have shown that JE-CS is specially useful when the individual predictions may not be reliable. For instance, in systems where the individual agents have small case bases or where there is some degree of committee bias, justifications are useful to assess the confidence on each individual prediction. In that way, predictions that are not endorsed by strong justifications will have small impact on the voting system and therefore on the overall prediction made by the committee.

We have shown that JE-CS is more robust than CCS. Specifically, CCS has problems when the degree of committee bias increases (as shown in Chapter 4), while the accuracy achieved by JE-CS in the presence of bias is the same than where there is no bias. However, we have seen that JE-CS achieves larger increases in classification accuracy over CCS in domains where the agents can build complex justifications (such as in sponge and soybean in our experiments); in simple domains (such as zoology) where agents cannot build interesting justifications the gains achieved by JE-CS over CCS are not that large. We can conclude that while the requisites of CCS to work is that every individual prediction of the agent members of a committee is reliable (and that they are uncorrelated), the requisites of JE-CS to work are that there have to be at least some of the predictions of the agent members that are reliable and that the justifications build by the agents have to contain enough information for the agents to assess the confidence of each individual prediction. The previous conclusion is consistent with all the experiments presented in this section:

- in the soybean and sponge data sets, the agents can build rich justifications and thus is where JE-CS achieves higher benefits; in the biased scenario there are more chances that some individual agents provide weak predictions, thus CCS performs bad, and JE-CS outperforms it;
- in systems with many agents there are also more chances of agents providing weak predictions and thus CCS performs worse than JE-CS;

- finally, in the zoology data set, agents cannot build rich justifications and thus JE-CS cannot assess which justified predictions are weak and which are strong. Therefore, JE-CS does not perform much better than CCS.

As a final conclusion we can say that justifications are an interesting tool that provide useful information that can be exploited in several ways. In this chapter we have exploited the information contained in the justifications to assess the confidence of the predictions made by individual agents, but they can have many other uses as we will see in Chapters 8 and 9.

Chapter 8

Case Retention Collaboration Strategies

In this chapter we are going to introduce collaboration strategies for case retention. The goal of the case retention strategies presented in this chapter is to achieve compact and competent case bases, i.e. case bases with a small number of cases and that have a high classification accuracy. Specifically, we are going to present strategies for case retention different from previously proposed retention strategies in that they are specifically designed for multi-agent systems or they are based on justifications in order to decide what is to be retained or not. Finally, the performance of the retention strategies is empirically evaluated.

8.1 Introduction

The performance of case based reasoning strongly depends on the contents of the case base. For that reason, building and maintaining compact and competent case bases has become a main topic in Case Based Reasoning research. Empirical results have shown that storing every case into the case base does not automatically improve classification accuracy [77]. In fact, adding cases to an already saturated case base does not increase classification accuracy but decreases problem solving performance (since the retrieve process cost increases as the number of cases in the case base increases). Therefore, it would be desirable that a case base is compact and competent, i.e. that the case base has a small number of cases (in order to have a high problem solving efficiency) and that the case base is competent (in order to have a high classification accuracy).

All the collaboration strategies presented in this chapter are designed to work during the last process of the CBR cycle (as Figure 8.1 shows): the Retain process. During the Retain process, a CBR agent incorporates new cases into its case base. The collaboration strategies presented in this chapter have the goal of taking benefit of the fact that the CBR agents are in a multi-agent system in order to improve the performance of the Retain process. To see that

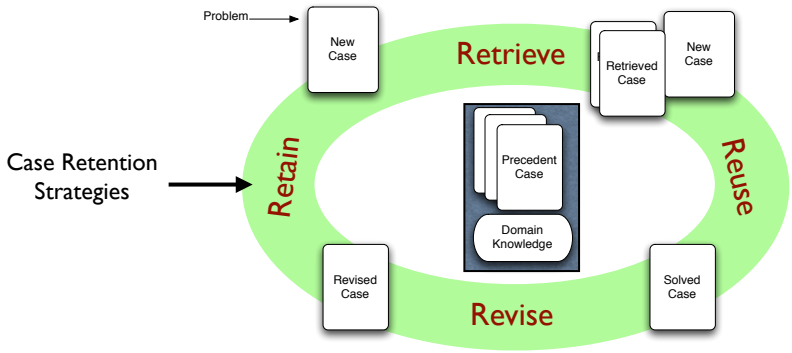


Figure 8.1: Case Retention strategies are designed to work during the last process of the CBR cycle: Retain

collaboration can have an important role in case retention, consider the following situation: we have a $\mathcal{M}$AC system composed of two agents A_1 and A_2; agent A_1, because of the conditions of the environment in which it is working, can very easily have access to new cases with a certain solution class S_1; and agent A_2 can very easily have access to new cases with solution class S_2; if both agents reach an agreement so that A_1 cedes some cases with solution S_1 (that would have been discarded otherwise) to A_2 and viceversa their case bases may be better (in the sense of achieving a higher classification accuracy) than the case bases they could have by working in isolation.

Moreover, collaboration strategies for case retention are not different to individual case retention strategies only because the agents may have access to cases discarded by other agents. Individual retention strategies are designed to maximize the performance of an individual CBR system, while collaboration strategies for case retention may have the goal of maximizing the performance of a committee. In this case, the retention strategy must have in mind that in order to improve the performance of committee the individual classification accuracy is not the only important factor: as we have explained in previous chapters, committees rely on the ensemble effect and it is a requisite of the ensemble effect that the errors made by the individual agents are not correlated. Thus, a collaboration strategy for case retention has at least two goals: increasing individual accuracy and maintaining (or decreasing) error correlation.

In this chapter we are going to present strategies for case retention that differ from the currently available case retention strategies (see Section 2.2.1.1) because they are collaborative case retention strategies or because they are based on justifications. Specifically, in this chapter we are going to present three different case retention collaboration strategies: Section 8.2 presents several variations of the *multi-agent case retention* collaboration strategy (MAR-CS) that is a retention strategy that takes into account the multi-agent aspect of case

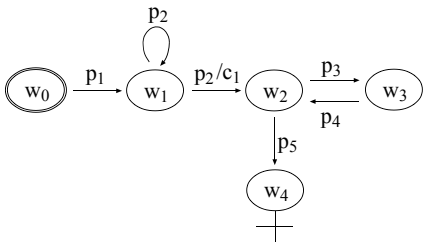
retention; then Section 8.3 presents the *Justification-based Selection of Training Examples* (JUST) strategy, that is an individual case retention strategy that uses justifications in order to select which cases to retain; finally, Section 8.4 presents the *Collaborative Case Bargaining* collaboration strategy, that is a retention strategy that combines both justifications and the multi-agent aspect of retention. The chapter ends with a conclusions section.

8.2 Multi-agent Case Retention Strategies

CBR case retention strategies are in charge of deciding which new cases to incorporate into the case base. This decision takes place after the *Revise Process* (see Figure 8.1). Thus, the CBR approach to case retention consists on the following: given a new problem the decision of whether to retain it or not is taken after revising it. However, this is not the only possible approach to case retention. A main issue on machine learning is to select which are the examples of the target problem to learn from. Each time a learning system receives a new example, it has two options: use the example to learn (retain) or discard it. When a learner retains every example it observes, we are talking of *passive learning*. But when the learner has some strategy to select which are the examples that it is going to learn from, we are talking of *active learning* [20]. The basic idea in active learning is that the learner receives a set of unlabelled examples and decides which of them are interesting to learn from; then the teacher labels the examples that the learner has found interesting and only then they are used for learning. The main goal of active learning is to minimize the number of solved examples (labelled examples) needed to learn any task without appreciably degrading the performance. Thus, we see a fundamental difference between the active learning, and the CBR approach:

- Active learning tries to minimize the number of questions to the teacher, i.e. active learning tries to select which are the interesting examples *before* knowing their solution, to avoid the cost associated with labelling them (asking for the right solution from a teacher, or running the revise process in the case of CBR).
- CBR assumes that this solution is known, since the retain process is performed after the Revise process in the CBR cycle, and thus CBR case retention does not try to avoid the cost of labelling the cases.

In this section, we are going to present individual decision policies inspired on both approaches. Specifically, we are going to present a collaboration strategy that has two individual decision policies: the *Individual Case Retention* decision policy (D_{ICR}) and the *Case Offering* decision policy (D_O). The Individual Case Retention decision policy is in charge of deciding whether a new case will be retained individually or not, and the Case Offering decision policy is in charge of offering cases to other agents. In the following section we are going to present Individual Case Retention decision policies inspired both in CBR



p_1	:	$Request(?A_c, all, ?c)$
p_2	:	$Inform(?A_j, !A_c, true/false)$
p_2/c_1	:	$Inform(?A_j, !A_c, true/false)$
		$\|!W_1 w_1 A_j\| = !(n-1)$
p_3	:	$Inform(!A_c, ?A_j, !c)$
p_4	:	$Inform(!A_j, !A_c, ok)$
p_5	:	$Inform(!A_c, all, termination)$

Figure 8.2: Interaction protocol for the *Multi-agent Retention* collaboration strategy.

case retention and in active learning, and also several Case Offering decision policies. Then, we will analyze the resulting collaboration strategies of using several combinations of individual decision policies. Let us start by defining the collaboration strategy:

Definition 8.2.1. *The* Multi-agent Retention Collaboration Strategy *(MAR-CS) is a collaboration strategy* $\langle I_{MAR-CS}, D_{IR}, D_O \rangle$*, where* I_{MAR-CS} *is the MAR-CS interaction protocol shown in Figure 8.2,* D_{IR} *is the Individual Case Retention decision policy used to decide whether to retain cases individually or not, and* D_O *is the Case Offering decision policy used to decide when to offer cases to the other agents.*

Figure 8.2 shows the I_{MAR-CS}. However, before using the protocol an agent A_i using MAR-CS performs a series of actions: when an agent $A_i \in \mathcal{A}^c$ has the opportunity the retain a new case c and has to decide whether to retain c or not, A_i uses its own D_{IR} to decide whether to individually retain c or not. Then, the D_O is used to decide whether to offer c to other agent or not. If D_O decides that the case will not be offered, the protocol is not engaged, and the collaboration strategy ends. But if D_O decides that the case will be offered to other agents, the protocol I_{MAR-CS} is started.

The interaction protocol I_{MAR-CS} applies to a set of agents $\mathcal{A}^c$ and works as follows:

- In state w_0, A_i broadcasts the new case c to the rest of agents in $\mathcal{A}^c$ with message p_1.

- In state w_1, the agents that have received the case c use their Individual Case Retention decision policies D_{IR} to decide if they are interested on the case or not. Then, they answer to A_i expressing if they are interested in c or not with message p_2. Once every agent that has received c has answered A_i, the protocol moves to state w_2. Let $interested(c, \mathcal{A}^c) \subseteq \mathcal{A}^c$ be the set of agents interested on case c.

- In state w_2, A_i uses its Case Offering decision policy D_O to decide which agent or agents form the set of agents $\mathcal{A} \subseteq interested(c, \mathcal{A}^c)$ that will receive the case c. Then, A_i sends a message p_3 to each agent $A_j \in \mathcal{A}$ telling them that they can retain case c, and waits for their confirmation messages p_4. Once all the agents in $\mathcal{A}$ have answered, A_i broadcasts a termination message p_5 and the protocol goes to state w_4 that is a final state.

- In state w_3, the agents that receive the message p_3 retain the case c and once the case has been retained send a confirmation message p_4 to A_i.

- State w_4, is a final state and thus the protocol ends.

Notice that in state w_1 every agent could retain case c without permission of A_i, however, we are going to assume that the agents are collaborative, that the individual agents will follow the rules of the protocol, and only retain cases when they receive the permission to do so with message p_3. Moreover, in Section 8.2.5 we will show that agents in fact have a motivation to follow the protocol, since by doing so the performance of the committee increases.

In the following sections, we are going to present several Individual Case Retention decision policies inspired in CBR case retention in Section 8.2.1 and in active learning in Section 8.2.2. Then we present several Case Offering decision policies 8.2.3 and, finally, retention strategies will be presented in Section 8.2.4.

8.2.1 CBR Case Retention Decision Policies

In this section we are going to present three decision policies that an agent can use as its D_{IR} decision policy. All the decision policies presented in this section are in fact boolean predicates that return true when the new case should be retained and false otherwise.

Definition 8.2.2. *The* Never Retain *Individual Case Retention decision policy consists on never retaining any case and is defined as follows:*

$$D_{NR-IR}(c) = false$$

Definition 8.2.3. *The* Always Retain *Individual Case Retention decision policy consists on retaining every received case into the case base and is defined as follows:*

$$D_{AR-IR}(c) = true$$

Definition 8.2.4. *The* On-Failure Retain *Individual Case Retention decision policy consists on retaining only those cases that cannot be solved correctly using the current case base.* $D_{OFR-IR}(c)$ *is defined for an agent* A_i *as follows:*

$$D_{OFR-IR}(c) = \begin{cases} true & \text{If } c.S \neq Sol(\mathcal{S}, c.P, \{A_i\}), \\ false & \text{otherwise.} \end{cases}$$

where $Sol(\mathcal{S}, c.P, \{A_i\})$ *is the solution class predicted by* A_i *for the problem* $c.P$ *before retaining it (as defined by Equation 4.2).*

Notice that the first two decision policies presented in this section (Never Retain and Always Retain) are only defined for comparison purposes. Moreover, the interesting decision policy (On-Failure Retain) is inspired in the CBL2 technique presented in Section 2.2.1.1.

8.2.2 Active Learning Decision Policies

The main difference of the active learning decision policies with the CBR case retention policies is that the active learning ones take the decision of retaining or not new cases without having access to the correct solution of them.

The idea of the *example selection* in active learning is to identify a region of uncertainty inside the problem space that corresponds to the region of the problem space where the learner (in our framework the CBR agent) cannot provide predictions with a high confidence (as we explained in Section 2.2.1.1). Once the region of uncertainty is defined, it is recommended to retain only cases belonging to that region. QbC is one of those example selection techniques that uses a querying mechanism to determine the region of uncertainty (as we explained in Section 2.2.1.2).

In this section we are going to present a decision policy called *Informative Disagreement* Individual Case Retention (D_{ID-IR}), inspired in the Query by Committee (QbC) modification proposed by Argamon-Engelson and Dagan [7]. The main difference is that Informative Disagreement focus in measuring disagreement in a committee of agents. The main steps of the Informative Disagreement decision policy (for an agent A_i that has received a case c to retain) are the following ones:

1. A_i convenes a committee of agents $\mathcal{A}^c$.
2. Each agent in $\mathcal{A}^c$ individually solves the problem and sends its individual prediction to A_i.
3. A_i then measures the *agreement* among the individual agents' predictions: if most of the agents agree on the solution class of the new case c, it is assumed that the new case does not fall in the region of uncertainty, and therefore it is not retained. However, if there is a large amount of disagreement among the agents (i.e. most of them predict different solution classes), the case is considered to be in the region of uncertainty, and it is retained.

Notice that the idea behind Informative Disagreement is that those cases for which there is a large amount of agreement are considered cases that the committee already can solve with a high confidence, and therefore are non-interesting cases that are not worth being retained. However, those cases for which there is a high disagreement are cases with low confidence of being solved correctly, and thus they can contribute to improve the performance of the committee.

Let us first provide a measure of the disagreement among the predictions made by a set of agents:

$$Diagreement(c, \mathcal{A}^c) = \frac{V_r}{(K-1) * V_w} \tag{8.1}$$

where:

- K is the number of solution classes,
- $V_w = Ballot(Sol(\mathcal{S}, \mathcal{A}^c), \mathcal{A}^c)$ is the sum of votes for the most voted solution class,
- $V_r = \sum_{S_k \in \{\mathcal{S} - Sol(\mathcal{S}, \mathcal{A}^c)\}} Ballot(S_k, \mathcal{A}^c)$ is the sum of votes for the rest of solution classes.

In other words, the disagreement is measured dividing the votes for the least voted solutions by the votes for the most voted solution (and applying a normalizing factor $K-1$). Notice that when all the agents vote for the same solution class the disagreement is 0 and if each agent votes for a different solution class the disagreement is 1.

Using this disagreement measure we can define now the Informative Disagreement decision policy as a threshold based procedure:

Definition 8.2.5. *The* Informative Disagreement *Individual Case Retention decision policy consists on retaining only those cases for which the disagreement about their correct solution class is above a threshold d_0:*

$$D_{ID-IR}(c, \mathcal{A}^c) = (Disagreement(c, \mathcal{A}^c) > d_0)$$

Notice that in order to use the Informative Disagreement decision policy, an agent has to convene a committee (for instance using the Committee collaboration strategy) before deciding whether to retain the case or not.

Finally, notice that once a high disagreement is detected (and thus the case is considered interesting) we can distinguish between two groups of agents in the committee $\mathcal{A}^c$: those that had solved c correctly and those that have not. Clearly, the agents that have not solved c correctly are those that have a higher need of retaining c. We will use this distinction later when we combine this decision policy with other Case Offering decision policies.

8.2.3 Case Offering Decision Policies

In this section we are going to define three basic decision policies that the agents can use as their D_O decision policy.

Notice that the D_O is a complex policy that consists of two parts: first, D_O is used to decide whether to initiate the I_{MAR-CS} protocol, and then it is used in state w_2 of the same protocol to decide which agents are going to retain the case at hand. All the decision policies defined in this section will consist on two parts (D_O^1 and D_O^2), the first part is defined as a boolean predicate, and the second one is a function that returns the set of agents that will retain the case c.

Definition 8.2.6. *The* Never-Offer *Case Offering decision policy consists of never offering any case to the rest of agents and is defined as follows:*

$$\begin{array}{l} D^1_{NO-O}(c, \mathcal{A}^c) = false \\ D^2_{NO-O}(c, \mathcal{A}^c) = \emptyset \end{array} \tag{8.2}$$

Definition 8.2.7. *The* Always-Offer *Case Offering decision policy consists on always offering those cases that the agent has decided not to retain individually to the rest of agents, but only allowing a single agent to retain one case (in order to not increase the redundancy in the system). The first part is defined as follows:*

$$D^1_{AO-O}(c, \mathcal{A}^c) = \neg D_{ICR}(c, \mathcal{A}^c)$$

i.e. only offer a case when it os not individually retained. The second part is defined as follows:

$$D^2_{AO-O}(c, \mathcal{A}^c) = \begin{cases} \emptyset & \text{If } interested(c, \mathcal{A}^c) = \emptyset, \\ random(interested(c, \mathcal{A}^c)) & \text{otherwise.} \end{cases}$$

where $interested(c, \mathcal{A}^c)$ is the set of agents that answered in state w_1 of the protocol saying that they were interested on retaining the case c. Moreover, $random(interested(c, \mathcal{A}^c))$ is a function that returns a set consisting on a single agent randomly selected from the set of interested agents.

Definition 8.2.8. *The* Always-Offer-Copy *Case Offering decision policy consists of always offering the received cases to the rest of agents and allowing as many agents as are interested to retain the case. It is defined as follows:*

$$\begin{array}{l} D^1_{AOC-O}(c, \mathcal{A}^c) = true \\ D^2_{AOC-O}(c, \mathcal{A}^c) = interested(c, \mathcal{A}^c) \end{array} \tag{8.3}$$

Notice that the Never-Offer decision policy has only been defined for comparison purposes, while the Always-Offer and the Always-Offer-Copy decision policies are more interesting. The main difference between them is that in Always-Offer, the Committee Redundancy (see Section 4.4.2) will not increase,

while in the Always-Offer-Copy decision policy, the Committee Redundancy will increase since more than one agent in the committee is allowed to retain the same case.

In the following section we will analyze the possible combinations of the Individual Case Retention decision policies with the Case Offering decision policies.

8.2.4 Case Retention Strategies

In the previous sections we have defined four different Individual Case Retention decision policies (D_{NR-IR}, D_{AR-IR}, D_{OFR-IR} and D_{ID-IR}) and three different Case Offering decision policies (D_{NO-O}, D_{O-O} and D_{OC-O}). In order to use MAR-CS, an agent requires one decision policy of each kind. Thus, we have 12 possible combinations (although not all of them make sense). In this section we are going to present all the interesting combinations (those that will be tested in the experiments section).

- *Never Retain - Never Offer* (NR-NO): This is the MAR-CS collaboration strategy using D_{NR-IR} and D_{NO-O} as individual decision policies. No cases are retained nor offered, thus a $\mathcal{MAC}$ system composed of agents using this collaboration strategy is equivalent to a $\mathcal{MAC}$ system where the agents have no learning capabilities. Moreover, notice that in order to be coherent in notation we should call this collaboration strategy MAR-CS$_{NR-NO}$, but we will simply refer to it as NR-NO for short; the same comment applies to the rest of versions of MAR-CS presented in this section.

- *Always Retain - Never Offer* (AR-NO): uses the D_{AR-IR} and D_{NO-O} individual decision policies. This collaboration strategy is equivalent to agents working individually, since they retain all the cases they receive and never offer them to the rest of agents.

- *On-Failure Retain - Never Offer* (OFR-NO): this collaboration strategy uses the D_{OFR-IR} and D_{NO-O} individual decision policies. This collaboration strategy is also equivalent to agents working individually (since they retain all the cases they receive and never offer them to the rest of agents) but where the individual agents only retain those cases that they cannot solve correctly.

- *On-Failure Retain - Always Offer* (OFR-AO): uses the D_{OFR-IR} and D_{O-O} individual decision policies. In this collaboration strategy agents individually retain the cases that they cannot correctly solve, and the rest of cases are offered to the other agents. In some of the other agents are interested in the case, one of them will retain it.

- *Informative Disagreement - Never Offer* (ID-NO): uses the D_{ID-IR} and D_{NO-O} individual decision policies. This collaboration strategy is equivalent to agents working individually (since they retain all the cases they

receive and never offer them to the rest of agents) where the individual agents actively select which cases to retain using the Informative Disagreement decision policy (i.e. convening a committee and measuring the degree of disagreement among the individual predictions). Notice that this is a selfish collaboration strategy, since a committee is convened in order to obtain the information needed to decide retention, but after that only the convener agent has the option to actually retain the case.

- *Informative Disagreement - Always Offer* (ID-AO): uses the D_{ID-IR} and D_{AO-O} individual decision policies. In this collaboration strategy D_{ID-IR} is used to decide wether c is interesting or not: if the case is considered not interesting, it is not retained nor offered to any other agent (since the other agents will also consider it non interesting if they also use D_{ID-IR}); if the case is considered interesting to be retained, A_i will only retain c if A_i is one of the agents that have failed to solve c correctly in the committee convened to measure the disagreement. Otherwise, one of the agents of the committee that have failed to solve it correctly (and there must be at least one since there has been disagreement) will retain it.

- *On-Failure Retain - Always Offer Copy* (OFR-AOC): uses the D_{OFR-IR} and D_{AOC-O} individual decision policies. In this collaboration strategy agents individually retain the cases that they cannot correctly solve. Then, regardless of whether the case has been retained or not, c is offered to the rest of agents, and all the agents interested in the case will retain a copy of c.

- *Informative Disagreement - Always Offer Copy* (ID-AOC): uses the D_{ID-IR} and D_{AOC-O} individual decision policies. In this collaboration strategy D_{ID-IR} is used to decide whether c is interesting or not: if the case is considered not interesting, c is not retained nor offered to any other agent (since the other agents will also consider it non interesting if they also use D_{ID-IR}); if the case is considered interesting to be retained, all the agents that have failed to solve c correctly in the committee convened to measure the disagreement will retain a copy of c.

Moreover, notice that there are some combinations of decision policies not listed above. For instance, we haven't defined *Always Retain - Offer*, since it is equivalent to *Always Retain - No Offer*; we haven't defined *Always Retain - Offer Copy* either, since with that combination every agent will retain every case and they will all end with identical case bases (if we assume that every agent in the $\mathcal{M}$AC system uses the same collaboration strategy). Finally, we haven't defined any collaboration strategy using *Never Retain* with *Offer* or *Offer Copy* since they will be equivalent to NR-NO (if we make the same assumption as before).

8.2.5 Experimental Evaluation

In this section we are going to empirically evaluate the versions of the MAR-CS collaboration strategies. Specifically, we are going to compare the classification accuracy of agents solving problems individually and using the Committee collaboration strategy for a $\mathcal{M}$AC system composed of 5 agents. Moreover, we are also going to discuss results concerning the characteristics of case bases achieved using each one of the different collaboration strategies (including case base size, completeness, redundancy, and bias).

In order to test the generality of the collaboration strategies we are going to present results using three data sets: sponge, soybean and zoology. Moreover, in all the experiments reported in this section agents use LID as the learning method and all the presented results are the average of five 10-fold cross validation runs.

In an experimental run the data set is divided into two sets: the training set, containing the 90% of the cases in the data set and the test set, containing the remaining 10% of the cases in the data set. At the beginning of the experiment, a 10% of the cases of the training set are distributed among the agents. Then, the rest of the cases in the training set are sent to the agents one by one (each case is only sent to one agent). Each time an agent receives a case of the training set, it applied its retention strategy to decide retention. From time to time, the test set is solved by the agents to evaluate their classification accuracy. The experiment ends when all the cases in the training set have been sent to the agents. Moreover, in order to test the retention strategies, we have set up the worst scenario for a retention strategy: the cases in the training set are not randomly sent to the agents but are sent with a high degree of bias, i.e. some agents have a very high probability of receiving cases of certain classes and a very small probability of receiving cases of other classes. Thus, if the individual agents simply retain all the cases they receive without performing any filtering or collaboration, they will end up with a very biased case base. Specifically we have induced a Case Base Bias of $\mathbb{B} = 0.45$ for the sponge data set, $\mathbb{B} = 0.4$ for the zoology data set, and $\mathbb{B} = 0.24$ for the soybean data set, that are very high biases for those data sets (see Section 4.5.3). Moreover, we have set the threshold $d_0 = 0.1$ (used in the ID-IR decision policy) for the sponge data set, $d_0 = 0.04$ for the zoology data set, and $d_0 = 0.03$ for the soybean data set. Those values have been chosen as appropriate values for each data set: Since the number of solution classes K appears in the divisor of the disagreement formula (see Equation 8.1) committees working in data sets with a lower number of possible solution classes tend to have higher values of disagreement while committees working in data sets with a higher number of solution classes tend to have lower values of disagreement. Therefore, we have set a high threshold value for d_0 in the sponge data set that has only three solution classes, and a low value of d_0 for the soybean and zoology data sets that have a larger number of solution classes.

We will first present experimental results concerning classification accuracy, and after that we will analyze the obtained case bases with the different retention

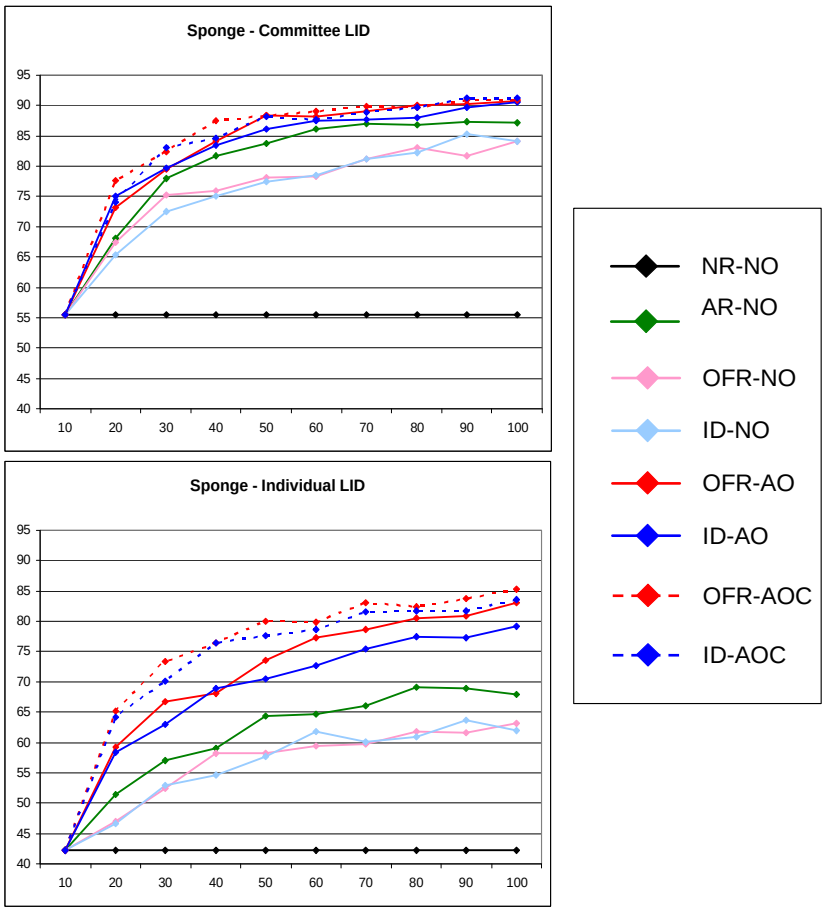


Figure 8.3: Classification accuracy comparison between agents using the different MAR-CS collaboration strategies in the sponge data set.

strategies.

8.2.5.1 Classification Accuracy Evaluation

Figure 8.3 shows the classification accuracy evolution of the agents in a $\mathcal{MAC}$ system composed of 5 agents using the different retention strategies in the sponge data set. Both individual and committee classification accuracy is shown: the right hand side plot shows the individual classification accuracy and the left hand side plot shows the committee classification accuracy. For each collaboration strategy, a plot is shown, representing the evolution of classification accuracy in the test set as the cases in the training set are sent to the agents (the horizontal axis represents the percentage of cases of the training set that have been sent to the agents). From the eight collaboration strategies shown,

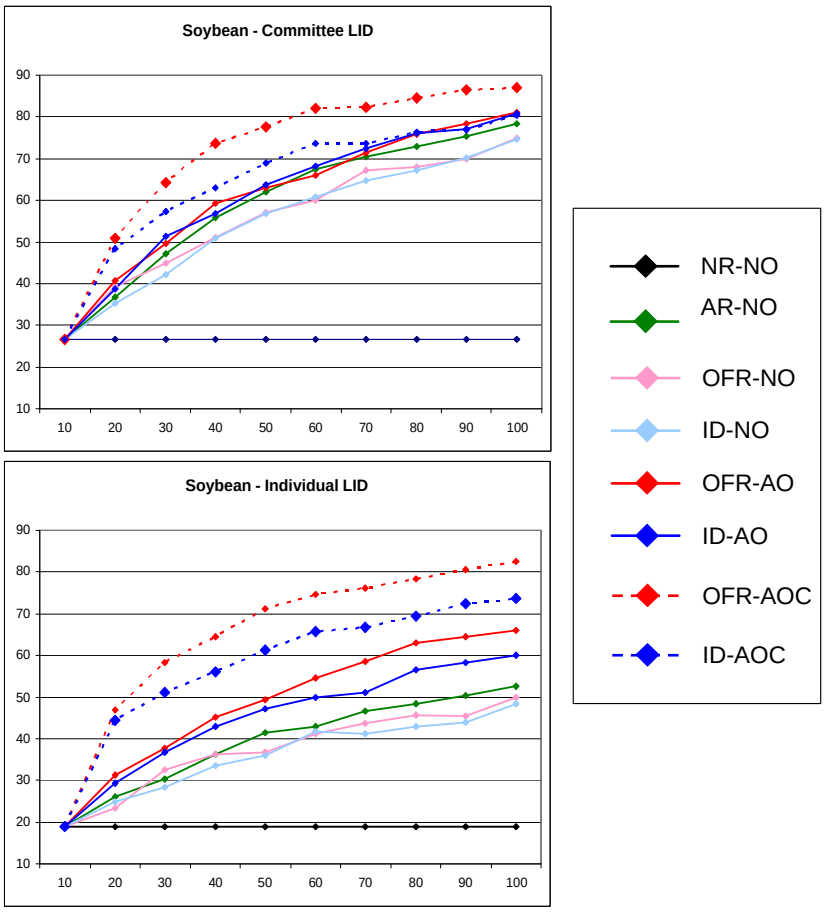


Figure 8.4: Classification accuracy comparison between agents using the different MAR-CS collaboration strategies in the sponge data set.

two of them should be considered as the baselines for comparison, namely, NR-NO and AR-NO. NR-NO corresponds to a CBR agent that does not perform case retention (i.e. it only has in its case base the cases that has received from the initial 10% of the training set), and AR-NO corresponds to a CBR agent that simply retains every case it receives.

The main result that Figure 8.3 shows is that strategies that offer cases to other agents (OFR-AO, ID-AO, OFR-AOC, and ID-AOC) clearly outperform any of the strategies that do not offer cases to the other agents (OFR-NO, ID-NO, AR-NO, and NR-NO). The right hand side plot in Figure 8.3 shows us that agents using retention strategies that offer cases achieve a classification accuracy in the range of the 80%-85%, while agents using non offering retention strategies achieve classification accuracy values below the 70%. In the sponge

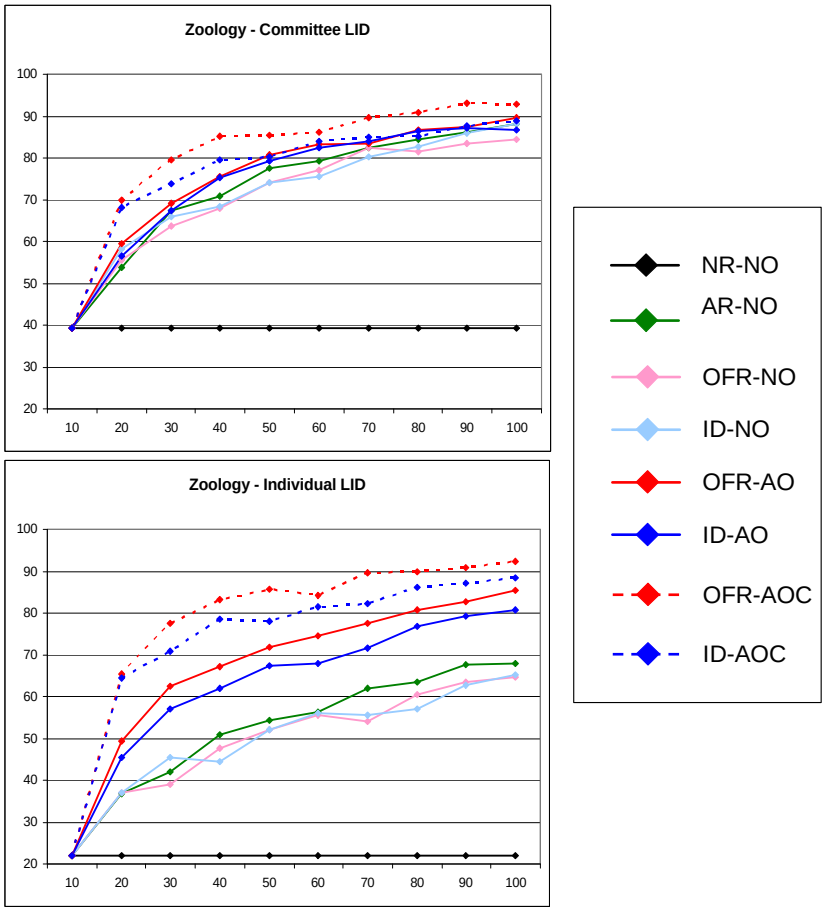


Figure 8.5: Classification accuracy comparison between agents using the different MAR-CS collaboration strategies in the sponge data set.

data set, the retention strategies that achieved the lowest classification accuracy are OFR-NO and ID-NO, with an individual classification accuracy of 63.14% and 62.00% respectively and a committee classification accuracy of 84.07% for both. The best strategy is not very clear, considering the individual classification accuracy, OFR-AOC and ID-AOC achieved the highest classification accuracy values (85.21% and 83.64% respectively), but considering the committee classification accuracy, all the retention strategies that offer cases to other agents achieve very similar classification accuracy (above 90%), being ID-AOC the strategy that achieved the highest accuracy (91.29%) by a small difference. In order to further compare the results of the different collaboration strategies we have to look at the size of the case bases (as the next section does).

Figure 8.4 shows the classification accuracy evolution of the agents in the

	Individual	CCS	CB size	$\mathbb{C}$	$\mathbb{R}$	$\mathbb{B}$
NR-NO	42.29%	55.57%	5.00	0.10	0.00	0.45
AR-NO	68.00%	87.21%	50.4	1.00	0.00	0.45
OFR-NO	63.14%	84.07%	18.28	0.36	0.00	0.43
ID-NO	62.00%	84.07%	17.78	0.35	0.00	0.44
OFR-AO	83.07%	90.71%	34.24	0.68	0.00	0.17
ID-AO	79.14%	90.50%	29.20	0.58	0.00	0.19
OFR-AOC	85.21%	90.93%	55.94	0.56	0.25	0.13
ID-AOC	83.64%	91.29%	47.36	0.45	0.27	0.14

Table 8.1: Case Base properties comparison between agents using the different MAR-CS collaboration strategies in the sponge data set.

soybean data set. In the soybean data set we can observe again that all the collaboration strategies that offer cases to other agents achieve higher classification accuracy values than the non offering collaboration strategies. Moreover, in the soybean data set we can see that OFR-AOC achieved clearly the highest classification accuracy values, achieving 82.61% for individual agents and 86.97% for the committee. The other three offering strategies (ID-AOC, OFR-OC, and ID-OC) achieved lower classification accuracies than OFR-AOC, but higher than any of the non offering strategies. Moreover, in the soybean data set, we can clearly see that strategies based on D_{OFR-IR} clearly achieved higher accuracy values than strategies based on D_{ID-IR} (this wasn't so clear in the sponge data set).

Figure 8.5 shows the classification accuracy evolution of the agents in the zoology data set. Results for the zoology data set are very similar to those obtained in the soybean and sponge data sets: offering strategies achieve higher accuracy values than non offering strategies (except ID-AO in that achieves a slightly lower committee accuracy than AR-NO and that ID-NO), and strategies based on D_{OFR-IR} achieve usually higher classification accuracy values than strategies based on D_{ID-IR}(except OFR-NO that achieves lower classification accuracy than ID-NO). The strategy that achieved the highest classification accuracy is again OFR-AOC.

Summarizing, the conclusions that we can draw from the classification accuracy results are: offering strategies achieved higher classification accuracy than non offering strategies; strategies that allow copies of cases (based on D_{AOC-O}) achieve higher individual classification accuracy values; and in the soybean and zoology data sets strategies based on D_{OFR-IR} achieve higher classification accuracy values. Let us now compare the properties of the case bases achieved by the CBR agents after using the different collaboration strategies.

8.2.5.2 Case Base Evaluation

Table 8.1 shows the case base properties of the CBR agents at the end of the experimental runs for the sponge data set. Each row of the table shows the

results for a different collaboration strategy. The columns correspond to the following properties of the case base: the accuracy achieved by individual agents, the accuracy achieved by the committee of agents (using CCS), the average size of individual case base, and the average Committee Completeness ($\mathbb{C}$), Committee Redundancy ($\mathbb{R}$) and Committee Bias ($\mathbb{B}$).

Let us first consider case base size. As we have seen in the previous section, the offering collaboration strategies (OFR-AO, ID-AO, OFR-AOC, and ID-AOC) achieve higher classification accuracy values than the non offering collaboration strategies; Table 8.1 shows that offering strategies end with larger case bases. For instance, an agent using OFR-NO ends with a case base with an average of 18.28 cases, while an agent using OFR-AO ends with an average of 34.24 cases. The only non offering collaboration strategy that ends with large case bases is AR-NO (that retains all the cases that the agent receives). Comparing the non offering collaboration strategies, AR-NO achieves the highest accuracy, but with the largest case bases, while OFR-NO and ID-NO achieve lower accuracy values, but with about one third of the cases than AR-NO. Comparing now the offering collaboration strategies, we see that OFR-AOC and ID-AOC achieve the highest classification accuracy values, but with very large case bases (55.94 and 47.36 respectively). In contrast, OFR-AO and ID-AO achieve nearly the same accuracy values than OFR-AOC and ID-AOC with smaller case bases (34.24 and 29.20 cases respectively). Moreover, we can see that strategies based on D_{ID-IR} retain less cases in general than strategies based in D_{OFR-IR} while achieving the same classification accuracy values. Thus, in the sponge data set, we can conclude that the best collaboration strategy is either ID-AOC if we only care about classification accuracy, or ID-AO if we consider also the size of the individual case bases.

Let us analyze now the collaboration strategies in terms of the *ensemble space* (see Section 4.4). In general terms, Table 8.1 shows that offering strategies are able to decrease bias and to increase case base completeness (not taking AR-NO into account), both good effects. Moreover, strategies that allow copies of cases increase redundancy (that, as we saw in Chapter 4, is good if redundancy is not increased too much) and decrease bias even more. Concerning the individual classification accuracy, only case base completeness and case base bias are important. As we said in Section 4.4, the best individual case bases are those with a high completeness and a low bias; Table 8.1 shows that the results completely confirm that statement: for instance, OFR-AOC and ID-AOC achieve the lowest case base biases and high case base completeness, thus they achieve the highest individual classification accuracy values. AR-NO achieves the maximum completeness (1.00), but also with a large case base bias (0.45), and thus does not reach high accuracy levels. Concerning committee classification accuracy we have to take redundancy into account. OFR-AOC and ID-AOC achieve the maximum individual classification accuracy values (having accuracy values larger than OFR-AO and ID-AO respectively), but they achieve almost the same committee accuracy values than OFR-AO and ID-AO respectively. The reason is that, although OFR-AOC and ID-AOC increase committee redundancy (that is good till a certain degree), they reduce committee completeness; these two

	Individual	CCS	CB size	$\mathbb{C}$	$\mathbb{R}$	$\mathbb{B}$
NR-NO	19.02%	26.58%	5.40	0.10	0.00	0.25
AR-NO	52.70%	78.34%	55.26	1.00	0.00	0.25
OFR-NO	49.77%	74.88%	28.99	0.52	0.00	0.22
ID-NO	48.27%	74.66%	28.20	0.51	0.00	0.22
OFR-AO	69.38%	81.04%	48.05	0.87	0.00	0.12
ID-AO	60.00%	80.85%	40.12	0.73	0.00	0.17
OFR-AOC	82.61%	86.97%	85.94	0.62	0.38	0.10
ID-AOC	73.55%	80.39%	60.90	0.39	0.46	0.11

Table 8.2: Case Base properties comparison between agents using the different MAR-CS collaboration strategies in the soybean data set.

	Individual	CCS	CB size	$\mathbb{C}$	$\mathbb{R}$	$\mathbb{B}$
NR-NO	21.98%	39.21%	1.80	0.10	0.00	0.40
AR-NO	67.92%	88.12%	18.18	1.00	0.00	0.40
OFR-NO	64.75%	84.36%	7.13	0.39	0.00	0.38
ID-NO	65.15%	88.12%	7.16	0.39	0.00	0.40
OFR-AO	85.35%	89.61%	10.86	0.60	0.00	0.23
ID-AO	80,79%	86.73%	9.76	0.54	0.00	0.26
OFR-AOC	92.28%	92.87%	16.51	0.37	0.37	0.24
ID-AOC	88.32%	88.91%	13.67	0.31	0.36	0.23

Table 8.3: Case Base properties comparison between agents using the different MAR-CS collaboration strategies the zoology data set.

effects cancel each other and the classification accuracy does not increase much from OFR-AO to OFR-AOC or from ID-AO to ID-AOC.

Table 8.2 shows the case base properties of the CBR agents at the end of the experimental runs for the soybean data set. The analysis that we can make is similar to that made in the sponge data set. However, recall that in the soybean data set (unlike the sponge data set), OFR-AOC outperforms the rest of collaboration strategies; this can be explained by looking at Table 8.2: OFR-AOC is the collaboration strategy that has achieves the lowest bias, and has achieved large completeness and low redundancy compared to ID-AOC. Therefore, OFR-AOC is the collaboration strategy that has achieved a point inside the ensemble space that is better than the points achieved by the other collaboration strategies.

Finally Table 8.3 shows the case base properties of the CBR agents at the end of the experimental runs for the zoology data set showing similar results than in the sponge and soybean data sets: offering strategies reduce bias and increase completeness and strategies that allow copies of cases increase redundancy.

Summarizing all the experimental results presented in this section, the main conclusion is that collaborating with other agents can greatly improve case retention (since offering strategies outperform non offering strategies). Moreover,

we can also draw other conclusions: allowing multiple copies of cases during retention (as strategies based on the D_{AOC-O} decision policy do) increases Committee Redundancy and decreases Committee Completeness, there fore they are not always better than collaboration strategies that do not allow multiple copies of cases (multiple copies has been better in the soybean and zoology data sets, but not in the sponge data set). It remains as future work to explore collaboration strategies that try to keep a controlled degree of Committee Redundancy. Moreover, there is also no clear winner between strategies based on D_{ID-IR} and strategies based on D_{OFR-IR}: D_{ID-IR} has worked better in the sponge data set and D_{OFR-IR} has worked better in soybean and zoology data sets.

8.3 Justification-based Case Reduction

In this section we are going to present JUST(*Justification-base Selection of Training Examples*), a case base reduction method that uses justifications in order to assess the utility of the cases in a case base. JUST is not a collaboration strategy, but an individual method that any agent in a $\mathcal{M}$AC system can use in order to reduce the size of its case base while maintaining the same level of classification accuracy. Thus, JUST can be applied not only to CBR agents but to any CBR system. Therefore, in the remainder of this section we will talk about a CBR system and not about CBR agents.

The novelty of JUST resides in using justifications to assess the utility of cases. Specifically, JUST uses a case utility function called *Justification-based Case Utility* (JCU) to select which are the cases that should be kept in the case base and which are the cases that can be discarded.

In this section we are first going to present how justifications can be used to assess the utility of a case for a CBR system, thus Section 8.3.1 will present the JCU case utility function, while JUST will be presented in Section 8.3.2.

8.3.1 Justification-based Case Utility

The *Justification-based Case Utility* (JCU) is a function that uses justifications to estimate the utility of a set of cases C_A that are being considered to be added into the case base. JCU tries to predict if the set of cases C_A can reduce the number of errors that the CBR system will make in the future. The more errors that the cases in C_A can prevent, the highest their JCU utility. Moreover, notice that JCU considers all the cases in the set C_A as a whole, so it does not evaluate the individual utility of the cases in C_A, but the overall utility for a specific set of cases.

JCU requires two sets of cases: C_A and C_E. C_A is the set of cases for which JCU is going to measure the utility, and C_E is a set of cases that are not present in the case base and that we will call the *examination cases*. The utility value returned by JCU is an estimation of how many of the errors that the CBR system will make while solving the problems in C_E could have been prevented if the cases in C_A would have been in the case base. Notice that if C_E is a good

sample of the problem space, then JCU is a good estimator of the utility of the set of cases C_A.

Before defining JCU, let us introduce the notion of *valid counterexample* that refines the definition of a counterexample given in the previous chapter:

Definition 8.3.1. *The* set of valid counterexamples *of a JER* $\mathbf{J}$ *from a set of cases* C_A *is* $VCE(\mathbf{J}, C_A) = \{c \in C_A | \mathbf{J}.J \sqsubseteq c.P \wedge c.S \neq \mathbf{J}.S \wedge c.S = S_{\mathbf{J}.P}\}$*, i.e. the set of cases* $c \in C_A$ *subsumed by the justification* $\mathbf{J}.J$ *that do not belong to the predicted solution class* $\mathbf{J}.S$ *and that belong to the correct solution class* $S_{\mathbf{J}.P}$ *of the problem* $\mathbf{J}.P$ *for which the JER* $\mathbf{J}$ *was constructed.*

Notice that to build the set of valid counterexamples of a JER $\mathbf{J}$, the correct solution class of the problem $\mathbf{J}.P$ for which the JER has been built must be known.

Let us define how JCU works for a CBR system that has a case base C_i and wants to evaluate the utility of a set of cases C_A using the *examination cases* in C_E. Let us call the *exam* $E = \{c.P | c \in C_E\}$ the set of problems in the examination cases. In order to compute the utility, the CBR system proceeds as follows:

1. A_i solves all the problems in E, providing a justified prediction for each one expressed as a set of JERs. Thus, A_i builds the set $\mathbf{J}_E = \{\mathbf{J} | \mathbf{J}.P \in E\}$ containing the JERs built for solving all the problems in the exam.

2. Since the exam E has been constructed using the problems of the cases in C_E, A_i knows which is the correct solution for each problem. Thus, A_i can determine which problems in E have been solved correctly and which not. Let $\mathbf{J}_E^- = \{\mathbf{J} \in \mathbf{J}_E | \mathbf{J}.S \neq S_{\mathbf{J}.P}\}$ be the set of JERs that A_i has built for the incorrectly solved problems in E (where $S_{\mathbf{J}.P}$ represents the correct solution for the problem $\mathbf{J}.P$).

3. Next, A_i counts how many JERs in $\mathbf{J}_E^-$ have at least one valid counterexample (see Section 7.2.3) among the cases in C_A. We will call this number n_E, and represents the predicted number of errors that the cases in C_A would have prevented if they would have been present in the case base while solving the problems in the exam E, i.e. $n_E = \#(\{\mathbf{J} \in \mathbf{J}_E^- | VCE(\mathbf{J}, C_A) \neq \emptyset\})$.

4. The JCU utility value for the cases in C_A computed with respect to the examination cases C_E is:

$$JCU(C_A, C_E) = \frac{n_E}{\#(C_E)}$$

Notice that step 3 is the key step in JCU. The idea behind n_E is that if a case c is a counterexample of a JER $\mathbf{J}$ and it is present in the case base of the CBR system, it is very unlikely that the system will provide a justification for which there is a counterexample in its own case base. Therefore, JCU assumes

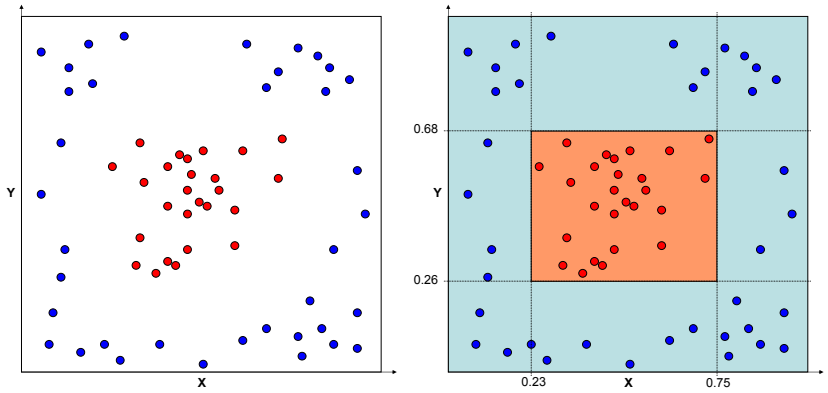


Figure 8.6: Artificial data set to show the behavior of JCU.

that by adding a counterexample of a JER **J** to the case base of the system, the system will not provide the same incorrect justification again. Moreover, JCU requires that the counterexample belongs to the correct solution class of the problem $\mathbf{J}.P$ for which the incorrect JER was generated (i.e. that it is a valid counterexample). This assumes that, if the case c is added to the case base, it is likely that the next time the system has to solve a problem similar to $\mathbf{J}.P$, it will provide a correct justified prediction.

Finally, JCU returns the estimated number of errors that the set of cases C_A can prevent normalized by the number of examination cases to ensure that the utility value is in the interval $[0, 1]$. If the cases in C_A are predicted to fix no errors, JCU will predict an utility value of 0; and if the system has incorrectly solved all the problems in the exam, and the cases in C_A can make that the system can solve all the problems correctly if they are added to the case base, then the predicted utility value is 1. As a final remark, remember that JCU computes the utility of the set of cases C_A as a whole, and not of the individual cases in C_A. However, the utility of a single case can also be computed by creating a set of cases C_A that contains only a single case.

8.3.1.1 Exemplification

In this section we will present an exemplification in order to illustrate how JCU works. The left hand side of Figure 8.6 shows the set of cases that are present in the case base of a CBR agent A_i for a domain where the cases have only two real valued attributes (X and Y), and in which cases can have two possible solution classes: *red* or *blue* (in this example we talk about a CBR agent, but recall that JCU is applicable to any CBR system). On the right hand side of Figure 8.6, we show the model that A_i has built of the data. In this example, the CBR agent uses a decision tree to index the cases in the case base (as in the example presented in Section 7.2), and right hand of Figure 8.6 shows the

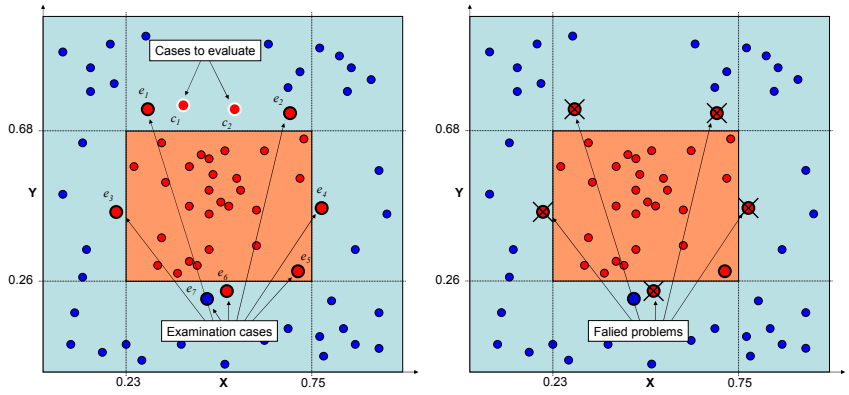


Figure 8.7: Cases $C_A = \{c_1, c_2\}$ are going to be evaluated using the set of examination cases shown. On the right hand side, the incorrectly solved problems of the exam are crossed.

different areas in which the learnt decision tree partitions the problem space.

A_i is interested on evaluating the utility of a set of cases $C_A = \{c_1, c_2\}$, shown in the left hand side of Figure 8.7 surrounded by a thick white line. In order to evaluate the utility, A_i will use JCU using the set of examination cases shown in the figure surrounded by a thick black line $C_E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$.

The first step in JCU consists on solving each one of the problems in the exam $E = \{e_1.P, e_2.P, e_3.P, e_4.P, e_5.P, e_6.P, e_7.P\}$, providing a justified prediction for each problem expressed as a set of JERs. A_i does that obtaining the corresponding JERs $\mathbf{J}_E = \{\mathbf{J}_1, \mathbf{J}_2, \mathbf{J}_3, \mathbf{J}_4, \mathbf{J}_5, \mathbf{J}_6, \mathbf{J}_7\}$.

The second step consists on finding the JERs for those problems incorrectly solved. As the right hand side of Figure 8.7 shows, 5 problems have not been solved correctly. Specifically, the 5 incorrect JERs are: $\mathbf{J}_E^- = \{\mathbf{J}_1, \mathbf{J}_2, \mathbf{J}_3, \mathbf{J}_4, \mathbf{J}_6\}$ (recall that a JER is a record containing a solution class, a justification, a problem and the agent who has generated it):

- $\mathbf{J}_1 = \langle blue, \langle X > 0.23 \wedge Y > 0.68 \rangle, e_1.P, A_i \rangle$
- $\mathbf{J}_2 = \langle blue, \langle X > 0.23 \wedge Y > 0.68 \rangle, e_2.P, A_i \rangle$
- $\mathbf{J}_3 = \langle blue, \langle X \leq 0.24 \rangle, e_3.P, A_i \rangle$
- $\mathbf{J}_4 = \langle blue, \langle X > 0.75 \wedge Y \leq 0.68 \rangle, e_4.P, A_i \rangle$
- $\mathbf{J}_6 = \langle blue, \langle X > 0.23 \wedge X \leq 0.75 \wedge Y \leq 0.26 \rangle, e_5.P, A_i \rangle$

The next step in JCU is counting how many incorrect JERs have at least one valid counterexample among the cases in C_A. As the left hand side of Figure 8.8 shows, both c_1 and c_2 are counterexamples of $\mathbf{J}_1$ and $\mathbf{J}_2$ since they satisfy the justification $\langle X > 0.23 \wedge Y > 0.68 \rangle$ and do not belong to the predicted solution

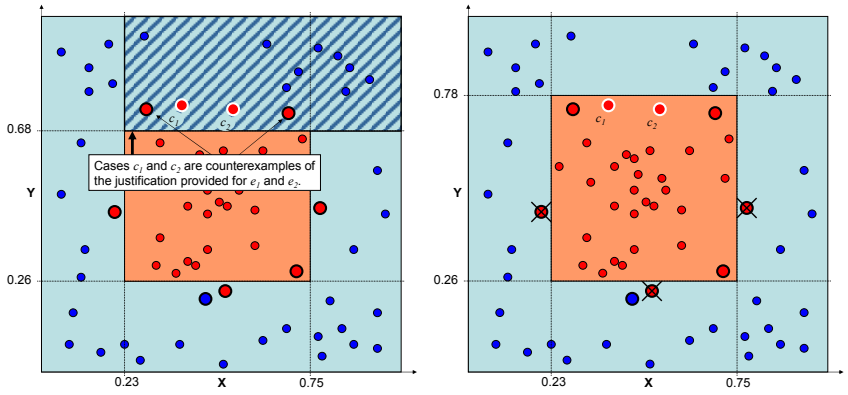


Figure 8.8: Cases $C = \{c_1, c_2\}$ are counterexamples of the justification given for two exam problems. On the right and side we show that effectively if cases in C are added to the case base, those two problems are now correctly solved.

class *blue*. Moreover, since c_1 and c_2 have *red* as their solution class (that is precisely the solution class of cases e_1 and e_2) they are valid counterexamples. Therefore $n_E = 2$, and A_i can compute the JCU utility value of C_A:

$$JCU(C_A, C_E) = \frac{2}{7} = 0.2857$$

Finally, the right hand side of Figure 8.8 shows the effect of adding c_1 and c_2 to the case base of A_i. Notice that since c_1 and c_2 are valid counterexamples of the JERs built for $e_1.P$ and $e_2.P$, JCU predicts that by adding c_1 and c_2, both problems ($e_1.P$ and $e_2.P$) would have been solved correctly. Figure 8.8 confirms that: notice that by adding c_1 and c_2 to the case base, the red area has grown towards the positive Y axis direction covering now both problems $e_1.P$ and $e_2.P$ (shown with a wide white line surrounding them). Thus, if the CBR agent tries to solve again $e_1.P$ and $e_2.P$, the predicted class will be *red*, that is the correct one.

Notice that JCU tries to predict how the classification accuracy of the case base will improve when adding new cases into the case base. An alternative to use JCU could be the following one: solve the exam using the current case base C_i, then add the new cases C_A into the case base and solve the problem again; comparing the results obtained with and without the cases in C_A, the CBR agent can measure if the cases in C_A can prevent any errors to be made in the exam. However, if an agent A_i has to compare the utility of a collection of sets of cases to decide which of them has a higher utility, the problems in the exam would have to be solved for each different set of cases in order to evaluate which of them is can prevent more errors. Using JCU we can compute the utility of several sets of cases solving only once the problems in the exam. This is a great advantage, and two of the strategies presented in this chapter

(JUST and CCB-CS) take benefit of it.

8.3.2 Justification-based Selection of Training Examples

In this section we are going to present the *Justification-based Selection of Training Examples* (JUST) strategy. JUST is a case retention strategy that considers the case base as a whole and whose goal is to reduce the number of cases in a case base while maintaining (within some error margin) the classification accuracy of the original case base. In fact, JUST is not a collaboration strategy, but an individual strategy that any CBR system can use (including CBR agents).

JUST is an iterative strategy that selects cases from a case base C and adds them to another (reduced) case base C^r, until certain termination criterion $\mathcal{T}$ is met. The termination criterion could be any property of the new case base C^r, but we will focus on these two:

- $\mathcal{T}_M$: terminate for a case base C^r with at most M cases,
- $\mathcal{T}_\alpha$: terminate for a case base C^r with an accuracy level α.

At each round t, the case base C is divided on two sets of cases: C_t^r (the reduced case base) and C_t^u (the set of unseen cases). Initially, at round $t = 0$, $C_0^r = \emptyset$ and $C_0^u = C$, and at each round, JUST selects a subset of cases B_t of C_t^u to be added to C_t^r forming $C_{t+1}^r = C_t^r \cup B_t$ (and also updating $C_{t+1}^u = C_t^u - B_t$).

The main idea of JUST is that at each round t, the minimum subset of cases of C_t^u with the maximum JCU utility is added to C_t^r. In order to achieve that, at each round t, JUST creates an *exam* E_t consisting on a subset of cases of C_t^u. The exam will be used to evaluate the JCU utility of the potential sets of cases to add. Then, the minimum subset of cases $B_t \subseteq C_t^u$ with the highest JCU utility will be selected and added to the reduced case base.

Notice that selecting the minimum subset of cases $B_t \subseteq C_t^u$ with the highest JCU utility can be prohibitive since there are 2^n possible subsets of C_t^u (where n is the size of C_t^u), and the utility of each different subset should be computed. Therefore, in this section we are going to present an efficient way to select the minimum subset of cases with the highest utility.

We will introduce some necessary concepts and notation before presenting JUST. We will call E_t to the exam generated by JUST at round t. Moreover, at each round t the problems in an exam E_t are solved by the system using the cases in the reduced case base C_t^r and the result is a justified prediction (in the form of a JER) for every problem $P \in E_t$. We will note by $\mathbf{J}_{E_t}$ the set of JERs for the problems in the exam E_t, and $\mathbf{J}_{E_t}^-$ to the subset of incorrect JERs from $\mathbf{J}_{E_t}$ (i.e. those JERs that predict an incorrect solution class).

For each incorrect JER $\mathbf{J} \in \mathbf{J}_E^-$ produced in a round t, we can define its *refutation set*, as follows:

Definition 8.3.2. *The* refutation set $R_t^{\mathbf{J}}$ *of an incorrect JER* $\mathbf{J}$ *at a round* t *is defined as the set of cases of* C_t^u *that are valid counterexamples of* $\mathbf{J}$*, i.e* $R_t^{\mathbf{J}} = VCE(\mathbf{J}, C_t^u)$

Function JUST $(C, \mathcal{T}, m)$
  $t = 0$; $C_0^r = \emptyset$; $C_0^u = C$;
  Do
    $E_t = \underline{\text{select-exam}}(C_t^u, m)$;
    $\mathbf{J}_{E_t} = \underline{\text{build-JERs}}(E_t)$;
    $\mathbf{J}_{E_t}^- = \{\mathbf{J} | \mathbf{J} \in \mathbf{J}_{E_t} \wedge \mathbf{J}.S \neq S_{\mathbf{J}.P}\}$;
    $\mathcal{R}_t = \underline{\text{build-refutation-sets}}(\mathbf{J}_{E_t}^-, C_t^u)$;
    $B_t = \underline{\text{build-belying-set}}(\mathcal{R}_t)$;
    $C_{t+1}^r = C_t^r \cup B_t$; $C_{t+1}^u = C_t^u - B_t$; $t = t + 1$;
  While(not $\mathcal{T}$);
  Return(C_t^r);
End-Function

Figure 8.9: The JUST algorithm, where C is the initial case base, $\mathcal{T}$ is the termination criterion and m is the exam size.

The cases in the refutation set $R_t^{\mathbf{J}}$ for an incorrect JER $\mathbf{J}$ are the cases in C_t^u that can potentially prevent the system of making the same error again, and therefore they are candidates to be added to the reduced case base.

Therefore, at each round t, we can define the collection of refutation sets $\mathcal{R}_t = \{R_t^{\mathbf{J}} | \mathbf{J} \in \mathbf{J}_{E_t}^-\}$. Finally, we can also define the *belying set*:

Definition 8.3.3. *The* belying set $B_t \subseteq C_t^u$ *is the minimum set of cases from* C_t^u *that contains at least one case of each refutation set in* $\mathcal{R}_t$*, i.e.:*

$$\begin{array}{l} B_t \subseteq C_t^u \wedge \forall_{R_t^{\mathbf{J}} \in \mathcal{R}_t} B_t \cap R_t^{\mathbf{J}} \neq \emptyset \\ \nexists B_t' \subseteq C_t^u | \#(B_t') < \#(B_t) \wedge \forall_{R_t^{\mathbf{J}} \in \mathcal{R}_t} B_t' \cap R_t^{\mathbf{J}} \neq \emptyset \end{array} \quad (8.4)$$

Notice that the belying set is the minimum set of cases that, if added to the reduced case base C_t^r, can potentially prevent all the errors made while solving the exam E_t. Thus, the belying set is the minimum subset of cases from C_t^u with the maximum JCU utility. Therefore, the cases in the belying set B_t are the cases that will be added to C_t^r at round t.

Figure 8.9 presents the algorithm of JUST in detail. JUST receives always three parameters: the case base to reduce C, the termination criterion $\mathcal{T}$, and the exam size m. Initially (at round $t = 0$) JUST initializes the reduced case base $C_0^r = \emptyset$ and the set of unseen cases $C_0^u = C$. As we have previously said, JUST can use any termination criteria, but we will focus on two specific ones: $\mathcal{T}_M$ and $\mathcal{T}_\alpha$.

JUST starts each iteration by generating the exam E_t. An exam is generated by taking a random subset $C_t^E \subseteq C_t^u$ of m cases from the set of unseen cases and taking the problems in those cases: $E_t = \{c.P | c \in C_t^E\}$ (if C_t^u contains less than m cases, then all the cases in C_t^u are used as the exam). Then, each problem in the exam E_t is solved and the set containing the justified predictions $\mathbf{J}_{E_t}$ is generated. Now, since the CBR system knows which is the correct solution of

each problem in the exam (since the exam has been generated from cases of C_t^u), the set of incorrect JERs $\mathbf{J}_{E_t}^-$ is selected from $\mathbf{J}_{E_t}$. The next step is to generate the refutation sets for all the JERs in $\mathbf{J}_{E_t}^-$, forming the collection of refutation sets $\mathcal{R}_t$. Finally, the belying set B_t is created by selecting the minimum subset of cases of C_t^u that has at least one case of each refutation set in $\mathcal{R}_t$. The cases in B_t are added to the reduced case base C_t^r and are removed from the set of unseen cases C_t^u. At the end of each round the termination criterion $\mathcal{T}$ is checked, and JUST either ends or starts a new round. When the termination criterion decides that no more rounds are needed, C_t^r is returned as the reduced case base, while all the cases in C_t^u are discarded.

As a final remark, notice that the computation of the belying set (that involves computing the refutation sets) is equivalent to the following: find a minimum subset $B_t \subseteq C_t^u$ that maximizes $JCU(B_t, C_t^E)$. Thus, the core idea of JUST is to add at each iteration the minimum set of cases with highest utility (JCU utility) for the case base, while the refutation sets and the belying set are just an efficient way to find such a minimum set of cases with maximum utility.

Moreover, as in the definition of JCU, the exams are a key element in order to properly determine the utility of cases. If an exam is a good sample of the data, then JCU will be a good approximation of the case utility. Notice that JUST takes the size of the exams m as a parameter: if we set a large exam size, JUST will assess the utility of the cases with a better accuracy (assuming that the larger the exam, the better the sample of the data), and therefore the reduction of the case base achieved will be larger. However, the computational cost will be higher. If we set a small exam size, JUST requires less execution time (since at each round, the size of the exam the system has to solve is smaller), but the reduction of the case base achieved with JUST will not be as large as it could have been. In the experimental section, we will present an analysis of the performance of JUST for several exam sizes.

8.3.2.1 The Termination Criterion

In our experiments we have used two different termination criteria $\mathcal{T}$:

- $\mathcal{T}_M$: terminate for a case base C^r with at most M cases,
- $\mathcal{T}_\alpha$: terminate for a case base C^r with a certain accuracy level α.

When using the termination criterion $\mathcal{T}_M$ JUST will finish once C_t^r has reached the size M. In fact, JUST yields C_{t-1}^r when it detects at round t that $size(C_t^r) > M$.

When using the termination criterion $\mathcal{T}_\alpha$ (in our experiments, α takes values around 90%) JUST uses the answers of the exams as an estimation of the current classification accuracy. However, depending on the size of the exam, this estimation may be more or less reliable. If the size of the exam is large, the accuracy obtained by that exam is a good estimation of the classification accuracy of the CBR system; thus, when the accuracy obtained by the system in a large exam is above α, JUST can terminate. However, if the size of the

exam is small, JUST needs more than one exam to have a good estimate of the accuracy. The number of exams needed to have a good estimation can be determined assuming that the correctness of an answer can be modelled as a binomial distribution. The binomial distribution model tells us that, for estimating accuracy values around $\alpha = 90\%$ (having a certainty of 66% of having an error lower than the 4%) at least 60 answers are required (see Appendix C for a detailed explanation). For an exam size $m = 20$, 3 exams are enough to be 66% sure that the accuracy of the CBR system does not differ more than a 4% from the estimated one. Thus, if the average accuracy α' of 3 consecutive exams of size $m = 20$ is higher than α, JUST can terminate with a 66% certainty that the accuracy of the CBR system is in a $\pm 4\%$ margin around α'. For an exam size of $m = 10$, 6 consecutive exams are needed for the same result. Summarizing, the termination policy is the following: if the average accuracy in the last $60/m$ exams is above α, JUST will stop.

In the experiments section, we will present results using both termination criteria.

8.3.2.2 Exemplification

In this section we are going to present an exemplification of JUST using the two dimensional data set shown in Figure 8.6. The goal of JUST will be to build a case base with a reduced number of cases, that has the same expected classification accuracy than the original case base. The original case base has 64 cases: 37 cases with solution class *blue* and 27 cases with solution class *red*.

Figures 8.10 and 8.11 show the execution of JUST for the mentioned case base round by round. JUST has needed 6 rounds (from 0 to 5) in order to achieve the final reduced case base. For each round, 2 pictures are shown: the left hand side shows the 10 problems used as the exam (and which of them have been failed), and the right hand one shows the cases that have been incorporated to the reduced case base (the belying set) to fix the errors made in the exam. The problems in the the exam are shown using a thick black line around them, and the new problems added to the reduced case base at each round are shown using a thick white line around them.

Let us analyze the process round by round:

- In round $t = 0$ the reduced case base C_0^r is empty, and thus all the 10 problems in the exam E_0 are solved incorrectly. Moreover, since the system cannot provide any justifications for the predictions made in the exam (since it has no cases), all the cases in C_0^u of class *blue* are valid counterexamples of all the *blue* problems in the exam, and all the *red* cases in the exam are valid counterexamples of all the *red* problems in the exam. Thus, any subset of cases of C_0^u composed of a *red* case and a *blue* case is a belying set. Therefore, a random *red* case and a random *blue* case are selected and added into the reduced case base (as shown in the right hand picture of round $t = 0$).

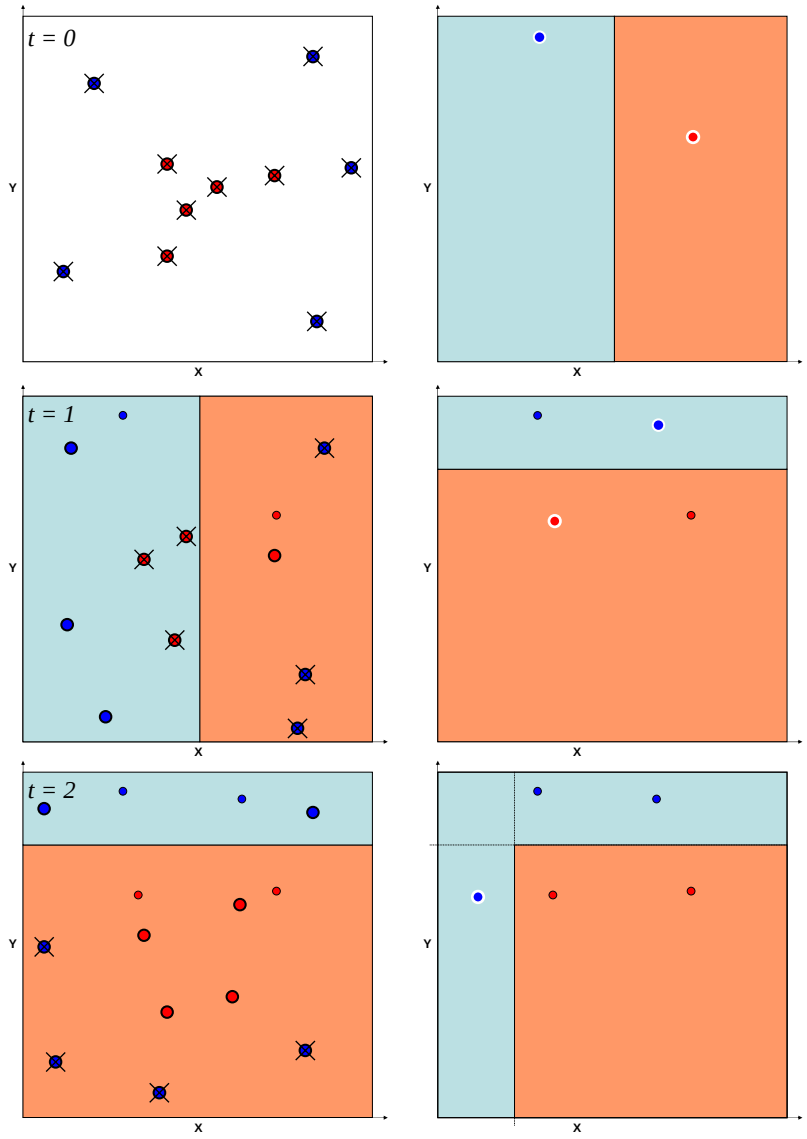


Figure 8.10: Three first rounds of the execution of JUST for a two dimensional case base.

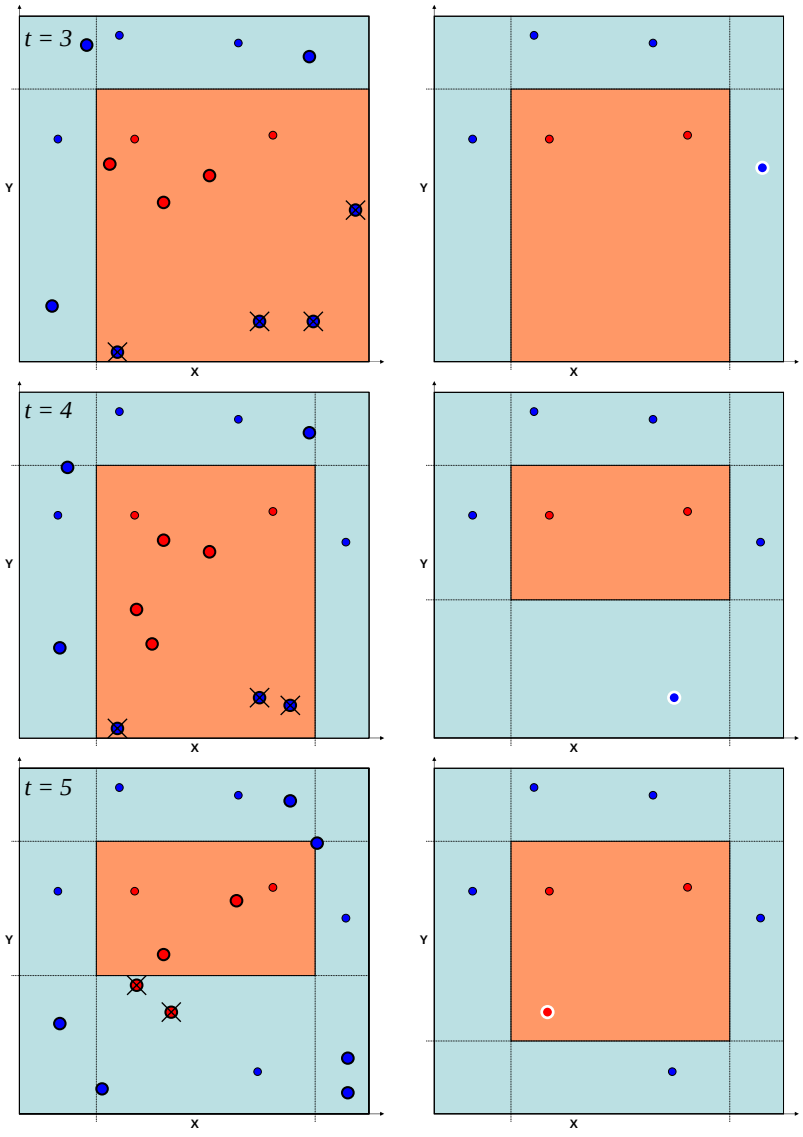


Figure 8.11: Rounds 3 to 5 of the execution of JUST for a two dimensional case base.

- In round $t = 1$, 6 problems of the exam are solved incorrectly (as shown in the left hand side of Figure 8.10 for round $t = 1$). The justification given for the failed *red* problems is $\langle X < 0.5 \rangle$, thus, any *red* case in C_1^u with a value for the feature X lower than 0.5 is a valid counterexample of those three cases. For the three incorrectly solved *blue* problems, any case in C_1^u with a value for the feature X higher or equal than 0.5 is a valid counterexample. Thus, two more cases are added into the reduced case base: a red case with $X < 0.$ and a blue case with $X \geq 0.5$ (as shown in the right hand side of Figure 8.10 for round $t = 1$).

- In round $t = 2$ only four *blue* problems are incorrectly solved in the exam, and all of them have the same justification: $\langle Y < 0.77 \rangle$. Thus, any *blue* case in C_2^U satisfying such justification is a valid counterexample of all those incorrect justifications. One is randomly selected, and added into the reduced case base as shown in the picture. Notice that in just three steps and with just 5 cases, the reduced case base already has detected that the top and left parts of the problem space contain *blue* problems.

- In round $t = 3$, again four *blue* problems are incorrectly solved in the exam. All of them have again the same justification: $\langle X \geq 0.22 \wedge Y < 0.77 \rangle$. A belying set consisting of only one *blue* case satisfying that justification is enough, as shown in the right hand picture for round $t = 3$.

- In round $t = 4$, only three *blue* problems are incorrectly solved. Again a belying set consisting of a single blue case is enough, as shown in the right hand side picture for round $t = 4$.

- In round $t = 5$, only two *red* problems are incorrectly solved. Both of them have the same justification: $\langle X \geq 0.22 \wedge X < 0.80 \wedge Y < 0.41 \rangle$. A single *red* case satisfying that justification is enough. The right hand picture for round $t = 5$ shows the reduced case base at the end of round $t = 5$, consisting of only 8 cases, 3 *red* ones and 5 *blue* ones.

No more rounds are shown, since with that 8 cases, all the problems in the original case base can be correctly solved, and thus no more cases are added. Therefore, JUST has achieved a reduction from 64 cases to just 8 cases in this domain.

8.3.3 Experimental Evaluation

This section presents experimental results comparing the performance achieved by a CBR system after using the JUST case base reduction strategy with the performance of the system without reducing the case base.

We have used the sponge, soybean and zoology data sets as our test bed. Moreover, we have made experiments comparing the JUST technique with three base strategies: a base CBR system that does not use any case base reduction technique, a CBR sytem that uses the CBL2 [2] case base reduction technique,

	Sponges		*Soybean*		*Zoo*	
	Accuracy	CB size	Accuracy	CB size	Accuracy	CB size
JUST	88.12%	32.34%	88.59%	55.00%	95.44%	38.86%
CBL2	82.14%	22.71%	81.00%	28.62%	95.24%	18.59%
CNN	86.36%	33.57%	84,40%	43,87%	95.30%	18.19%
CB	88.21%	100.00%	88.50%	100.00%	95.45%	100.00%

Table 8.4: Comparison of the classification accuracy and case base size of JUST (using an exam size $m = 20$) against CB2 and with the complete case base (CB).

and a CBR system that uses the CNN [37] case base reduction technique (see Section 2.2.1.1 for a small explanation of these two methods). In an experimental run, a 10% of the cases are separated from the rest and will be used as the test set. The other 90% of the cases is used as the system case base. Then, the case reduction technique is applied and the classification accuracy is measured using the test set.

Table 8.4 shows the results obtained by three CBR systems, one using the JUST case base reduction technique, another using the CB2 case base reduction technique and the third one using the complete case base for the three datasets (sponges, soybean and zoo). We have used JUST with an exam size of $m = 20$, and a termination criterion of reaching an accuracy of about $\alpha = 90\%$ for the sponges and soybean data sets, and of about $\alpha = 96\%$ in the zoo data set (we have chosen those parameters as slightly greater values than the accuracy values of the complete case bases).

Table 8.4 shows that JUST has been able to reduce the size of the case bases to the 32.34% of the total number of cases in the sponges case base, to the 55.00% in the soybean case base and to the 38.86% in the zoo case base. This reduction is achieved without losing classification accuracy: notice that the accuracy for JUST in the sponges data set is 88.12% while the accuracy without case reduction is 88.21%, the difference being not statistically significant. For the soybean data set, JUST has achieved a classification accuracy of 88.59% while the CBR system without case reduction achieves a 88.50% of classification accuracy; again the difference is not statistically significant. In the zoo data set, the accuracy achieved by JUST is 95.44%, and the accuracy achieved with the complete case base is 95.45%. Moreover, the termination criterion of JUST requested case bases with a 90% of classification accuracy in soybean and sponges and 96% in the zoo data set. Notice that JUST has stopped before reaching that accuracy in all the case bases. The reason is that the termination criterion used in our experiments has a margin of error of $\pm 4\%$ (see Section 8.3.2.1). A termination criterion with a lower margin of error could be used if need be.

Comparing JUST with CB2 in Table 8.4, notice that CB2 obtains reduced case bases that are even smaller than the achieved by JUST: 22.71% in the sponges data set, 28.62% in the soybean data set and 18.59% in the zoo data set

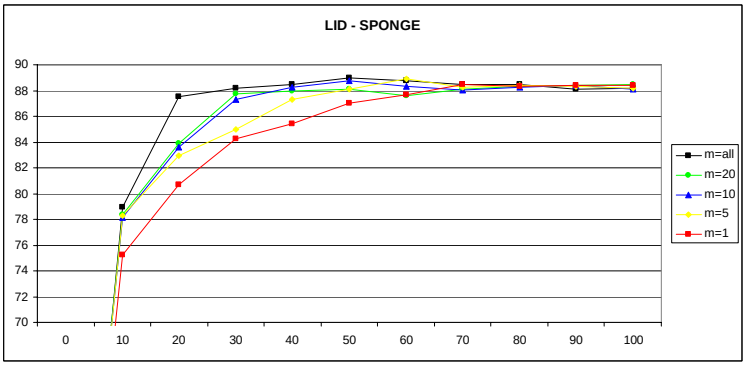


Figure 8.12: Comparison of the accuracy evolution in the reduced case bases for several exam sizes in the sponges dataset using JUST.

versus 32.34%, 55.00% and 38.86% achieved with JUST. However, CB2 reduces the case base without preserving the classification accuracy in two of the three data sets; CB2 has been able to keep the degree of accuracy of the complete case base only in the zoo data set, in the other two data sets, the accuracy achieved by CB2 is appreciably lower than that of the complete case base: 82.14% in the sponges data set and 81.00% in the soybean data set. JUST, however, maintains the accuracy of the complete case base, namely 88.12% and 88.59% respectively. CB2 has problems in two data sets because cases are discarded in a very eager way. JUST, however, has a broader view of the problem and never discards any case until termination is decided. Thus, JUST is able to discard a considerable number of cases while maintaining the accuracy levels of the complete case base.

Finally, comparing JUST to CNN, we see that although CNN is more robust than CBL2 (since it achieves higher classification accuracy values than CBL2 in the sponge and soybean data sets) it achieves lower classification accuracy values than JUST. Moreover, the case base reduction achieved by CNN is an intermediate between JUST and CBL2. Summarizing, CBL2 achieves the higher case base reduction, but at the cost of losing classification accuracy; CNN achieves higher classification accuracy values than CBL2, since it does not discard cases as eagerly as CBL2; finally JUST is even more robust than CNN since it systematically achieves higher classification accuracy values than both CBL2 and CNN while still reducing significantly the size of the case base. In fact, the classification accuracy values achieved by JUST are undistinguishable to that of using the complete case base.

In order to test the effects of the size of the exams in JUST we have experimented with several exam sizes: 1, 5, 10, 20 and unlimited (when exam size is unlimited, the whole set of cases C_t^u is used as the exam). Figures 8.12, 8.13 and 8.14 show the accuracy results for JUST in the sponges, soybean and zoology data sets respectively for several exam sizes. The plot where the exam size reads $m = all$ represents a system where all the cases in C_t^u are used as

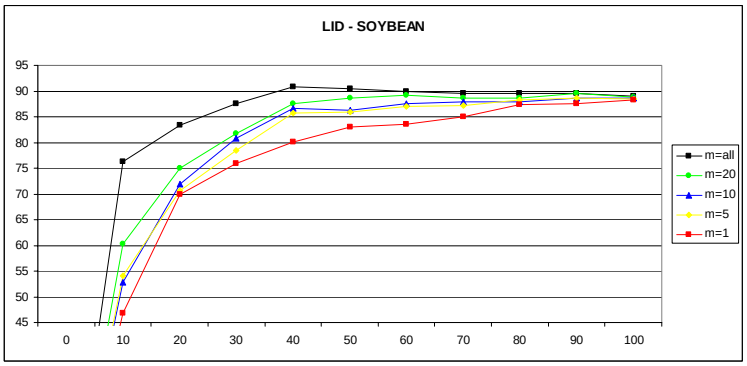


Figure 8.13: Comparison of the accuracy evolution in the reduced case bases for several exam sizes in the soybean dataset using JUST.

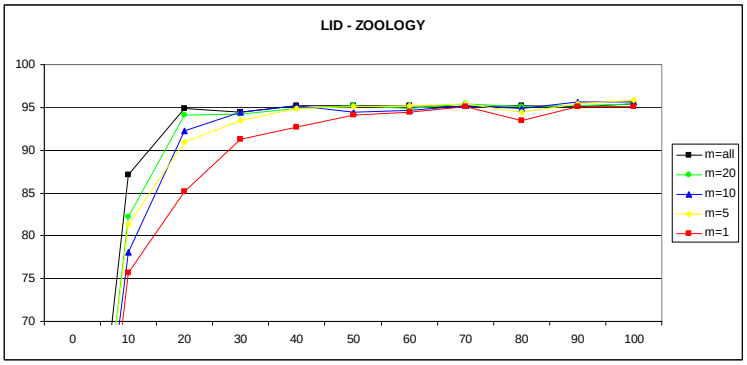


Figure 8.14: Comparison of the accuracy evolution in the reduced case bases for several exam sizes in the zoology dataset using JUST.

the exam (i.e. an unlimited exam size). These experiments are performed using the case base size termination criterion $\mathcal{T}_M$ (see Section 8.3.2.1) for sizes 10%, 20%, and so on, up to 100% percentage of the complete case base. Figures 8.12, 8.13 and 8.14 plot the accuracy achieved by JUST varying the desired size of the reduced case base. For each exam size, a different plot is shown.

Figure 8.12 shows that as the exam size increases, JUST is able to reach higher accuracies with smaller case bases. For instance, reaching an accuracy higher than 85% with an exam size $m = 1$, JUST needs a case base of the 40% of the size of the complete case base, while with the exam size is $m = 5$, only a 30% of the original cases are needed. In the extreme, when the exam size is unlimited (i.e. all the cases in C_t^u are used as the exam at each iteration), only a 20% of the cases are needed. This is because when the exam size is larger, JUST can obtain a more accurate estimation of the utility of the cases to be added. Moreover,

notice that when the termination criterion is to obtain a case base with more than the 70% of all the cases, there is no difference in the classification accuracy by varying the exam size. Notice also that in some experiments JUST has been able to obtain case bases that reach a higher accuracy than the complete case base. For instance, when the exam size is unlimited, the accuracy achieved with a case base with the 50% of the cases of the complete sponges case base is 89.00% while the accuracy of the complete case base is 88.21% (more on this later).

Figure 8.13 shows the experiments using the soybean data set. Notice that as the exam size increases, as before, the accuracy achieved by JUST also increases. Moreover, the accuracy achieved by JUST in the soybean data set with an unlimited exam size is much higher than the accuracy with smaller exam sizes. For instance, with a case base containing the 40% of the cases in the complete case base, JUST with an unlimited exam size achieves an accuracy of 90.88% while the complete case base accuracy is 88.50%. This means that the exam size needed by JUST in the soybean data set to achieve a good performance is larger than the exam size needed in the sponges data set. The reason seems to be that the soybean data set has 19 solution classes and the sponges data set only 3. The larger the number of classes, the larger the exams should be in order to obtain representative information of the weak points of the reduced case base.

Figure 8.14 shows the experiments using the zoology data set. Notice that the same results observed in sponge and zoology are also observed in the zoology data set: the larger the exam size, the less cases that JUST needs to achieved a high level of classification accuracy.

The overall conclusion is that the larger the exam size, the higher the performance of JUST, i.e. as we increase the exam size, we will obtain reduced case bases that are smaller and more accurate. This confirms the expected result that using larger exams JCU provides better estimations of the utility.

However, as we increase the exam size, we also increase the computational cost of JUST. Let us analyze JUST in computational cost as the number of retrievals performed during the case base reduction process. The cost of JUST can be divided in two costs: the cost of solving the exams, and the cost of building the belying sets. Let T be the number of iterations that JUST has executed, n the number of cases in the complete case base C, and m the exam size. The cost of solving the exams is at most $T \times min(m, n)$ retrievals, and the cost of building the belying set is also at most $T \times min(m, n)$. Therefore, the complexity of JUST is of order $T \times min(m, n)$. As explained in Section 8.3.2, the maximum number of iterations is n, the number of cases in the complete case base C. Therefore, the worst case complexity is $n \times min(m, n)$, i.e. $O(n^2)$.

We have also performed an empirical evaluation of the JUST complexity varying the exam size in the soybean data set, as the following table shows (where the number of iterations and the number of retrievals needed by JUST in the soybean data set):

m	1	5	10	20	all
retrievals	256.8	713.0	458.0	1158.0	1627.7
iterations	256.8	142.6	45.8	57.9	8.2

The termination criterion used to perform those experiments is to reach an accuracy of the 90%. We see that the number of retrievals increases as the exam size increases (as predicted by the theoretical complexity of $n \times min(m, n)$). However, the practical complexity is much lower than the theoretical complexity, specially for large exam sizes, where the number of iterations is much smaller than the theoretical maximum n. This result shows that JUST can be used with large exam sizes without having to pay a high computational cost. Notice that the practical cost for an unlimited exam size, is 1,627.7 retrievals in average, while the theoretical bound is $n^2 = 276 \times 276 = 76{,}176$ retrievals, since in the soybean data set the complete case base C has 276 cases (the other 10% is reserved as the test set). We can conclude that if the computational cost is not a problem in our CBR system an unlimited exam size should be used in order to obtain the maximum benefit from JUST. Moreover, although the cost of JUST with large exam sizes is not prohibitive (as we have seen in our experiments), smaller exam sizes may be used in order to reduce the computational cost if need be.

Summarizing the results, we have seen that JUST is a effective case base reduction method. By varying the exam size m, we can modify the behavior of JUST: with small exam sizes we can obtain moderate case base reductions at a low cost, and with large exam sizes we can obtain large case base reductions, but at a higher computational cost. This is clearly an advantage with respect to other case base reduction methods that are not parametric, since JUST can be adapted to several CBR systems that have different size and computational time restrictions. Moreover, JUST can accept another parameter: the termination criterion. By changing the termination criterion, we can request JUST to obtain reduced case bases that satisfy any desired conditions. Moreover, we have seen in the experiments section that there are reduced case bases that achieve higher accuracies than the complete case base. For instance, in Figure 8.13 the optimal point (with an unlimited exam size) is to build a reduced case base with the 40% of the original cases (since this is where the maximum accuracy was reached). Therefore JUST allows to manually select the best point in the plot, however it remains as future work to automatically find this optimal accuracy point.

8.4 Collaborative Case Bargaining

In this section we are going to present the *Collaborative Case Bargaining* (CCB-CS) collaboration strategy. CCB-CS is a collaboration strategy that combines the two approaches already presented in this chapter: it takes into account the multi-agent aspect of case retention (as MAR-CS), and it uses justifications (and specifically JCU) in order to assess the utility of new cases (as JUST).

Moreover, with CCB-CS we will introduce the idea of *delayed retention*. A delayed retention strategy is one where cases that are candidates to be retained,

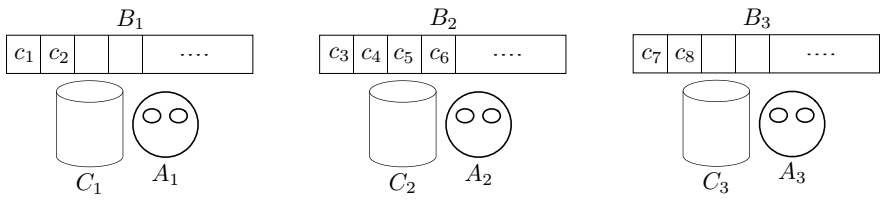


Figure 8.15: Agents using *delayed retention* need to have a pool de delayed retention cases.

instead of being considered for retention, are stored in a buffer: the *Pool of Delayed Retention Cases* (often called the *pool*, for short). When the pool of a CBR agent is full, all the cases in the pool will be examined to decide which of them retain. The main advantage of delayed retention is that it allows the CBR agents to consider cases in batches instead of one by one (and this is beneficial, as we will show later). Moreover, in the remained of this section we will note by B_i the pool of cases of an agent A_i. Figure 8.15 shows a $\mathcal{MAC}$ system composed of three agents with their respective pools.

The basic idea of CCB-CS is that when an agent A_i in a $\mathcal{MAC}$ system has its pool B_i full, A_i will convene a committee of agents $\mathcal{A}^c$. The agents will share the cases in their pools, and a bargaining process will start to decide which agent will retain each case. During bargaining, JCU will be used to assess the utility of the cases that are being bargained for. Moreover, the bargaining protocol ensures that cases are retained by the agent that most needs them (i.e. by the agent for which each case the highest utility).

Thus, CCB-CS combines three ideas: it is a delayed retention strategy since the agents do not consider cases one by one; it is a multi-agent collaboration strategy for case retention and uses a bargaining mechanism to distribute cases among the agents; and finally the agents use justifications (and specifically JCU) to assess the utility of the cases they are bargaining for.

Let us introduce some definitions and notation before presenting CCB-CS in detail.

We will note by $\widehat{B} = \bigcup_i B_i$ the union of the cases present in each one of the pools of all the agents taking part in CCB-CS; e.g. Figure 8.15 shows $\widehat{B}$ for a $\mathcal{MAC}$ system composed of three agents. Moreover, CCB-CS is an iterative collaboration strategy, and at each round t one case will be retained by an agent in $\mathcal{A}^c$. Thus, each case will be retained by a single agent to avoid increasing the Committee Redundancy (see Section 4.4.2). We will call $\widehat{B}_t$ to the set of cases that at a round t have not yet been retained by any agent (initially $\widehat{B}_0 = \widehat{B}$).

At each round t, every agent in $\mathcal{A}^c$ will provide an *utility record* for each case in $\widehat{B}_t$:

Definition 8.4.1. *An* utility record $\mathbf{U} = \langle A, C, V \rangle$ *is a tuple containing the utility value V estimated by the agent A for the case C.*

Thus, at each round t, each agent A_i will provide a set of utility records $\mathcal{U}_t^{A_i}$. Once all the agents have provided their utility records, the utility record $\overline{\mathbf{U}}_t$ with the highest utility value is selected, and the agent $\overline{\mathbf{U}}_t.A$ will be the one that will retain in round t the case with maximum utility $\overline{\mathbf{U}}_t.C$. Then, that case is removed from the set $\widehat{B}_t$ of cases to retain and a new round $t+1$ starts. This process is repeated until no agent is interested in any case, or until there are no more cases to bargain.

Let us now define CCB-CS formally:

Definition 8.4.2. Collaborative Case Bargaining *(CCB-CS) is a collaboration strategy $\langle I_{CCB-CS}, D_U \rangle$, where I_{CCB-CS} is the CCB-CS interaction protocol shown in Figure 8.16 and D_U is a utility assessment decision policy capable of estimating the utility of a case for an individual agent.*

The following section presents the I_{CCB-CS} interaction protocol, and after that, a D_U decision policy based on JCU is presented.

8.4.1 Collaborative Case Bargaining Protocol

In this section we are going to explain the I_{CCB-CS} interaction protocol (shown in Figure 8.16) for a committee of agents $\mathcal{A}^c$ that has been convened by a convener agent A_c.

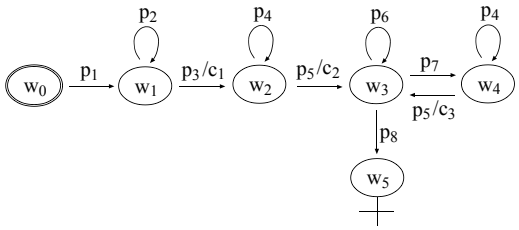
When a group of agents $\mathcal{A}$ in a $\mathcal{MAC}$ system want to use CCB-CS as their case retention strategy, each individual agent is required to store all the cases they receive as candidates to be retained in their local Pools of Delayed Retention Cases. Thus, the procedure that each agent follows is: store each case in the local pool; when the pool B_c of an agent A_c is full, then A_c convenes the committee $\mathcal{A}^c$ and the protocol I_{CCB-CS} starts. Specifically I_{CCB-CS} works as follows:

1. The protocol starts when the convener agent A_c broadcasts a message p_1 to all the other agents in $\mathcal{A}^c$ asking for the contents of their pools of delayed retention cases (notice than an agent starts the I_{CCB-CS} protocol when its pool is full). The protocol moves to state w_1.

2. In state w_1 every agent A_j that has received message p_1 answers a message p_2 to A_c with the contents of its pool B_j.

3. When A_c has received the contents of the pools of all the other agents, they are aggregated on a set $\widehat{B} = \bigcup_j B_j$ that contains the cases from the pools of all the agents in $\mathcal{A}^c$. Then, A_c broadcasts $\widehat{B}$ to all the other agents with message p_3 and the protocol moves to state w_2.

4. In state w_2 every agent A_j that has received message p_3 answers sending a confirmation message p_4 to A_c.



p_1 : $Request(?A_c, all, start)$

p_2 : $Inform(?A_j, !A_c, ?B_j)$

p_3/c_1 : $Inform(!A_c, all, ?B = \bigcup_j B_j)/$
$|w_1 w_1 B_j| = !(n-1)$

p_4 : $Inform(?A_j, !A_c, ok)$

p_5/c_2 : $Request(!A_c, all, utility)/$
$|w_2 w_2 ?A_j| = !(n-1)$

p_6 : $Inform(?A_j, !A_c, \mathcal{U}_t^{A_j})$

p_7 : $Inform(!A_c, all, \mathbf{U}_t^{max})$

p_5/c_3 : $Request(!A_c, all, utility)/$
$|w_4 w_4 A_j| = !(n-1)$

p_8 : $Inform(!A_c, all, termination)$

Figure 8.16: Interaction protocol for the CCB-CS collaboration strategy.

5. When all the confirmation messages have been received, A_c assumes that every agent has now the cases in $\widehat{B}$. Thus, the first round $t = 0$ starts and all the agents assume that $\widehat{B}_0 = \widehat{B}$. Then, A_c broadcasts a message p_5 requesting the utility records for the round $t = 0$ and the protocol moves from state w_2 to state w_3.

6. In state w_3, each agent A_j use its D_U policy to estimate the utility of each case c in $\widehat{B}_t$ and stores it in an utility record $\mathbf{U} = \langle A_j, c, D_U(c) \rangle$. Then, the set $\mathcal{U}_t^{A_j}$ of all the utility records computed by A_j at round t is send to A_c in message p_6.

7. When A_c has received the sets of utility records of all the other agents (including its own), the record with the highest utility value $\overline{\mathbf{U}}_t \in \bigcup_{A_j \in \mathcal{A}^c} \mathcal{U}_t^{A_j}$ is selected.

   - If $\overline{\mathbf{U}}_t.V > 0$, A_c sends a message to the rest of agents telling that the agent $\overline{\mathbf{U}}_t.A$ should retain the case $\overline{\mathbf{U}}_t.C$. The protocol moves to w_4.
   - Otherwise ($\overline{\mathbf{U}}_t.V = 0$), A_c sends a message to the rest of agents stating that the protocol is over and the remaining cases in $\widehat{B}_t$ will

be discarded. The protocol moves to state w_5, that is a final state and the protocol ends.

8. In state w_4 agent $\overline{\mathbf{U}}_t.A$ retains the case $\overline{\mathbf{U}}_t.C$ and sends a confirmation message p_4 to A_c. The other agents simply send a confirmation message to A_c. All the gents update $\widehat{B}_{t+1} = \widehat{B}_t - \{\overline{\mathbf{U}}_t.C\}$.

9. Once all the confirmation messages are received, a new round $t+1$ starts and A_c sends again a message p_5 (that moves protocol to state w_3) requesting for the new utility records in round $t+1$.

Notice that each iteration requires that the agents to reevaluate their utility assessments for the cases in $\widehat{B}_t$. The reason is that when an agent A_i retains a case in a round t, its utility assessment for the rest of the cases in $\widehat{B}_t$ may change. Thus A_i must reevaluate its utility values in round $t+1$. Moreover, A_i will use $\widehat{B}_{t+1}$ as the exam to assess its new utility values, while the other agents would have their utility values assessed using $\widehat{B}_t$. Thus, the utility values of A_i could not be compared with those of the rest of the agents. Therefore, in order to be able to compare the utility values assessed by all the agents, we require that every agent reevaluates their utility values using $\widehat{B}_{t+1}$.

CCB-CS may appear to be a complex way to distribute the cases among the agents but it is designed in this way because the order in which the cases are bargained does matter. In Section 8.4.2.1 we will discuss further these issues.

The next section presents the D_U decision policy with which the agents assess the utility of cases.

8.4.2 Collaborative Case Bargaining Decision Policies

The agents require a Utility Assessment decision policy D_U in order to use CCB-CS. In this section, we are going to present a D_U decision policy based on JCU:

Definition 8.4.3. *The* Justification-based Utility Assessment *decision policy D_U used to estimate the utility of a case c consists of using JCU with the cases in $\widehat{B}_t$ as the exam, i.e.:*

$$D_U(c) = JCU(\{c\}, \widehat{B}_t)$$

The next section presents an exemplification of how CCB-CS works using this decision policy.

8.4.2.1 Exemplification

Let us illustrate the behavior of the CCB-CS collaboration strategy with an exemplification. Consider a $\mathcal{MAC}$ system composed of 3 agents $\mathcal{A} = \{A_1, A_2, A_3\}$, that have individual pools of delayed retention cases B_1, B_2 and B_3 that can store 3 cases each. At a given time, the pools of the three agents

a) *Round 1*

	c_1	c_2	c_3	c_4	c_5
A_1	0	0.4	0.6	0	0.2
A_2	0.4	0	0.4	0	0
A_3	0	0	0	0.2	0.4

b) *Round 2*

	c_1	c_2	c_3	c_4	c_5
A_1	0	0	-	0	0
A_2	0.5	0	-	0	0
A_3	0	0	-	0.25	0.5

c) *Round 3*

	c_1	c_2	c_3	c_4	c_5
A_2	-	0	-	0	0
A_1	-	0	-	0	0
A_3	-	0	-	0.33	0.66

d) *Round 4*

	c_1	c_2	c_3	c_4	c_5
A_1	-	0	-	0	-
A_2	-	0	-	0	-
A_3	-	0	-	0	-

Table 8.5: Evolution of the utility values, for 3 agents A_1, A_2 and A_3 and a set $B = \{c_1, c_2, c_3, c_4, c_5\}$ of 5 cases in the CCB protocol.

contain the following cases: $B_1 = \{c_1, c_2, c_3\}$, $B_2 = \{c_4\}$ and $B_3 = \{c_5\}$, where $c_1 = (P_1, S_1)$, $c_2 = (P_2, S_2)$, etc.

When the pool B_1 of agent A_1 is full, agent A_1 initiates the I_{CCB-CS} interaction protocol. Both A_2 and A_3 broadcast the cases in their pools so that all the agents have access to the set of all delayed retention cases $\widehat{B} = \{c_1, c_2, c_3, c_4, c_5\}$.

When the first round $t = 0$ starts, $\widehat{B}_0 = \widehat{B}$ and all the agents estimate the JCU utility of the cases in $\widehat{B}_0$. Let us focus on how agent A_1 uses JCU: first, A_1 takes the set $E = \{P_1, ..., P_5\}$ and builds a JER for each problem in E. Assume that A_1 fails to correctly solve three problems, P_2, P_3 and P_5, and therefore the set $\mathbf{J}_E^- = \{\mathbf{J}_2, \mathbf{J}_3, \mathbf{J}_5\}$ has three JERs. A_1 builds then the refutation sets for those three JERs: $R_0^{\mathbf{J}_2} = \{c_2, c_3\}$, $R_0^{\mathbf{J}_3} = \{c_3\}$ and $R_0^{\mathbf{J}_5} = \{c_2, c_3, c_5\}$. With these refutation sets $\mathcal{R} = \{R_0^{\mathbf{J}_2}, R_0^{\mathbf{J}_3}, R_0^{\mathbf{J}_5}\}$ the JCU utility value of the 5 cases in $\widehat{B}_0$ for the agent A_1 can be assessed using JCU:

- $JCU(c_1, \widehat{B}_0) = \#(\emptyset) = 0/5 = 0.0$
- $JCU(c_2, \widehat{B}_0) = \#(\{R_0^{\mathbf{J}_2}, R_0^{\mathbf{J}_5}\}) = 2/5 = 0.4$
- $JCU(c_3, \widehat{B}_0) = \#(\{R_0^{\mathbf{J}_2}, R_0^{\mathbf{J}_3}, R_0^{\mathbf{J}_5}\}) = 3/5 = 0.6$
- $JCU(c_4, \widehat{B}_0) = \#(\emptyset) = 0/5 = 0.0$
- $JCU(c_5, \widehat{B}_0) = \#(\{R_0^{\mathbf{J}_5}\}) = 1/5 = 0.2$

Thus, the set of utility records of A_1 at round 0 is: $\mathcal{U}_0^{A_1} = \{\langle A_1, c_1, 0.0\rangle, \langle A_1, c_2, 0.4\rangle, \langle A_1, c_3, 0.6\rangle, \langle A_1, c_4, 0.0\rangle, \langle A_1, c_5, 0.2\rangle\}$.

In the same way, A_2 and A_3 compute their JCU utility values and send their utility records to A_1, that can now examine all the utility records to determine the winner. Table 8.5.a shows the utility values for the three agents: the winner is the agent A_1. The utility computed by the agent A_1 for case c_3 is the highest

one and therefore A_1 retains the case c_3, the case is not available any more, and the rest of agents are notified.

When A_2 and A_3 answer with an acknowledgment to A_1, A_1 sends again a message to A_2 and A_3 requesting for the utility records of the remaining cases $\widehat{B}_1 = \{c_1, c_2, c_4, c_5\}$ for the second round of the protocol. A_1 has to recompute its own JCU utility values since has retained a new case, and the new JCU utility values are shown in Table 8.5.b. This time there is a tie between A_2 and A_3 that is resolved randomly: the winner is A_2, that receives the case c_1 to be retained.

The third round for the cases $\widehat{B}_2 = \{c_2, c_4, c_5\}$ has JCU utility values that can be seen in Table 8.5.c, where the winner is A_3 that receives the case c_5.

In the fourth round, no agent wants any case in $B_3 = \{c_2, c_4\}$, as shown in Table 8.5.d where all the JCU utility values are zero. A_1 sends a message to A_2 and A_3 telling that the I_{CCB-CS} interaction protocol is over, the cases c_2 and c_4 are discarded, and the pools of the three agents are cleared.

One may think that if every agent has access to all the cases during the CCB-CS collaboration strategy, why isn't it the best policy to allow each agent to retain every case? In fact, allowing each agent to retain every case is not the best policy (as we are going to show in the experiments section), since the resulting system would be equivalent to a single agent (since as each agent would have all the cases). The experiments section will show how a group of agents using CCB-CS can outperform a single agent that has all the cases.

CCB-CS may appear to be a complex way to distribute the cases among the agents. However, (as we have said earlier) it is designed in this way since the order in which the cases are bargained does matter. A simpler sequential protocol that would consider the cases one at a time with a predefined order (such as a simple auction, where the cases will be auctioned one at a time with a predefined order) would lead to suboptimal results. Let us illustrate what would had happened in the exemplification if a sequential protocol is used:

1. The utility for the case c_1 is requested to all the agents, and the winner is the agent A_2. Therefore, A_2 retains c_1 and the new utility values for A_2 are (-, 0, 0, 0, 0).

2. Then, the utility for the case c_2 is asked. The winner is the agent A_1 and the new utility values for A_1 are (-, -, 0.33, 0, 0).

3. The case c_3 is assigned also to A_1, and his new utility values are (-, -, -, 0, 0).

4. finally, both c_4 and c_5 will be retained by A_3 and no case will be discarded.

Therefore, using this sequential protocol, agent A_1 would retain the cases c_2 and c_3, agent A_2 would retain the case c_1 and agent A_3 would retain the cases c_4 and c_5. Moreover, recall that one goal of CCB-CS is to minimize the number of cases retained while maintaining the competence of the case bases. Therefore, the sequential protocol is clearly outperformed by CCB-CS that covers all the errors retaining just 3 cases, while the sequential protocol needs 5 cases.

Moreover, we could also think that instead of this protocol, a combinatorial auction [73] could be used. A combinatorial auction is a multiple-item auction in which each bidder offers a price for a collection of items (of the bidder's choosing) rather than placing a bid on each item separately. The auctioneer selects a set of these combinatorial bids which raises the most revenue without assigning any items to more than one bidder. In our scenario, the utility values should be considered as the bids. However, this assumes that the bids for the different sets of items (sets of cases in our scenario) are independent, a property that is not satisfied by the utility of cases in CBR since the utility of new cases depend on the previously retained cases. For instance, in the example, the utility of the case c_2 for the agent A_1 at the beginning was 2, however, after retaining the case c_3 this utility dropped to zero.

8.4.3 Experimental Evaluation

In this section we are going to empirically evaluate the CCB-CS collaboration strategy. Specifically, we are going to compare the classification accuracy of agents solving problems individually and using the Committee collaboration strategy for a $\mathcal{M}$AC system composed of 5 agents. Moreover, we are also going to discuss results concerning the characteristics of case bases achieved using each one of the different collaboration strategies (including case base size, completeness, redundancy, and bias).

In order to test the generality of the collaboration strategies we are going to present results using three data sets: sponge, soybean and zoology. Moreover, in all the experiments reported in this section agents use LID as the learning method and all the presented results are the average of five 10-fold cross validation run.

In an experimental run the data set is divided into two sets: the training set, containing the 90% of the cases in the data set and the test set, containing the remaining 10% of the cases in the data set. At the beginning of the experiment, a 10% of the cases of the training set are distributed among the agents. Then, the rest of the cases in the training set are sent to the agents one by one (each case is only sent to one agent). Each time an agent receives a case of the training set, the agent's retention strategy is applied to decide retention. From time to time, the test set is sent to the agents to evaluate their classification accuracy. The experiment ends when all the cases in the training set have been sent to the agents. Moreover, in these experiments we have not forced any case base bias, since CCB-CS is not affected by it. To understand this fact, consider that each time CCB-CS is used, all the cases in the pools of the agents are joined together in order to start the bargaining, and thus it is not relevant which concrete agent received each individual case.

Moreover, results obtained by CCB-CS are compared with the results obtained by OFR-AO and ID-AO presented in Section 8.2. The comparison focuses on OFR-AO and IF-AO because they (together with CCB-CS) do not allow copies of cases.

We will first present experimental results concerning classification accuracy,

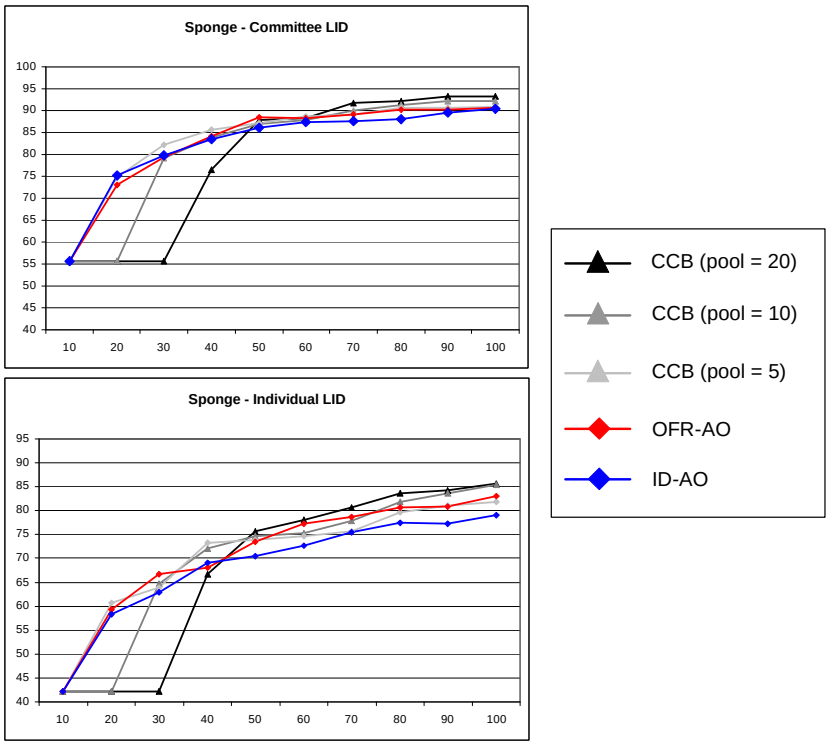


Figure 8.17: Classification accuracy of a 5 agent $\mathcal{M}$AC system using the CCB-CS collaboration strategy in the sponge data set.

and after that we will analyze the case base properties with the different retention strategies.

8.4.3.1 Classification Accuracy Evaluation

Figure 8.17 shows the classification accuracy evolution of the agents in a $\mathcal{M}$AC system composed of 5 agents using CCB-CS in the sponge data set. Both individual and committee classification accuracy is shown: right hand side plot shows the individual classification accuracy and left hand side plot shows the committee classification accuracy. Five plots are shown: three of them correspond to agents using CCB-CS with pools of delayed retention cases of size 5, 10 and 20 respectively, and the other two correspond to agents using OFR-AO and ID-AO.

Let us first consider the committee results (the left hand size plot of Figure 8.17). Analyzing the effect of the pool size in CCB-CS we observe that a large pool size (i.e. 20) has two effects: first, the learning plot of the agents does not start growing from the beginning (since it is delayed retention, and the agents do not retain any case until one of them has its pool full); and second, when

the plot starts growing, it grows faster and reaches higher classification accuracy values than the plots of agents with smaller pools. For instance, with a pool size of 5, agents increase their accuracy value from the beginning of the experiment, and reach a classification accuracy value of 90.86%, while agents using a pool size of 20 do not start increasing their accuracy until a 30% of the training set is sent, but reach an accuracy value of 93.21% at the end of the experiment. Thus, having a large pool is better in the long term since after the initial delay, the learning curve grows faster and achieves higher classification accuracy values.

Comparing CCB-CS with OFR-AO and ID-AO (which do not perform delayed retention) the classification accuracy of the agents start increasing from the beginning of the experiments, while CCB-CS requires that the pool of some agent is full to start case retention. Thus, at the beginning OFR-AO and ID-AO achieve higher classification accuracy values, but as more cases of the training set are sent to the agents, CCB-CS quickly catches up OFR-AO and ID-AO. At the end of the experiments, both CCB-CS with pool sizes of 10 and 20 achieve higher classification accuracy values than OFR-AO and ID-AO. Thus CCB-CS is able to achieve higher classification accuracy values than OFR-AO and ID-AO; the reason is that CCB-CS does not discard cases as quickly as OFR-AO or ID-AO. In fact, the larger to pool size, the less eager that CCB-CS is discarding cases, and thus the higher classification accuracy value is achieved.

Considering now the individual classification accuracy results on the right hand side plot of Figure 8.17 we observe results similar to those of the committee: larger pools produce a larger delay on retaining cases, but they also produce higher classification accuracy values.

Figure 8.18 shows the classification accuracy evolution of the agents in a $\mathcal{M}$AC system composed of 5 agents using CCB-CS in the soybean data set. Figure 8.18 shows very similar results to that of the sponge data set: the larger the pool size, the longer the delay, but also the higher the accuracy value at the end of the experiments. Notice that both in the sponge and soybean data sets, using CCB-CS with a pool size of 20, the classification accuracy of the committee is higher than the accuracy of OFR-AOC and ID-AOC (the versions of MAR-CS that allowed multiple copies of cases): for instance, in the soybean data set, CCB-CS with a pool size of 20 reaches an accuracy value of 87.39% while OFR-AOC reaches an accuracy of 86.97% and OFR-AO an accuracy of 81.03%.

Figure 8.19 shows the classification accuracy evolution of the agents in a $\mathcal{M}$AC system composed of 5 agents using CCB-CS in the zoology data set. The results obtained in the zoology data sets are similar to those obtained in sponge and zoology. However, notice that the zoology data set contains only 101 cases: 10 or 11 cases will constitute the test set and the rest of 90 or 91 cases will constitute the training set. If the pool size of the agents is 20, it may happen that no agent has its pool of cases full when the experiment finishes (19 cases per agent in a 5 agent system means that with 95 cases the pools of the agents may still not be full). This can be seen in Figure 8.19, where the agents do not start retaining cases until the 90% of the training set has been sent. Thus, we can conclude that the zoology data set is too small a data set to use CCB-CS

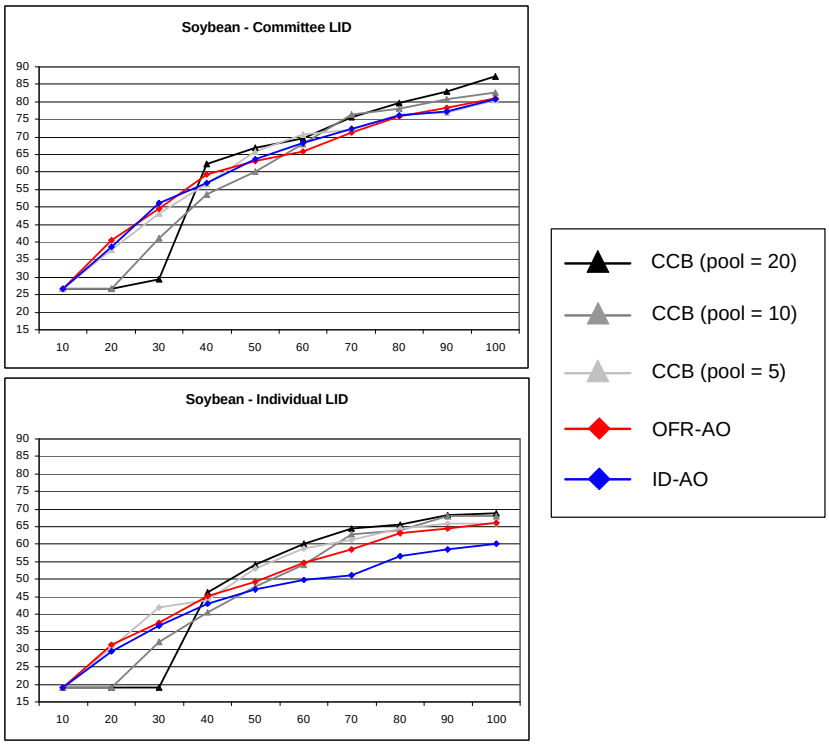


Figure 8.18: Classification accuracy of a 5 agent $\mathcal{MAC}$ system using the CCB-CS collaboration strategy in the soybean data set.

properly. However, as we have previously said, the tendencies observed show that if the zoology data set was larger, the obtained results would have been similar to those obtained in the sponges or soybean data sets.

8.4.3.2 Case Base Evaluation

Table 8.6 shows the case base properties of the CBR agents at the end of the experimental runs for the sponge data set. Each row of the table shows the results for a different collaboration strategy. The columns show the following properties of the case base: the accuracy achieved by individual agents, the accuracy achieved by the committee of agents, the average size of individual case bases, and the average Committee Completeness ($\mathbb{C}$), Committee Redundancy ($\mathbb{R}$) and Committee Bias ($\mathbb{B}$). Moreover, all the values shown in Table 8.6 correspond to the end of an experimental run. Table 8.6 shows that case base size, $\mathbb{C}$, $\mathbb{R}$ and $\mathbb{B}$ do not change appreciably by varying the pool size: agents using CCB-CS end the experiments with $\mathbb{C} = 0.68$, $\mathbb{R} = 0$ and $\mathbb{C} = 0.22$ approximately, independently of the pool size used. Moreover, notice that the case base size achieved by the agents using CCB-CS is similar to that of the

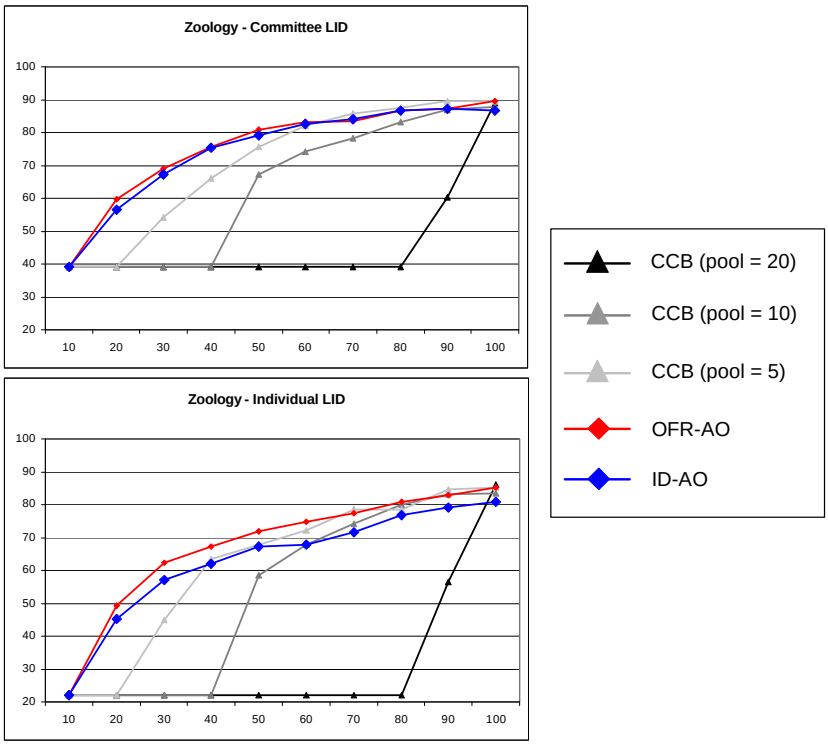


Figure 8.19: Classification accuracy of a 5 agent $\mathcal{MAC}$ system using the CCB-CS collaboration strategy in the zoology data set.

agents using OFR-AO. The only remarkable difference is that Committee Bias is higher with CCB-CS than with OFR-AO or ID-AO. This is a surprising result, since we would expect that a higher Committee Bias implies a lower classification accuracy value.

The explanation is that the *ensemble space* analysis ($\mathbb{C}$, $\mathbb{R}$ and $\mathbb{B}$) is only a characterization of many possible committees, that serves as a heuristic that helps us to predict how a given committee will perform. In fact, there are many different committees with equal characterization but with different performance, and the ensemble space analysis gives us the average results among all the possible committees with the same characterization. Moreover, CCB-CS makes use of JCU to decide which agent retains each case, and thus a deeper analysis of the case bases of the agents is made than the one provided by the ensemble space characterization. The result is that CCB-CS is able to find a distribution of cases that works better than the average predicted by the ensemble space analysis. However, the ensemble space is still a good tool for analyzing the expected (average) behavior of committees.

Table 8.7 shows, for agents that use CCB-CS, OFR-AO and ID-AO, the

	Individual	CCS	CB size	$\mathbb{C}$	$\mathbb{R}$	$\mathbb{B}$
CCB-CS (pool = 20)	85.71%	93.21%	34.78	0.69	0.00	0.22
CCB-CS (pool = 10)	85.36%	92.07%	34.13	0.68	0.00	0.21
CCB-CS (pool = 5)	81.78%	90.86%	34.27	0.68	0.00	0.22
OFR-AO	83.07%	90.71%	34.24	0.68	0.00	0.17
ID-AO	79.14%	90.50%	29.20	0.58	0.00	0.19

Table 8.6: Case Base properties of agents using CCB-CS in the sponge data set.

	Individual	CCS	CB size	$\mathbb{C}$	$\mathbb{R}$	$\mathbb{B}$
CCB-CS (pool = 20)	68.70	87.39	44.35	0.81	0.00	0.19
CCB-CS (pool = 10)	68.11%	82.73%	44.89	0.82	0.00	0.20
CCB-CS (pool = 5)	65.78%	80.39%	46.70	0.86	0.00	0.21
OFR-AO	69.38%	81.04%	48.05	0.87	0.00	0.12
ID-AO	60.00%	80.85%	40.12	0.73	0.00	0.17

Table 8.7: Case Base properties of agents using CCB-CS in the soybean data set.

main case base properties at the end of the experiments in the soybean data set. Table 8.7 shows that the results obtained in the soybean data set are very similar to those obtained in the sponge data set: the case base size of CCB-CS is very similar to that of OFR-AO and ID-AO, while the classification accuracy achieved using large pool sizes (10 or 20) is higher than that of both OFR-AO and ID-AO. Moreover, notice that CCB-CS using a pool size of 20 also outperforms OFR-AOC (that performed impressively well in the soybean data set, as we saw on Table 8.2), since CCB-CS with pool size of 20 achieves a committee accuracy of 87.39% with a case base size of 44.35 cases per agent in average, while OFR-AOC achieves an accuracy value of 86.97% with an average case base size of 85.94 cases. Thus CCB-CS achieves higher accuracy with a much smaller case base size.

Finally, Table 8.8 shows, for agents that use CCB-CS, OFR-AO and ID-AO, the main case base properties at the end of the experiments in the zoology data set. As we have said before, the zoology data set is very small and, since at

	Individual	CCS	CB size	$\mathbb{C}$	$\mathbb{R}$	$\mathbb{B}$
CCB-CS (pool = 20)	86.13%	89.11%	9.81	0.54	0.00	0.30
CCB-CS (pool = 10)	83.56%	87.72%	9.52	0.52	0.00	0.32
CCB-CS (pool = 5)	85.31%	89.70%	10.67	0.59	0.00	0.31
OFR-AO	85.35%	89.61%	10.86	0.60	0.00	0.23
ID-AO	80,79%	86.73%	9.76	0.54	0.00	0.26

Table 8.8: Case Base properties of agents using CCB-CS in the zoology data set.

the end of the experiments the agents will surely have some cases in their pools that still have to be considered for retention, the amount of cases in the pools of the agents may be large compared with the size of the data set. A direct effect of this can be seen in Table 8.8 where the average case base size of agents with pool sizes of 10 and 20 is smaller than the case base size with pools of size 5 (while we have seen in the sponges and soybean data sets that the average number of cases retained does not depend on the size of the pools). This is the reason that the accuracy achieved using CCB-CS is not higher than that achieved with OFR-AO or with ID-AO.

Summarizing the experimental results, we can conclude that CCB-CS is an effective collaboration strategy for case retention, and that completely avoids the problem of case base bias (since all the cases received by all the agents are put together before starting the bargaining). We have also seen that depending on the size of the pool of delayed retention cases CCB-CS has a different effect: larger pool delay the retention and can cause problems if the agents receive few cases (as in the zoology data set), but large pools have the advantage of not discarding cases very eagerly and therefore decide retention with more information, achieving higher classification accuracy values. Thus CCB-CS can outperform retention strategies such as OFR-AO or ID-AO that decide retention case by case, and that discard cases more eagerly. Moreover, CCB-CS with a large pool size, can also outperform strategies such as OFR-AOC and ID-AOC that allow copies of cases.

8.5 Conclusions

In this chapter we have presented several case retention strategies. We have presented strategies based on two main ideas: the first idea is to take into consideration the multi-agent aspect of case retention and the second idea is to use justifications in order to decide retention. Specifically, we have presented:

- The MAR-CS collaboration strategy (with eight different versions of it: NR-NO, AR-NO, OFR-NO, ID-NO, OFR-AO, ID-AO, OFR-AOC and ID-AOC), that takes into consideration the multi-agent aspect of case retention.
- The JUST case base reduction technique, that uses justifications to decide retention for individual CBR systems.
- And finally CCB-CS, that combines both ideas (multi-agent retention and justifications).

Retention strategies can be classified in several ways: they can be classified depending on if they are *addition case retention* strategies or *deletion case retention* strategies; they can also be classified depending on whether they treat cases one by one or in batches; and finally they can also be classified depending on if they are *on-line* case retention strategies or *off-line* case retention strategies.

All the strategies presented in this chapter are addition case retention strategies (since they decide which cases to add to the case base and not which to delete). Notice that JUST is a case addition strategy, since JUST starts with an empty case base and selects the cases to be added to that case base. Moreover, all the MAR-CS versions consider cases one by one (1-by-1 strategies), while JUST and CCB-CS consider cases in batches (batch strategies). The main difference between them is that 1-by-1 strategies are more sensitive to the order in which the cases arrive to the agents, while batch strategies are less sensitive (since they do not discard cases as eagerly). Finally, MAR-CS and CCB-CS are on-line case retention strategies (since they are used during the normal operation of the CBR agent), and JUST is an off-line strategy (since it is applied as a background process by the agents and not during problem solving). Moreover, on-line and off-line strategies are complementary, i.e. an agent can use an on-line case retention strategy (such as CCB-CS), and also use JUST off-line from time to time to reduce the size of its case base.

In the experimental evaluation, we have shown that those collaboration strategies that offer cases to other agents outperform those collaboration strategies that do not offer cases to other agents (since OFR-AO, ID-AO and OFR-AOC systematically outperform OFR-NO and ID-NO). Moreover, in some data sets (such as soybean) allowing multiple copies of cases (like OFR-AOC and ID-AOC do), and thus moderately increasing committee redundancy, can also improve classification accuracy (albeit at the cost of also increasing the average case base size of the agents).

Concerning the use of justifications and the JCU measure to decide which cases to retain, they also improve the retention process of individual CBR systems. Specifically, we have presented JUST, that uses JCU to determine which are the cases in a case base that are interesting to be retained and which can be discarded. The experimental results have shown that JUST can effectively reduce the size of a case base while maintaining (and even improving) the classification accuracy of a case base.

The CCB-CS collaboration strategy (that also uses JCU) outperforms all the versions of the MAR-CS collaboration strategy. The reason is that the use of justifications in combination with delayed retention allows the agents to accurately assess the utility of cases, and thus achieve a distribution of cases among the agents better than the one achieved by MAR-CS. Moreover, CCB-CS uses delayed retention, i.e. cases to be retained are stored in a pool; the retention process does not start until the pool of cases of an agent in the $\mathcal{M}$AC system is full. This can be a problem only in situations where the agents receive very few cases to retain (as we have shown in the zoology data set). Thus, for a data set that is large enough, we have shown that the larger the pool size of the agents, the better CCB-CS performs. The reason is that the larger the pools, the better the utility estimation made by JCU(since the larger the pools, the larger the exam used by JCU).

Another reason in which justifications help CCB-CS to outperform MAR-CS is the following one: agents using CCB-CS retain only those cases having the highest utility values. Moreover, a case has a high JCU utility when it can

prevent an agent for making many errors in the future. Thus, retaining a single case with high utility means it is able to fix several errors, and this can only be detected using justifications. An agent using MAR-CS (for instance using OFR-AO) would retain a new case every time that the agent solves a problem incorrectly. Therefore, an agent using MAR-CS retains a case for each error detected, while an agent using CCB-CS is able to retain cases that fix several errors (thus achieving a higher reduction).

From the experiments presented in this chapter we can draw three main conclusions: the first one is that collaboration during retention improves the performance of CBR agents; the second one is that discarding cases eagerly in the retention process has a negative effect, and thus batch retention strategies outperform 1-by-1 strategies; finally, the information provided by justifications is useful to assess the utility of cases, and thus is useful to case retention; this has been shown in the experiments since retention strategies based on justifications outperform those strategies that do not use justifications.

Finally, in Chapter 4 we saw that a moderate degree of Committee Redundancy can improve the classification accuracy of a committee. This effect takes place in the MAR-CS collaboration strategy applied to the the soybean data set. However, the CCB-CS collaboration strategy does not allow copies of cases, and thus does not allow any increase of Committee Redundancy. For this reason, future work should consider improving CCB-CS so that it can take into account case copy strategies in order to achieve a certain degree of redundancy.

Chapter 9

Case Bartering Collaboration Strategies

This chapter presents the *Case Bartering Collaboration Strategies* (CB-CS), with which a committee of agents can perform a redistribution of the cases in their case bases (by means of a regulated process of case bartering), with the goal of improving their performance, both as individual agents and as a committee.

9.1 Introduction

In previous chapters we have shown that committees of agents can outperform individual agents in terms of classification accuracy. However, a committee has to satisfy some conditions in order to have a good classification accuracy. Specifically, Chapter 4 has presented the *Ensemble Space* as a way to characterize a committee using three indicators: *Committee Completeness* ($\mathbb{C}$), *Committee Bias* ($\mathbb{B}$) and *Committee Redundancy* ($\mathbb{R}$). Chapter 4 experimentally shows that a committee must have a high committee completeness, a low committee bias and a low (but greater than zero) committee redundancy in order to achieve a high classification accuracy. Those results have motivated some of the retention techniques presented in the previous chapter. These techniques tried to achieve case bases that satisfy the conditions needed for the committee to have a high classification accuracy by means of a collaborative process on the cases to be retained. However, case retention techniques presented in the previous chapter can only be applied when a committee of agents evolves together, i.e. they collaborate during their training; if a committee contains agents that have not had the opportunity to use collaborative case retention strategies (because they have learned individually), other strategies must be used for the committee to achieve high classification accuracy values. In this chapter we are going to present the *Case Bartering* collaboration strategies, designed specifically for such situations.

The goal of case bartering is to achieve a redistribution of cases among the

agents that improves both the performance of the individual agents and the committee as a whole. The idea behind case bartering is that the utility of individual cases is not the same for all the agents in a $\mathcal{MAC}$ system, i.e. there are cases that can have a high utility for some agents that are useless for some other agents (because they may be redundant with the cases already present in their case base, or for any other reason). Thus, case bartering is a collaboration strategy that is used by the agents in order to reach bartering agreements so that an agent cedes cases with low utility in exchange of cases with higher utility for itself to another agent (that accepts because it has the complementary utility values for the exchanged cases). For example, if an agent A_1 in a $\mathcal{MAC}$ system has retained a large amount of cases of a certain class S_1, but has troubles in finding cases of another class S_2, it may happen that there is another agent A_2 in the $\mathcal{MAC}$ system that is in the opposite situation (with a lot of cases of S_2 and very few of S_1); thus, it will be beneficial for both agents if they reach an agreement to trade cases so that both A_1 and A_2 can get cases of the class that they had troubles to find cases of.

In order to use the case bartering collaboration strategy, agents need a policy to decide which bartering agreements to reach. In this chapter we are going to present two different decision policies to reach bartering agreements: the first one will be based in the ensemble space, and the second one in justifications. Each set of decision policies defines in fact a different case bartering collaboration strategy.

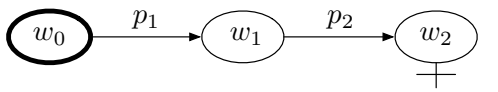
The structure of the chapter is as follows. First, Section 9.2 formally defines the case bartering collaboration strategy. Section 9.3 presents the bias based decision policies (inspired in the ensemble space). Then, Section 9.4 presents the case bartering interaction protocol and Section 9.5 presents an exemplification of the case bartering collaboration strategy. After that, Section 9.6 presents an alternative set of decision policies for case bartering based on justifications. Finally, Section 9.7 presents the experimental evaluation of the case bartering collaboration strategy. The chapter ends with the conclusions section.

9.2 The Case Bartering Collaboration Strategy

This section introduces the Case Bartering Collaboration Strategy (CB-CS), that allows a group of agents to reach agreements to barter cases in order to achieve higher individual and committee performance. CB-CS is an iterative collaboration strategy composed of a series of rounds. Each round is divided in two steps:

- Information Gathering,
- Case Bartering.

During the Information Gathering step, the agents will collect information in order to be able to know which cases to barter during the Case Bartering step. During the Case Bartering step, agents will send offers to each other to



p_1 : $Request(?A_i, ?A_j, distribution)$

p_2 : $Inform(!A_j, !A_i, \{?d_{j,1}, ..., ?d_{j,K}\})$

Figure 9.1: Interaction protocol used by the agents in the D_{BG} decision policy.

barter cases (that we will call *bartering offers*) that can be accepted or rejected. If an agent accepts an offer to barter cases received from another agent, the cases are actually exchanged in the Case Bartering step.

Definition 9.2.1. *The* Case Bartering Collaboration Strategy *(CB-CS) is a collaboration strategy* $\langle I_{CB-CS}, D_G, D_B, D_A \rangle$*, where* I_{CB-CS} *is the CB-CS interaction protocol shown in Figure 9.3,* D_G *is the decision policy used by the agents to gather information in the Information Gathering step,* D_B *is the decision policy used by the agents to decide which bartering offers to generate, and* D_A *is the decision policy used to decide which bartering offers to accept.*

In the remainder of this chapter we will define two different sets of decision policies: first we will define a set of decision policies (D_{BG}, D_{BB}, and D_{BA}) based on the ensemble space, while later at Section 9.6 we will introduce another set of decision policies (D_{JG}, D_{JB}, and D_{JA}) based on justifications. Moreover, in some situations, the definition of those decision policies will involve the definition of a sub-protocol of I_{CB-CS}. Finally, we will present the interaction protocol I_{CB-CS}.

9.3 Bias Based Case Bartering

This section presents a set of decision policies for case bartering based on the ensemble space. Specifically, the policies presented in this section have the goal of minimizing Committee Bias ($\mathbb{B}$). Recall (see Section 4.4) that committee bias measures the average case base bias of the individual agents, and that case base bias measures the degree in which a case base has some areas of the problem space undersampled (while other parts may be oversampled). Therefore, as Chapter 4 shows, diminishing bias the classification accuracy is expected to increase.

The set of decision policies presented in this section define the *bias based case bartering* collaboration strategy (*CB-CS bias*).

9.3.1 Bias Based Information Gathering

The goal of the Information Gathering step of CB-CS is to obtain information on order to be able to generate bartering offers in the Case Bartering step.

Therefore, during Information Gathering the agents must obtain information in order to be able to estimate their case base bias and the committee bias, so that proper bartering offers can be generated. An agent A_i in a committee $\mathcal{A}$ that wants to estimate both its individual bias and the committee bias needs the following information:

- The distribution of cases among the different solution classes (i.e. how many cases of each solution class) that each agent in $\mathcal{A}$ has,
- The real distribution of cases among the different solution classes in the application domain.

The distribution of cases of each individual agent is easy to obtain. Specifically, given the set of possible solution classes $\mathcal{S} = \{S_1, ..., S_K\}$, an agent has to know:

$$d_{j,k} = \frac{\#(c \in C_j | c.S = S_k)}{\#(C_j)}$$

for each agent $A_j \in \mathcal{A}^c$. Where $d_{j,k}$ represents the fraction of cases of the case base of A_j that belong to the solution class S_k. Therefore, during during the Information Gathering step, every agent in $\mathcal{A}$ can compute and share this information.

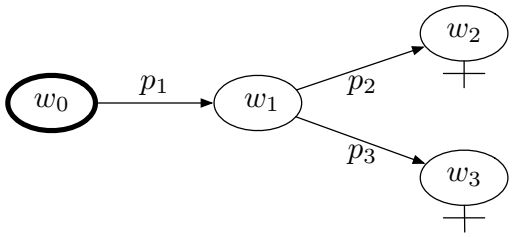
However, there is no obvious way in which an agent A_i can compute which is the real distribution of cases in the domain. Therefore, in the policies presented in this section, the agents will estimate such distribution. To make this estimation, the agents will assume two facts: the first one is that the case bases of the agents have no overlapping (i.e. that committee redundancy is $\mathbb{R} = 0$) and that aggregating the case distributions of the agents in $\mathcal{A}$ is a good estimation of the distribution of cases in the application domain. Therefore, if an agent knows the distribution of cases among the rest of agents in the committee, it can compute $D_k = \sum_{j=1...n} d_{j,k}$ for each solution class, and assume that the distribution of cases in the application domain is:

$$d_k = \frac{D_k}{\sum_{l=1...K} D_l}$$

With this estimation, an agent A_i can estimate its own case base bias and the committee bias using equations presented in Definitions 4.4.3 or 4.4.4, as follows:

$$\mathbb{B}(C_i) \simeq \sqrt{\sum_{m}^{k=1} (d_{i,k} - d_k)^2}$$

The idea behind this estimation is that each agents' case base can be considered an independent sample of cases of the problem space. Thus, if we aggregate such samples, the resulting sample is likely to be less biased than the individual



p_1 :	$Request(?A_i, ?A_j, ?\mathbf{O})$
p_2 :	$Inform(!A_j, !A_i, accept)$
p_3 :	$Inform(!A_j, !A_i, reject)$

Figure 9.2: Interaction protocol used by the agents in the D_{BB} decision policy.

sample. Thus, computing the bias of an agent with respect to the aggregated sample is likely to be a good estimation of the real bias of that individual agent.

We can now define the policy used by the agents in the Information Gathering step:

Definition 9.3.1. *The* Bias based Information Gathering *decision policy D_{BG} used by an agent A_i in CB-CS consists of requesting the values $d_{j,k}$ for each solution class $S_k \in \mathcal{S}$ to each agent $A_j \in \mathcal{A}$ during the Information Gathering step using the interaction protocol shown in Figure 9.1.*

9.3.2 Bias Based Case Bartering

During the Case Bartering step agents are allowed to send bartering offers to other agents. A bartering offer communicates an offer to barter two sets of cases between two agents. In this section we are going to define bartering offers that exchange a single case for another case, and also define a decision policy that can be used by an agent in a $\mathcal{MAC}$ system to generate such bartering offers.

Definition 9.3.2. *A* Bartering Offer $\mathbf{O} = \langle S_O, S_R, A_O, A_R \rangle$ *is a tuple where an* offeror *agent A_O is offering the* receiver *agent A_R a case of the class S_O in exchange of a case of the class S_R.*

Assuming that the goal of an agent A_i participating in the CB-CS collaboration strategy is to decrease its own case base bias, the bartering offers that A_i is interested in accepting are those that would decrease its own case base bias. Moreover, only those offers that would reduce the case base bias of the receiver agent are likely to be accepted by the receiver. Thus, we are going to present a method with which an agent can generate all those bartering offers that will reduce its case base bias and that are likely to be accepted by the receiver agent.

Moreover, notice that in the real world, bartering involves the exchange of two physical goods, but the agents in a $\mathcal{M}$AC system barter with cases that are just information. Thus, agents in a $\mathcal{M}$AC system have two options: the first one is to barter with cases as if they where physical goods, i.e. when an agent A_i cedes a case c_1 to an agent A_j in exchange of a case c_2, A_i deletes c_1 from its case base and A_j deletes c_2 from its case base; this is what we call the *non-copy mode*. And the second option is to barter copies of the cases, i.e. agents do not delete the cases they give to the other agents when bartering; this is what we call the *copy mode*. There are fundamental differences on whether the agents barter cases in the copy or non-copy modes:

- In the *non-copy mode* agents only perform a redistribution of cases among their case bases. Thus, only case base bias and committee bias are affected by the bartering process. Moreover, individual case base completeness is not affected and the number of cases in the case base of each agent will not vary.
- In the *copy mode* agents are increasing the committee redundancy since copies of the cases are made. Moreover, individual case base completeness also increases with each barter.

Thus, we are going to define two different decision policies, one for the non-copy mode and another for the copy mode.

Definition 9.3.3. *The* Bias based Non-Copy Case Bartering *decision policy D_{BB-NC} used by an agent A_i in CB-CS consists of 4 steps:*

1. *For each pair of solution classes $S_O \in \mathcal{S}$ and $S_R \in \mathcal{S}$, such that $S_O \neq S_R$ and such that A_i has at least one case of S_O in its case base.*
2. *A_i checks if adding one case with solution class S_R and deleting a case with solution class S_O from C_i will decrease its individual bias.*
3. *If so, A_i checks if there is another agent A_j in the system in the opposite situation (i.e. that deleting one case with solution class S_R and adding a case with solution class S_O from C_j will decrease the individual case base bias of A_j).*
4. *If that agent exists, a bartering offer $\mathbf{O} = \langle S_O, S_R, A_i, A_j \rangle$ is generated and sent to A_j using the interaction protocol shown in Figure 9.2.*

Definition 9.3.4. *The* Bias based Copy Case Bartering *decision policy D_{BB-C} used by an agent A_i in CB-CS consists of 4 steps:*

1. *For each pair of solution classes $S_O \in \mathcal{S}$ and $S_R \in \mathcal{S}$, such that $S_O \neq S_R$ and such that A_i has at least one case of S_O in its case base.*
2. *A_i checks if adding one case with solution class S_R to C_i will decrease its individual case base bias.*

3. *If so, A_i checks if there is another agent A_j in the system in the opposite situation (i.e. that adding a case with solution class S_O to C_j will decrease the individual case base bias of A_j).*

4. *If that agent exists, a bartering offer $\mathbf{O} = \langle S_O, S_R, A_i, A_j \rangle$ is generated and sent to A_j using the interaction protocol shown in Figure 9.2.*

Agents using both D_{BB-C} and D_{BB-NC} use the interaction protocol shown in Figure 9.2 in order to communicate the bartering offer to other agents. The protocol works as follows:

1. In state w_0, A_i sends a message p_1 to A_j containing a bartering offer $\mathbf{O}$.

2. In state w_1, A_j uses its D_{BA} decision policy to decide whether to accept the bartering offer or not:

   - If A_j accepts, a message p_2 is send to A_i, the protocol ends, and the cases are actually exchanged.
   - If A_j does not accept, a message p_3 is send to A_i and the protocol ends.

Thus, the bias based case bartering strategy (*CB-CS bias*) an work on the copy mode or in the non-copy mode. In the remaining of this chapter we will note *CB-CS bias copy* and *CB-CS bias non-copy* to make reference to the bias based bartering strategy working in the copy or non-copy modes.

We would like to remark that more complex case offering generation decision policies could be designed inspired in the ensemble space. However, our goal is just to show that case bartering using decision policies based on the ensemble space can improve the performance of committees of agents, and the simple decision policies defined in this section are sufficient for this purpose.

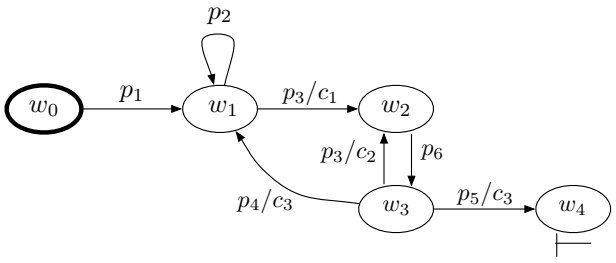
9.3.3 Bias Based Offer Acceptance

When an agent A_i receives a bartering offer, A_i requires a decision policy to decide whether to accept the offer or not.

Definition 9.3.5. *The* Bias Based Acceptance *decision policy D_{BA} used by an agent A_i determines to accept a bartering offer $\mathbf{O}$ only if it will reduce the the individual case base bias of A_i:*

$$D_{BA}(\mathbf{O}) = \begin{cases} true & if\, \mathbb{B}(C_i') < \mathbb{B}(C_i) \\ false & otherwise \end{cases}$$

where C_i' is the case base that A_i will have after performing the barter specified in $\mathbf{O}$.



p_1 : $Request(?A_c, all, ?n = \#(\mathcal{A}) \wedge ?R_{max} \wedge newround)$

p_2 : $Inform(?A_i, !A_c, finished)$

p_3/c_1 : $Request(?A_c, ?A_i, token)/$
$|!w_1 w_1 A_i| = !n$

p_3/c_2 : $Request(?A_c, ?A_i, token)/$
$|!w_1 w_3 A_i| < !n - 1$

p_4/c_3 : $Request(!A_c, all, newround)/$
$|!w_1 w_1 A_3| = !n - 1$

p_5/c_3 : $Inform(!A_c, all, terminate)$
$|!w_1 w_1 A_3| = !n - 1$

p_6 : $Inform(!A_i, !A_c, success/failure)$

Figure 9.3: The CB-CS interaction protocol.

Notice that if the agents in a $\mathcal{MAC}$ system use the decision policies defined in this section, all the offers will be accepted, since the D_{BB-C} and D_{BB-NC} decision policies only generate those offers that will be accepted by the receiver agent. However, a decision policy for offer acceptance is needed, since an agent cannot assume that all the offers received will be favorable.

The next section explains how these three decision policies are used in the case bartering protocol. And Section 9.5 presents an exemplification of how CB-CS works using these decision policies.

9.4 The Case Bartering Interaction Protocol

This section presents the interaction protocol I_{CB-CS} used in the CB-CS collaboration strategy. I_{CB-CS} is an iterative protocol that consists of a series of rounds. Each round consists of two steps: *Information Gathering*, and *Case Bartering*.

During the Information Gathering step, the agents use their D_G decision policy to obtain information that will be used during the Case Bartering step by their D_B decision policy to generate bartering offers.

At the beginning of each round, all the agents perform the Information Gathering step at the same time. Then, the agents perform the Case Bartering

step one agent at a time. The protocol uses a token-passing mechanism and only the agent who owns the token will perform the Case Bartering step. At the beginning of each round the token is owned by the convener agent, and once the owner of the token has finished the Base Bartering step, the token is given to another agent, and so on until all the agents in the committee have owned the token in the current round. Once all the agents have performed the Information Gathering and Case Bartering steps, a new round starts. The protocol will last until a predefined number of rounds t_{max} have been made or when there have been no accepted bartering offers between agents in the committee in a given round.

I_{CB-CS} is shown in Figure 9.3 and works as follows for a committee of agents $\mathcal{A}$ (assuming that an agent $A_c \in \mathcal{A}$ acts as the convener agent and starts the protocol):

1. w_0 is the initial state, and the protocol starts when the convener agent A_c sends message p_1 to the rest of agents in $\mathcal{A}$ stating that the I_{CB-CS} starts. With message p_1, A_c also informs the rest of agents which is the committee of agents $\mathcal{A}$ that intervene in the protocol and the maximum number of rounds R_{max}.

2. In state w_1 each agent uses its own Information Gathering decision policy D_G in order to obtain information. In fact, state w_1 corresponds to the *Information Gathering* step of CB-CS. An agent confirms to A_c that has finished its Information Gathering step by sending a message p_2 to A_c.

3. When all agents have confirmed to A_c that they have finished the Information Gathering step (and A_c has also finished), A_c selects one agent $A_i \in \mathcal{A}$ and sends a message p_3 to him stating that A_i is allowed to make bartering offers (i.e. A_c "gives" the token to A_i).

4. In state w_2, an agent A_i owns the token, and uses its own Case Bartering decision policy D_B in order to send bartering offers. State w_2 corresponds to the *Case Bartering* step of CB-CS, and the sub-protocol associated with D_B is used (see Section 9.3.2 and Section 9.6.2). Once A_i has finished sending bartering offers, it sends a message p_6 to the convener agent A_c stating if it has succeeded in making some bartering (if A_i has achieved to make at least one case bartering, then it has succeeded; otherwise it has failed).

5. In state w_3, if there are some agents that have not yet owned the token in this round of the protocol, A_c selects one of them and sends it the token with a message p_3. When all the agents in the committee have owned the token once (including the convener agent), A_c decides if a new round has to take place. If all the agents have failed in the Case Bartering step or if the maximum number of rounds has been reached, the protocol will end; A_c will send a termination message p_5 to the rest of agents and the protocol will move to state w_4 that is a final state. Otherwise, A_c will sent

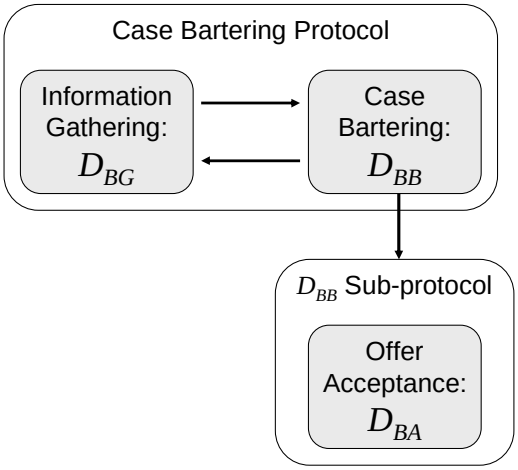


Figure 9.4: Relation between the protocols and decision policies used in bias based case bartering.

message p_4 to the rest of agents, and the protocol moves again to state w_1 since a new round will take place.

Finally, figure 9.4 shows the relation between all the protocols and decision policies involved in the justification based case bartering. Figure 9.4 shows that the main case bartering protocol requires two decision policies D_{BG} and D_{BB}. Moreover, the D_{BB} decision policy engages a sub-protocol that requires an additional decision policy D_{BA}.

The next section presents an exemplification of the CB-CS collaboration strategy.

9.5 Exemplification

In this section we are going to visualize the effect of case bartering using a two dimensional data set, shown in Figure 9.5. The example data set has three solution classes: *red*, *blue*, and *green*, and there are 64 red cases, 32 blue cases and 32 green cases. For this example, we are going to consider *CB-CS bias non-copy*.

In the exemplification we are going to consider a committee composed of 4 agents $\mathcal{A} = \{A_1, A_2, A_3, A_4\}$. Each one of those agents is going to receive a biased sample of the data set. Specifically, the samples of cases received by the 4 agents are shown in Figure 9.6: A_1 has 12, 4, and 8 cases of the blue, green, and red classes respectively; A_2 has 6, 6, and 30 cases of each class; A_3 has 4,9, and 20; and finally A_4 has 10, 13, and 6 cases of each class. Thus, with those numbers, the bias of the individual agents can be estimated:

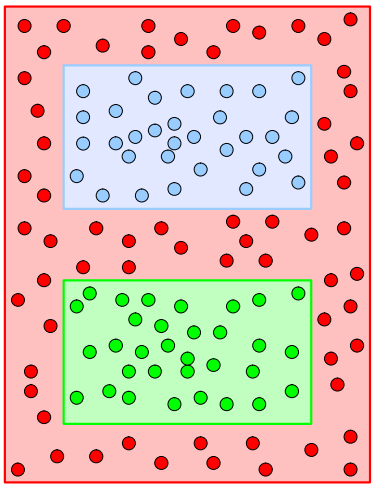


Figure 9.5: Two dimensional data set used in the bartering exemplification

$$\mathbb{B}(A_1) = \sqrt{\left(\frac{12}{24} - \frac{16}{64}\right)^2 + \left(\frac{4}{24} - \frac{16}{64}\right)^2 + \left(\frac{8}{24} - \frac{32}{64}\right)^2} = 0.3118$$

$$\mathbb{B}(A_2) = \sqrt{\left(\frac{6}{42} - \frac{16}{64}\right)^2 + \left(\frac{6}{42} - \frac{16}{64}\right)^2 + \left(\frac{30}{42} - \frac{32}{64}\right)^2} = 0.2624$$

$$\mathbb{B}(A_3) = \sqrt{\left(\frac{4}{33} - \frac{16}{64}\right)^2 + \left(\frac{9}{33} - \frac{16}{64}\right)^2 + \left(\frac{20}{33} - \frac{32}{64}\right)^2} = 0.1684$$

$$\mathbb{B}(A_4) = \sqrt{\left(\frac{10}{29} - \frac{16}{64}\right)^2 + \left(\frac{13}{29} - \frac{16}{64}\right)^2 + \left(\frac{6}{29} - \frac{32}{64}\right)^2} = 0.3664$$

The average Committee Bias is $\mathbb{B} = 0.2773$, quite a high value for a committee bias. Figure 9.7 shows how a decision tree algorithm used to index the cases in the case base would partition the problem space. Figure 9.7 shows that agents A_1 and A_4 (the left-most, and right-most ones in the figure) show the larger deviations from the original data set, as expected by its high individual case base bias.

Now imagine that we allow the four agents to use the CB-CS collaboration strategy. As we have said, the four agents will use the bias based decision policies, so they are going to focus on minimizing their individual case base

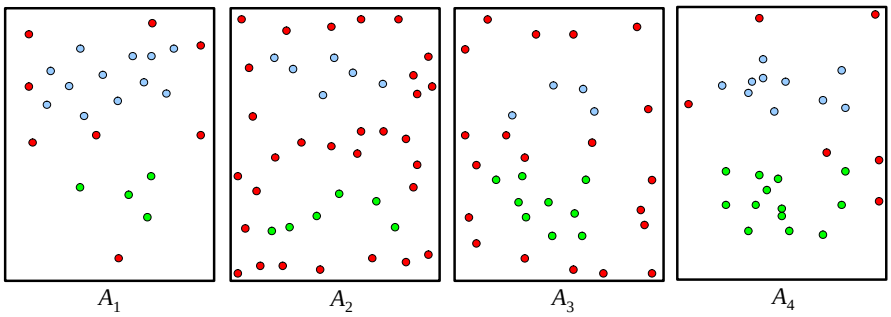


Figure 9.6: Initial distribution of cases among the four agents.

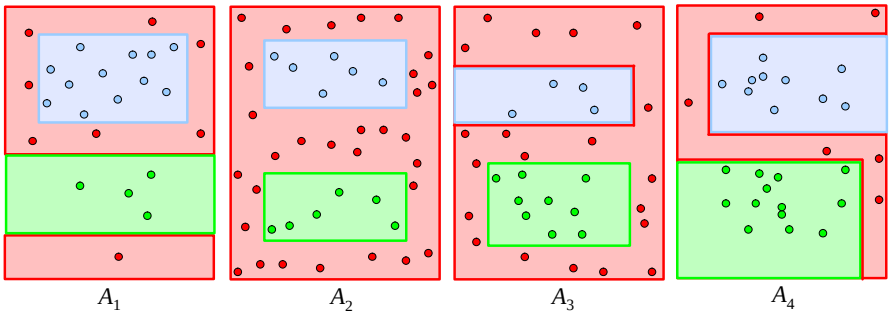


Figure 9.7: Partition of the problem space obtained using a decision tree algorithm to index the cases of the individual case bases.

biases. Moreover, for this exemplification the agents will use the non-copy mode, so no copies of the case will be made. After CB-CS the case bases of the agents are (Figure 9.6): A_1 has 7, 5, and 12 cases of the blue, green, and red class respectively; A_2 has 10,11, and 21 cases; A_3 has 8, 8, and 17 cases; and finally A_4 has 7, 8 and 14 cases respectively. Notice that, since the agents have not exchanged copies of cases but the cases themselves, the number of cases that each agent has after case bartering is exactly the same than before case bartering. Moreover, if we compute the individual case base bias of the agents after case bartering, we obtain:

$$\mathbb{B}(A_1) = \sqrt{\left(\frac{7}{24} - \frac{16}{64}\right)^2 + \left(\frac{5}{24} - \frac{16}{64}\right)^2 + \left(\frac{12}{24} - \frac{32}{64}\right)^2} = 0.0589$$

$$\mathbb{B}(A_2) = \sqrt{\left(\frac{10}{42} - \frac{16}{64}\right)^2 + \left(\frac{11}{42} - \frac{16}{64}\right)^2 + \left(\frac{30}{42} - \frac{21}{64}\right)^2} = 0.0168$$

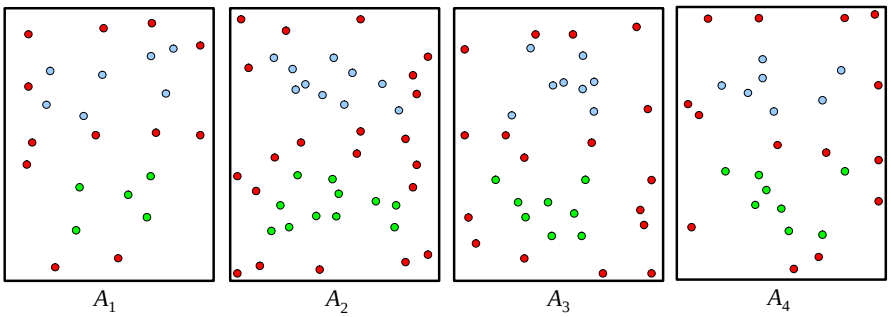


Figure 9.8: Distribution of cases among the four agents obtained after using CB-CS.

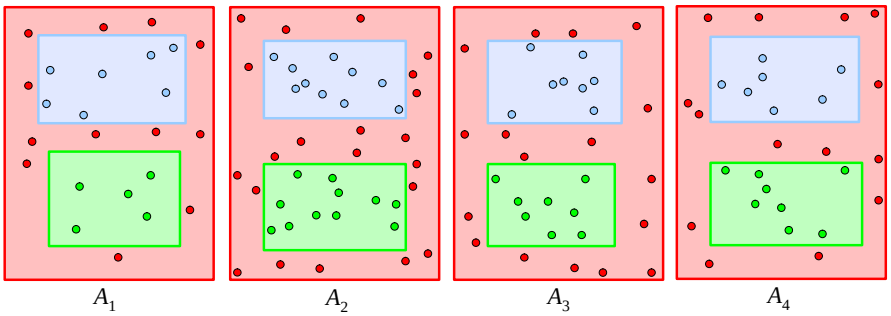
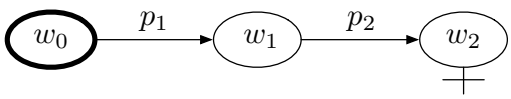


Figure 9.9: Partition of the problem space obtained using a decision tree algorithm to index the cases of the individual case bases obtained after using CB-CS.

$$\mathbb{B}(A_3) = \sqrt{\left(\frac{8}{33} - \frac{16}{64}\right)^2 + \left(\frac{8}{33} - \frac{16}{64}\right)^2 + \left(\frac{20}{33} - \frac{17}{64}\right)^2} = 0.0186$$

$$\mathbb{B}(A_4) = \sqrt{\left(\frac{7}{29} - \frac{16}{64}\right)^2 + \left(\frac{8}{29} - \frac{16}{64}\right)^2 + \left(\frac{6}{29} - \frac{14}{64}\right)^2} = 0.0323$$

The average Committee Bias is $\mathbb{B} = 0.0316$, i.e. after case bartering the average bias has been reduced by an order of magnitude. The effect can be seen in Figures 9.8 and 9.9; Figure 9.8 shows the sample of cases obtained by the agents after case bartering, and Figure 9.9 shows how a decision tree algorithm used to index the cases in the case base would partition the problem space with the case bases obtained after case bartering. Figure 9.9 shows that the partition of the problem space obtained after case bartering resembles much



p_1 : $Request(?A_i, ?A_j, ?E)$

p_2 : $Inform(!A_j, !A_i, ?\mathbf{J}_E)$

Figure 9.10: Interaction protocol used by the agents in the D_{JG} decision policy.

more the original partition of the data set than the partition obtained before case bartering.

Thus, the idea of case bartering is that each agent can obtain a case base that is more representative of the application domain.

Next section presents an alternative set of decision policies based on justifications.

9.6 Justification Based Decision Policies

This section presents a set of decision policies for case bartering based on justifications. In Chapter 8 we have shown that justifications can be used to evaluate the utility of a case or set of cases for a CBR agent. Specifically, Chapter 8 presented the JCU function that used justifications to estimate the utility that a set of cases has for a given CBR agent. The goal of the decision policies presented in this section is to provide two agents with a way to select the cases that when bartered are useful (in the sense that they have a JCU utility higher than zero) for both agents.

The Case Bartering Collaboration Strategy (CB-CS) requires three decision policies (D_G, D_B and D_A). In this section we will present three decision policies (D_{JG}, D_{JB} and D_{JA}) based on justifications, that can be used by the agents in a $\mathcal{MAC}$ system in the CB-CS collaboration strategy.

9.6.1 Justification Based Information Gathering

During the Information Gathering step of the CB-CS collaboration strategy, agents need to obtain information in order to be able to decide which case to barter in the Case Bartering step. Since the goal of the agents is to select cases with a JCU utility higher than zero, each agent $A_i \in \mathcal{A}$ should, in principle, obtain information about which cases of the other agents they are interested in. However, since the agents do not know which cases own the rest of the agents this is not feasible. Instead of that, a CBR agent A_i can easily know which of its own cases can be interesting (i.e. have a JCU utility higher than zero) for the other agents in the system. Thus, this is the information that the agents will collect in the Information Gathering step.

The main idea is that each agent $A_i \in \mathcal{A}$ will send an *exam* to the rest of the agents in the committee and request them to provide justified predictions for each of the problems in the exam. With the justifications of those predictions, A_i can evaluate which cases of its local case base C_i are interesting for each of the other agents in $\mathcal{A}$. Let us introduce some concepts before presenting the justification based Information Gathering decision policy.

Specifically, an *exam* built by an agent A_i is a set of problems for which A_i knows the correct solution. To build an exam E, an agent A_i chooses a subset of cases $C_E \subseteq C_i$ (that we will call the *examination cases*) from its case base. Then, the exam can be generated as $E = \{c.P | c \in C_E\}$ (recall that the case description $c.P$ of a case c is in fact a problem).

When an agent A_j receives an exam E, A_j solves each problem $P \in E$ providing a justified prediction in the form of a JER (Justification Endorsing Record) for each problem. Thus, the answer of an agent A_j for an exam E is a set of JERs $\mathbf{J}_E$ (containing a JER for each problem in the exam). When A_i receives the set $\mathbf{J}_E$ from an agent A_j, A_i can proceed as in JUST (see Section 8.3.2): determine which is the set of incorrect JERs $\mathbf{J}_E^-$, and for each incorrect JER $\mathbf{J} \in \mathbf{J}_E^-$, build the *refutation set* (see Definition 8.3.2). Let us note by $\mathcal{R}_{A_j}^t$ the collection of all the refutation sets computed for an agent A_j in a bartering round t. Notice that the cases in the refutation sets in $\mathcal{R}_{A_j}^t$ are the cases that have a JCU utility higher than zero for the agent A_j, and thus A_i is able to know which cases are interesting for A_j.

Moreover, since A_i is interested in obtaining information about all the other agents in the system, during information gathering A_i will obtain a collection of refutation sets $\mathcal{R}_{A_j}^t$ for each other agent $A_j \in \mathcal{A} - \{A_i\}$. We will note by $\mathbb{R}_{A_i}^t = \{\mathcal{R}_{A_j}^t\}_{A_j \in \mathcal{A} - \{A_i\}}$ to the set of all the collections of refutation sets obtained by an agent A_i at a round t.

We can now formally define the *Justification based Information Gathering* decision policy as:

Definition 9.6.1. *The* Justification based Information Gathering *decision policy D_{JG} used by an agent $A_i \in \mathcal{A}$ in CB-CS consists of generating an exam E from the set of examination cases $C_E \subseteq C_i$ (i.e. using a subset of the cases in its case base as the examination cases); and then engaging the interaction protocol shown in Figure 9.10 with each agent $A_j \in \mathcal{A} - \{A_i\}$. The sub-protocol works as follows:*

1. *A_i sends the exam E to A_j.*
2. *A_j solves each problem $P \in E$ and provides a justified prediction for each problem. Then, A_j sends all the justified predictions back to A_i.*
3. *With the justified predictions received from A_j, A_i builds the collection of refutation sets $\mathcal{R}_{A_j}^t$ (to be used later to generate bartering offers in the Case Bartering step).*

After having used the sub-protocol with the other agents, A_i has obtained the set $\mathbb{R}_{A_i}^t$, containing a collection of refutation sets for each agent $A_j \in \mathcal{A} - \{A_i\}$.

Moreover, in our experiments all the cases in the case base are used as examination cases (i.e. $C_E = C_i$) in order to obtain the maximum possible information during the Information Gathering step.

With the information gathered using the D_{JG} decision policy, the agents are now ready to generate bartering offers as next section explains.

9.6.2 Justification Based Case Bartering

During the Case Bartering step of the CB-CS collaboration strategy, agents are allowed to send bartering offers to other agents. Moreover, bartering offers used by the justification based case bartering differ in form from those used in the bias based case bartering. The main difference is that in the bias based bartering the bartering offers make reference only to the solution class of the cases, while in the justification based bartering the bartering offers make reference to specific cases. For this reason, in this section a new bartering offer is defined to be used in the justification based bartering, and also a new decision policy will be defined to generate such bartering offers.

During the Information Gathering step the information that each agent A_i has obtained is which cases of C_i are interesting for the rest of agents, but not which cases of the rest of agents are interesting for him. Therefore, the bartering offer will be made in two steps:

1. In a first step, an agent A_i sends a *bartering request* to another agent A_j, informing A_j that A_i has a set of cases in its case base that may be interesting for A_j.

2. In a second step, (and assuming that A_j has accepted the bartering request made by A_i) A_j sends a *bartering offer* to A_i, where A_j proposes to A_i two specific sets of cases to barter.

Notice that, since a bartering offer makes reference to specific cases, A_j cannot construct a bartering offer if A_i does not send a bartering request containing which cases A_i has that can be of interest for A_j.

Let us formally define bartering requests and bartering offers.

Definition 9.6.2. *A* bartering request $\mathbf{Q} = \langle D_{R\rightarrow O}, A_R, A_O \rangle$ *sent by an agent A_R (requester) to an agent A_O (offeror) is a tuple containing a set of case descriptions ($D_{R\rightarrow O}$), that A_R thinks A_O might be interested in acquiring.*

Notice that the bartering requests contains case descriptions only and not the complete cases (recall that a case c consists of a case description $c.P$ and a solution $c.S$). Thus, a bartering request incudes the case description of some cases in the case base C_i of an agent A_i that might be interesting for an offeror agent A_j.

When an agent A_j receives a bartering request from a requester agent A_i that upon consideration seems useful (using its D_A acceptance decision policy), A_j will answer to the requester with a message containing a bartering offer.

Definition 9.6.3. *A* Bartering Offer $\mathbf{O} = \langle D_{O\rightarrow R}, D_{R\rightarrow O}, A_O, A_R\rangle$ *is a tuple sent by an* offeror *agent* A_O *to an agent* A_R *(the* requester*) with a set* $D_{O\rightarrow R}$ *containing the descriptions of the cases that* A_O *offers to* A_R *in exchange of the cases whose descriptions are in* $D_{R\rightarrow O}$*. Moreover, the size of the sets* $D_{O\rightarrow R}$ *and* $D_{R\rightarrow O}$ *must be the same*

The restriction that the size of the sets $D_{O\rightarrow R}$ and $D_{R\rightarrow O}$ must be the same is imposed so that bartering is a smooth process. Let us illustrate the problems that can arise if we do not impose this restriction with an extreme example: in a bartering process involving sets of cases of different sizes, an agent A_i with a small case base could receive a large amount of cases in just one barter from another agent A_j; that would increase the error correlation of A_i dramatically with A_j since the majority of cases owned by A_i would be the cases received from A_j. Therefore, unequal size in bartering offers could make the committee to perform sub-optimally since a committee is interested in keeping a low error correlation in order to take benefit from the ensemble effect.

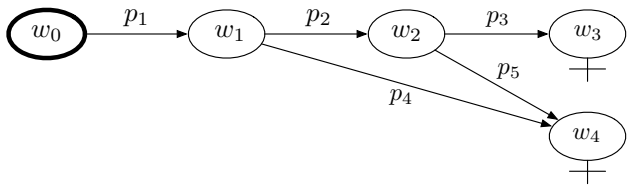
If a bartering offer is accepted, the offeror agent A_O will send the cases $B_{O\rightarrow R} = \{c \in C_O | c.P \in D_{O\rightarrow R}\}$ to A_R and the requester agent A_R will send the cases $B_{R\rightarrow O} = \{c \in C_R | c.P \in D_{R\rightarrow O}\}$ to A_O. Moreover, for the justification based decision policies we have only considered the *copy mode*, i.e. the agents barter copies of the cases. To consider the *non-copy mode* an agent must have a way to determine that the cases that it is giving to another agent are less useful than the cases received. However, to keep our experiments as simple as possible, we have not considered such scenario. In the conclusions section we will retake this discussion of explain some possibilities on how decision policies to work in the *non-copy mode* could be designed.

The set of decision policies presented in this section define the justification based case bartering strategy (*CB-CS justifications*).

9.6.2.1 Justification-based Case Bartering Policy

The D_{JB} decision policy used by an agent A_i decides, for every $A_j \in \mathcal{A} - A_i$, whether it is interesting to send a bartering request to A_j or not. If it is interesting, a sub-protocol is engaged to allow the agents to communicate the bartering requests and offers. The sub-protocol is shown in Figure 9.11, and requires four additional decision policies:

- D_{JRG}: that generates the bartering requests.
- D_{JOG}: that generates the bartering offers.
- D_{JRA}: that decides whether to accept or not a bartering request received from another agent.
- D_{JOA}: that decides whether to accept or not a bartering offer received from another agent.



p_1 :	$Request(?A_i, ?A_j, ?\mathbf{R})$
p_2 :	$Request(!A_j, ?A_i, ?\mathbf{O})$
p_3 :	$Request(!A_i, ?A_j, accept)$
p_4 :	$Request(!A_j, ?A_i, reject)$
p_5 :	$Request(!A_i, ?A_j, reject)$

Figure 9.11: Interaction protocol used by the agents in the D_{JB} decision policy.

Figure 9.12 shows the relation between all the protocols and decision policies involved in the justification based case bartering. Figure 9.12 shows that the main case bartering protocol requires two decision policies D_{JG} and D_{JB}.

In the remainder of this section we are going to define the D_{JRG}, D_{JOG}, D_{JRA} and D_{JOA} decision policies and also define how the sub-protocol engaged by D_{JB} uses them. Let us first formally define D_{JB}.

Definition 9.6.4. *The* Justification based Case Bartering *decision policy D_{JB} used by an agent A_i in a round t of CB-CS is a boolean function that is applied for each agent $A_j \in \mathcal{A} - \{A_i\}$ to decide whether to engage the sub-protocol with A_j or not:*

$$D_{JB}(A_j) = \begin{cases} true & if\ D_{JRG}(A_j) \neq \emptyset \\ false & otherwise \end{cases}$$

For each agent A_j the decision policy D_{JRG} is used to generate a bartering request for A_j, and if the bartering request generated is not empty ($D_{JRG}(A_j) \neq \emptyset$), then the sub-protocol is engaged with A_j.

The protocol of Figure 9.11 used by the D_{JB} decision policy is engaged when D_{JB} decides that it is interesting to send a bartering request $\mathbf{Q}$ to an agent A_j, i.e. when A_i has some cases that might be interesting for A_j. The protocol works as follows:

1. A_i sends a message p_1 to A_j with a bartering request $\mathbf{Q}$.
2. If A_j accepts the request (using its D_{JRA} acceptance decision policy), generates a bartering offer $\mathbf{O}$ as explained above and sends a message p_2 to A_i containing the bartering offer $\mathbf{O}$. If A_j rejects the request, a reject message p_4 is sent to A_i and the protocol ends.

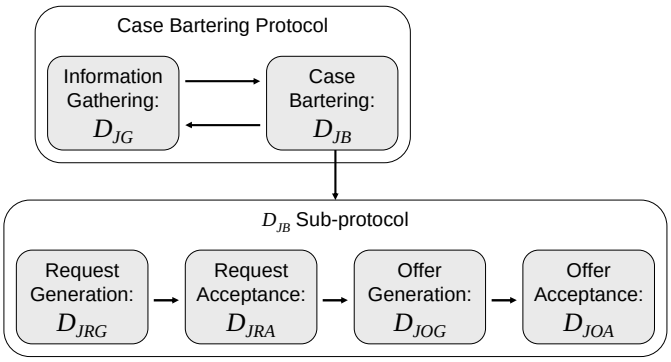


Figure 9.12: Relation between the protocols and decision policies used in justification based case bartering.

3. If A_i accepts the offer (using its D_{JOA} acceptance decision policy), an accept message p_3 is sent to A_j, the protocol ends and the cases are exchanged; otherwise, a reject message p_5 is sent to A_j and the protocol ends.

The following sections present each one of the decision policies used in the sub-protocol.

9.6.2.2 Justification-based Request Generation Policy

Let us focus on how an agent A_i uses the information obtained in the Information Gathering step to generate a bartering request to send to an agent A_j in a round t of CB-CS. First of all A_i has to determine which cases is it going to offer to A_j. To do so, A_i uses the collection of refutation sets $\mathcal{R}^t_{A_j} \in \mathbb{R}^t_{A_i}$ collected for A_j. Each refutation set in $\mathcal{R}^t_{A_j}$ contains the cases that can prevent A_j of making again one of the errors made in the exam that A_i sent to A_j in the Information Gathering step. Thus, by offering A_j a case from each refutation set, A_i is actually offering a set of cases that can prevent A_j of making each one of the errors made in the exam. Therefore, to build a bartering request, A_i selects a case from each of the refutation sets in $\mathcal{R}^t_{A_j}$, constructing the set of cases $B^t_{i\to j}$, and then the set of case descriptions $D^t_{i\to j} = \{c.P\}_{c\in B^t_{i\to j}}$ with which the bartering request $\langle D^t_{i\to j}, A_i, A_j\rangle$ is made.

Moreover, since an agent A_i may barter cases with more than one agent at each round of CB-CS, an additional issue must be taken care of. Imagine that A_i has already bartered cases with an agent A_k at a round t, an now A_i is going to send a bartering request to another agent A_j. Let us note by $B^t_{i\to k}$ the set of cases that A_j has already bartered with A_k and by $B^t_{i\to j}$ the set of cases that A_i is going to offer to A_j in the bartering request. Notice that if there is a non null intersection between $B^t_{i\to k}$ and $B^t_{i\to j}$ some cases will be sent to more than one agent. This may increase the degree to which agent A_j and A_k make

correlated predictions. Moreover, if A_i selects cases from the refutation sets in $\mathcal{R}^t_{A_j}$ that are not present in $B^t_{i\to k}$, the correlation between the predictions of A_j and A_k is not increased. Thus, since the goal of the agents using CB-CS is to increase their individual performance and also the committee performance, it is interesting to keep correlation as low as possible (since it is one of the preconditions for the ensemble effect to take place).

Thus, we can now formally define the D_{JRG} decision policy:

Definition 9.6.5. *The* Justification based Request Generation *decision policy D_{JRG} used by an agent A_i in a round t of CB-CS to generate a bartering request for an agent A_j consists of the the following steps:*

1. *Let us note by $\mathcal{A}^b \subset \mathcal{A}$ the set of agents to which A_i has already sent bartering requests. Thus, A_i starts by identifying the set of cases B^b that A_i has already bartered with the agents in $\mathcal{A}^b$.*
2. *After that, for each non empty refutation set $R^{\mathbf{J}}_t \in \mathcal{R}^t_{A_j}$:*
   (a) *A_i looks if there is any case $c \in R^{\mathbf{J}}_t$ that is not present in B^b:*
      - *If such a case exists, c is selected (if more than one case matches this condition, A_i randomly selects one of them).*
      - *Otherwise, A_i randomly selects a case from all the cases in $R^{\mathbf{J}}_t$.*
3. *All the cases selected in the previous step form the set of cases $B^t_{i\to j}$. Thus, A_i constructs the set $D^t_{i\to j} = \{c.P\}_{c\in B^t_{i\to j}}$ containing all the case descriptions of the cases in $B^t_{i\to j}$.*
4. *Finally, $D^t_{i\to j}$ is the output of the decision policy.*

Notice that in step 2.(a), cases not previously bartered (i.e. cases not present in B^b) are preferred to cases previously bartered. In this way, correlation is kept as low as possible.

9.6.2.3 Justification-based Offer Generation Policy

When an agent A_j receives a bartering request $\mathbf{Q}$ from an agent A_i, A_j uses the D_{JOG} decision policy to decide which cases to offer to A_i in exchange of the cases that A_i has offered.

Definition 9.6.6. *The* Justification based Bartering Offer Generation *decision policy D_{JOG} used by an agent A_j in a round t of CB-CS after having received a bartering request $\mathbf{Q}$ from an agent A_i consists of the the following steps:*

1. *Let us note by $\mathcal{A}^b \subset \mathcal{A}$ the set of agents to which A_j has already sent bartering requests. Thus, A_j starts by identifying the set of cases B^b that A_j has already bartered with the agents in $\mathcal{A}^b$.*
2. *After that, for each non empty refutation set $R^{\mathbf{J}}_t \in \mathcal{R}^t_{A_i}$:*

(a) A_j *looks if there is any case* $c \in R_t^{\mathbf{J}}$ *that is not present in* B^b*:*
- *If such a case exists,* c *is selected (if more than one case matches this condition,* A_j *randomly selects one of them).*
- *Otherwise,* A_j *randomly selects a case from all the cases in* $R_t^{\mathbf{J}}$*.*

3. *All the cases selected in the previous step form the set of cases* $B_{j\to i}^t$*. Thus,* A_j *constructs the set* $D_{j\to i}^t = \{c.P\}_{c\in B_{j\to i}^t}$ *containing all the case descriptions of the cases in* $B_{j\to i}^t$*.*

4. *Finally, since we imposed a restriction over bartering offers so that the size of* $D_{j\to i}^t$ *must be the same than* $D_{i\to j}^t$*,* A_j *performs an extra step:* A_j *selects two subsets* $\widehat{D}_{j\to i}^t \subseteq D_j$ *and* $\widehat{D}_i \subseteq D_i$ *such that* $\widehat{D}_{j\to i}^t$ *and* $\widehat{D}_{i\to j}^t$ *are of the same size. Then, the bartering offer* $\mathbf{O} = \langle \widehat{D}_{j\to i}^t, \widehat{D}_{i\to j}^t, A_j, A_i \rangle$ *is generated and sent to* A_i*.*

9.6.2.4 Justification Based Request Acceptance

In this section we will define two very simple decision policies to be used by the agents to accept or reject bartering requests and bartering offers coming from other agents in the $\mathcal{M}$AC system.

Definition 9.6.7. *The* Justification-based Bartering Request Acceptance *decision policy* D_{JRA} *is defined as accepting every bartering request that offers a non-empty subset of cases, i.e.*

$$D_{JRA}(\mathbf{Q}) = \begin{cases} true & if\ \mathbf{Q}.D_{R\to O} \neq \emptyset \\ false & otherwise \end{cases}$$

Notice that the previous definition assumes that each agent participating in the CB-CS collaboration strategy is collaborative, and that the agents will not offer cases in the bartering requests that are not interesting for the agents that receive the requests. If this assumption does not hold, a more elaborated decision policy should be defined.

9.6.2.5 Justification Based Offer Acceptance

Definition 9.6.8. *The* Justification-based Bartering Offer Acceptance *decision policy* D_{JOA} *is defined as accepting every bartering offer that offers a non-empty subset of cases, i.e.*

$$D_{JOA}(\mathbf{O}) = \begin{cases} true & if\ \mathbf{O}.D_{O\to R} \neq \emptyset \\ false & otherwise \end{cases}$$

Again, this acceptance decision policy assumes that every agent is collaborative.

The next section presents an empirical evaluation of case bartering that includes experiments with both bias based and justification based decision policies.

9.7 Experimental Evaluation

In this section we are going to empirically evaluate the CB-CS collaboration strategy. Specifically, we are going to compare the classification accuracy obtained by $\mathcal{MAC}$ systems composed of 3, 5, 7, 9, 11, 13, and 15 agents with and without case bartering. Moreover, we are also going to discuss results concerning the properties of case bases achieved using each one of the different collaboration strategies (including case base size, completeness, redundancy, and bias).

To test the case bartering strategy, the cases in the training set are not randomly sent to the agents but are sent with a certain degree of bias, i.e. some agents have a higher probability of receiving cases of certain classes and a smaller probability of receiving cases of other classes. In an experimental run, the training set is distributed among the agents (in a biased way) and the classification accuracy is measured; then the agents use the case bartering strategy and the classification accuracy is measured again in order to evaluate the difference. Specifically, we are going to compare four different strategies:

- *base*: represents a $\mathcal{MAC}$ system where the agents do not apply any case bartering strategy.
- *CB-CS bias copy*: the agents use the bias based strategy (see Section 9.3) in the copy mode (i.e. exchanging copies of cases).
- *CB-CS bias no-copy*: the agents use the CB-CS collaboration strategy using the bias based strategy in the non-copy mode (i.e. exchanging cases and not copies of cases).
- *CB-CS justifications*: the agents use the justification based strategy (see Section 9.6) to exchange copies of cases.

Moreover, we have limited the number of bartering rounds of the bias based strategies to 100 and of the justification based strategy to 10. We have allowed more rounds in the bias based strategies because them barter cases one by one while the justification based strategy barters batches of cases and requires less iterations.

In order to test the generality of the collaboration strategies we are going to present results using three data sets: sponge, soybean and zoology. Moreover, in all the experiments reported in this section agents use LID as the learning method and all the presented results are the average of five 10-fold cross validation runs.

We will first present experimental results concerning classification accuracy, and after that we will analyze the properties of the case bases with the different decision policies.

9.7.1 Accuracy Evaluation

Figure 9.13 shows the classification accuracy achieved by agents in $\mathcal{MAC}$ systems composed of 3, 5, 7, 9, 11, 13, and 15 agents in the sponge data set. Four bars

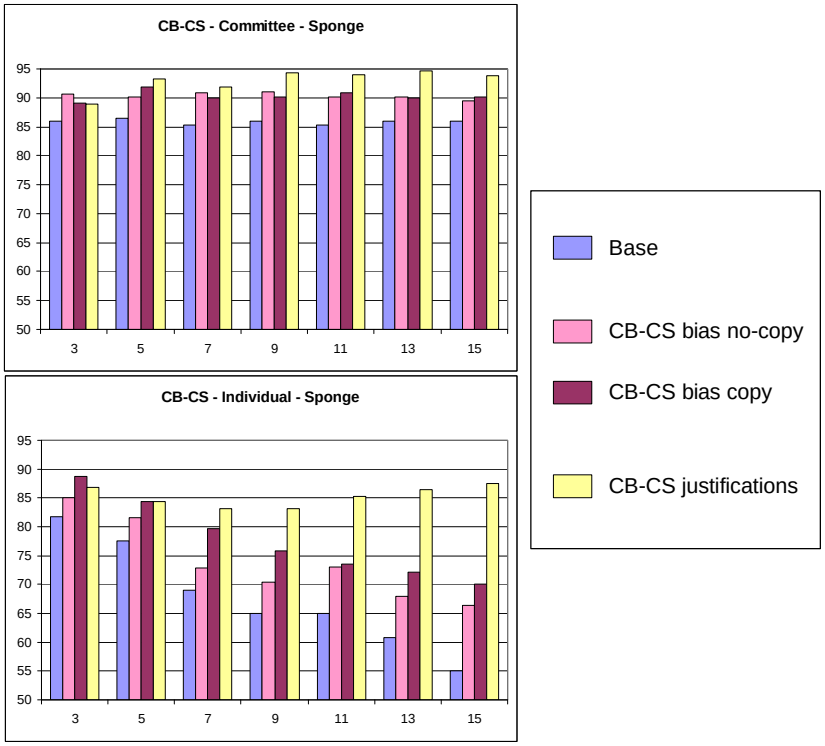


Figure 9.13: Classification accuracy comparison between agents using the different CB-CS decision policies in the sponge data set for several $\mathcal{M}$AC systems.

are shown for each $\mathcal{M}$AC system, corresponding to the *base*, *CB-CS bias no-copy*, *CB-CS bias copy* and *CB-CS justifications* strategies explained above. The left hand side plot shows the committee classification accuracy and the right hand side plot shows the individual classification accuracy. Figure 9.13 shows that the classification accuracy achieved using case bartering is always higher to that achieved without case bartering, both concerning the individual and the committee accuracy.

Concerning the bias based strategies, *CB-CS bias copy* and *CB-CS bias non-copy*, Figure 9.13 shows that both strategies are almost undistinguishable in the committee classification accuracy. However, *CB-CS bias copy* achieves higher individual classification accuracy than *CB-CS bias non-copy*. The explanation is that in the copy mode the agents barter copies of cases and therefore the average number of cases that an individual agent has after case bartering is higher than before case bartering. Thus, the individual classification accuracy increases due to the individual increase of case base size. However, the committee classification accuracy is not increased with respect to *CB-CS bias non-copy* because bartering copies of cases increases the error correlation among the agents.

Concerning *CB-CS justifications*, Figure 9.13 shows that agents using *CB-CS justifications* achieve the highest classification accuracy values for all the $\mathcal{M}$AC systems except for the 3 agents system, where *CB-CS non-bias copy* achieves higher committee accuracy and *CB-CS bias copy* achieves higher individual accuracy. Moreover, notice that using *CB-CS justifications* the classification accuracy of systems composed of many agents is equal (or even higher) than that of systems with less agents. The difference in classification accuracy between the *base* strategy and *CB-CS justifications* is specially high for systems with many agents: for instance, in the 15 agents $\mathcal{M}$AC system, the classification accuracy for the *base* strategy is 55.07% for the individual agents and 86.00% for the committee while the classification accuracy for *CB-CS justifications* is 87.57% for the individual agents and 93.86% for the committee. Moreover, the difference in classification accuracy between the bias based strategies and *CB-CS justifications* is also higher in systems with many agents. *CB-CS justifications* achieves higher accuracy than the bias based strategies because their goal is to minimize the case base bias, and the increase of classification accuracy is a consequence of this; thus they achieve to increase classification accuracy in an indirect way. *CB-CS justifications* directly tries to increase classification accuracy by exchanging cases with high JCU utility, that are cases that can fix errors.

Figure 9.14 shows the classification accuracy achieved in the soybean data set. Figure 9.14 shows that the bias based strategies achieve higher classification accuracy values than without using case bartering. Moreover, in the soybean data set, the classification accuracy achieved using the copy mode is higher than the one achieved using the non-copy mode. For instance, the accuracy achieved by a committee of 11 agents using bias based case bartering in the non-copy mode is 65.89% while working in the copy mode it is 74,01%. The explanation is that in the soybean data set the classification accuracy diminishes quickly if we reduce the number of cases that an agent has in its case base (as can be seen in the $\mathcal{M}$AC systems with many agents); therefore, working in the copy mode increases classification accuracy because it increases the average size of individual case bases.

Figure 9.14 shows that *CB-CS justifications* achieve even higher classification accuracy values than *CB-CS bias copy*. Moreover, notice that using *CB-CS justifications* the classification accuracy of systems composed of many agents is equal to that of systems with less agents (as we already observed in the sponge data set). Again, the explanation is that *CB-CS justifications* aims directly at increasing classification accuracy while the bias based strategies aim at reducing case base bias and the classification accuracy increase is a consequence.

Figure 9.15 shows the classification accuracy achieved in the zoology data set with results similar to those observed in Figures 9.13 and 9.14: case bartering systematically improves classification accuracy, *CB-CS bias copy* achieves higher classification accuracy than *CB-CS bias non-copy* and *CB-CS justifications* achieves the highest classification accuracy. Moreover, notice that in the zoology data set the highest classification accuracy is achieved in the 15 agents system, achieving a classification accuracy of 97.03% for the committee and a

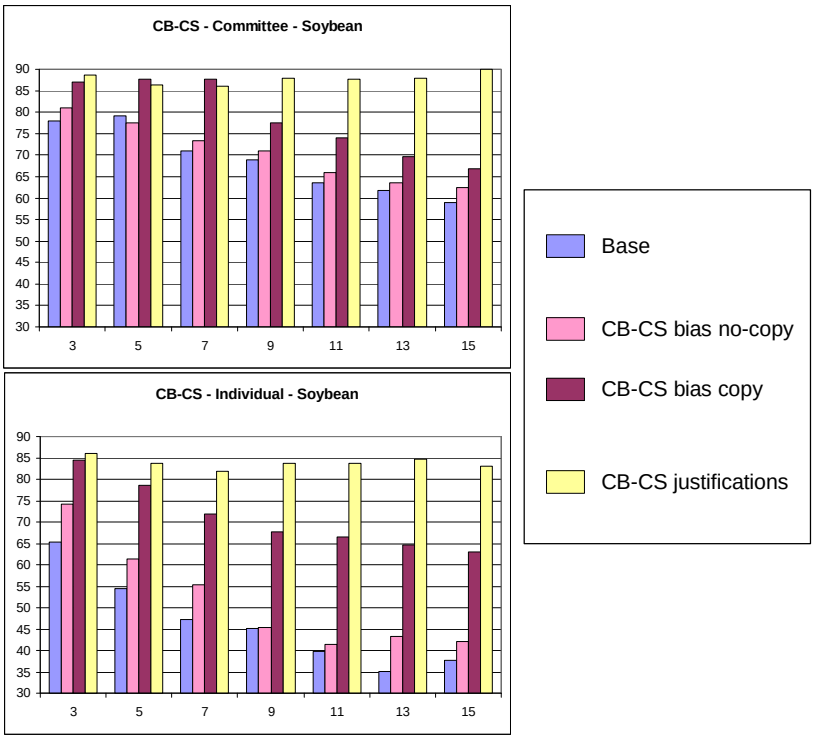


Figure 9.14: Classification accuracy comparison between agents using the different CB-CS decision policies in the soybean data set for several $\mathcal{M}$AC systems.

93.04% for the individual agents (a centralized CBR system owning all the cases achieves a classification accuracy of 95.64%). This result also happened in the sponge and soybean data sets, in the sponge data set the maximum accuracy is achieved by the 13 agents system (94.64%) and is also much higher than the accuracy achieved by a centralized CBR system owning all the cases (89.50%); in the soybean data set he maximum accuracy is achieved by the 15 agents system (89.90%) and is slightly higher than the accuracy achieved by a centralized CBR system owning all the cases (89.51%). The high classification accuracy achieved with the systems composed of many agents is because the justification based case bartering is able to boost the ensemble effect. Therefore, we can conclude that *CB-CS justifications* is able both to increase the classification accuracy of the individual agents and of keeping the error correlation among them low (since these are the preconditions of the ensemble effect).

The next section presents an analysis of the properties of the case bases of the individual agents after case bartering that will help us to better understand the results related to classification accuracy.

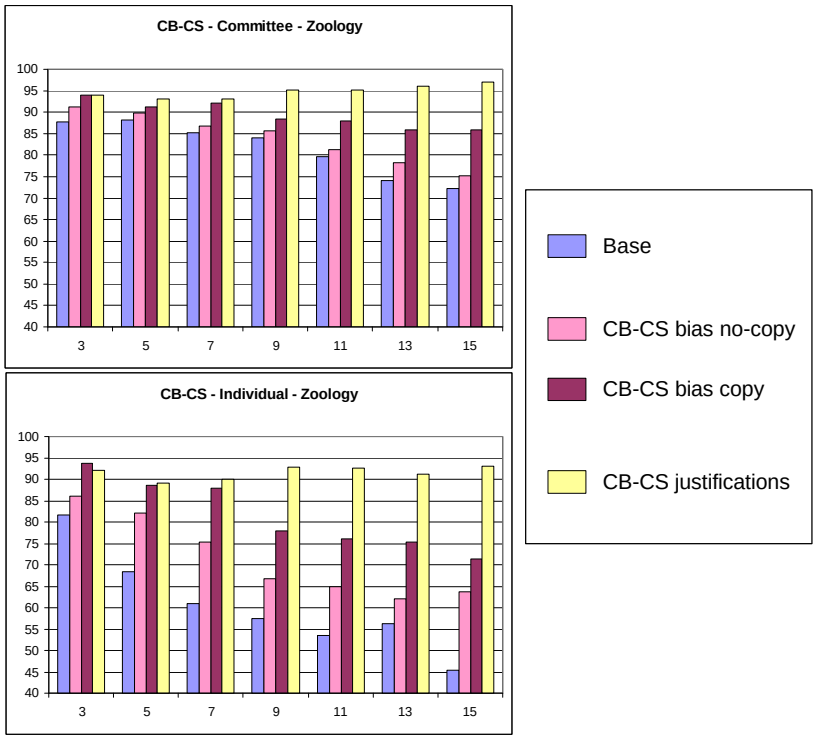


Figure 9.15: Classification accuracy comparison between agents using the different CB-CS decision policies in the zoology data set for several $\mathcal{MAC}$ systems.

9.7.2 Case Base Evaluation

Table 9.1 shows the case base properties of the CBR agents in a $\mathcal{MAC}$ system composed of 9 agents for the sponge data set. Each row of the table shows the results for a different strategy. The columns correspond to the following properties of the case base: the accuracy achieved by individual agents, the accuracy achieved by the committee of agents (using CCS), the average size of individual case bases, and the average Committee Completeness ($\mathbb{C}$), Committee Redundancy ($\mathbb{R}$) and Committee Bias ($\mathbb{B}$).

Concerning *CB-CS bias non-copy*, Table 9.1 shows that the only property of the case base that changes is the bias, that goes down from 0.45 to 0.20. Thus, the increment of the committee accuracy from 86.00% to 90.21% and of the individual accuracy from 65.07% to 70.42% is due entirely to this reduction of case base bias.

CB-CS bias copy has two effects in the case bases of the agents: decrease of bias (from 0.40 to 0.19) and slight increase of the redundancy (from 0.00 to 0.06). This increment in redundancy increases the average size of the case bases from 28.00 to 41.13 cases. If we recall the conclusions of Chapter 4, we concluded

	Individual	CCS	CB size	$\mathbb{C}$	$\mathbb{R}$	$\mathbb{B}$
Base	65.07%	86.00%	28.00	1.00	0.00	0.45
CB-CS bias no-copy	70.42%	90.21%	28.00	1.00	0.00	0.20
CB-CS bias copy	75.86%	91.00%	41.13	1.00	0.06	0.19
CB-CS justifications	83.21%	94.28%	95.02	1.00	0.30	0.13

Table 9.1: Case Base properties comparison between agents using the different CB-CS decision policies in the sponge data set in a 9 agents $\mathcal{M}$AC system.

	Individual	CCS	CB size	$\mathbb{C}$	$\mathbb{R}$	$\mathbb{B}$
Base	45.08%	68.99%	30.70	1.00	0.00	0.26
CB-CS bias no-copy	45.39%	71.07%	30.70	1.00	0.00	0.10
CB-CS bias copy	67.67%	77.60%	57.54	1.00	0.11	0.10
CB-CS justifications	83.71%	87.95%	119.35	1.00	0.36	0.10

Table 9.2: Case Base properties comparison between agents using the different CB-CS decision policies in the soybean data set in a 9 agents $\mathcal{M}$AC system.

that a low bias and a moderate redundancy were the optimal conditions for an ensemble. Thus, *CB-CS bias copy* is expected to outperform *CB-CS bias non-copy*. In fact, this expectation is accomplished, since the individual agents achieve an accuracy of 70.42% and 75.86% for the *CB-CS bias non-copy* and *CB-CS bias copy* and the committee achieves an accuracy of 90.21% and 91.00%.

Finally, the fourth row of Table 9.1 shows that *CB-CS justifications* has two effects in the case base: a decrease in bias, and a large increase in redundancy. The large increase in redundancy is reflected in a large increase in individual case base size, that has increased from an average of 28.00 cases per agent to 95.02. Moreover, this increase in case base size is useful, since it has boosted classification accuracy from 65.07% to 83.21% for the individual agents and from 86.00% to 94.28% for the committee. Moreover, the increase of classification accuracy is not only due to the increase in redundancy, since a committee of 9 agents with a redundancy of 0.30 achieves a classification accuracy of just 90.21% (as shown in Chapter 4). Therefore, the higher value of classification accuracy must be due to the ability of *CB-CS justifications* to properly select which cases are adequate to be bartered.

Table 9.2 shows the case base properties of the CBR agents for the soybean data set. The analysis is similar to that of the sponge data set: *CB-CS bias non-copy* increases the classification accuracy as a consequence of reducing bias; *CB-CS bias copy* introduces a small degree of redundancy that is able to increase classification accuracy even more; and finally the *CB-CS justifications* achieves the highest classification accuracy values for the same reasons explained in the sponge data set analysis.

Table 9.3 shows the case base properties of the CBR agents for the zoology data set showing the same results observed in the sponge and soybean data set.

CB-CS justifications is a more informed strategy than both *CB-CS bias*

	Individual	CCS	CB size	$\mathbb{C}$	$\mathbb{R}$	$\mathbb{B}$
Base	57.42%	83,96%	10.10	1.00	0.00	0.42
CB-CS bias no-copy	66.73%	85.54%	10.10	1.00	0.00	0.18
CB-CS bias copy	77.82%	88.51%	18.93	1.00	0.09	0.16
CB-CS justifications	92.80%	95.07%	36.28	1.00	0.32	0.25

Table 9.3: Case Base properties comparison between agents using the different CB-CS decision policies in the zoology data set in a 9 agents $\mathcal{MAC}$ system.

non-copy or *CB-CS bias copy*. This is because *CB-CS justifications* uses exams to analyze the contents of the case bases of the individual agents, and thus is able to decide which concrete cases would be interesting for certain agents. In contrast, the bias based strategies only analyze the contents of the case bases in terms of how many cases each agent has of each solution class and never make reference to concrete cases. Moreover, *CB-CS justifications* has a higher computational cost (since it has to process more information) than the bias based strategies. In fact, agents using *CB-CS justifications* have to solve $n-1$ exams at each bartering round (where n is the number of agents in the system) and the exams have an average of M/n cases (where M is the size of the training set); that represents $(n-1)*(M/n)$ retrieve operations per agent per cycle. Then, each agent has to analyze the results of the exams, that requires $(n-1)*(M/n)$ retrieve operations (in the worst case) per agent per cycle. Thus, the cost of the Information Gathering step in *CB-CS justifications* is $2*(n-1)*(M/n)$ retrieve operations per agent per cycle. In contrast, using the bias based strategies, agents only have to count the number of cases they have of each class. Thus, agents using *CB-CS justifications* have a higher computational cost than agents using the bias based strategies, but this cost is justified, since the additional information gathered by *CB-CS justifications* is useful in achieving better performance than the bias based strategies.

Summarizing the experimental results presented in this section we can conclude that case bartering improves classification accuracy both of individual agents and of committees of agents. Moreover, as we also observed in previous chapters, allowing some degree of redundancy increases the classification accuracy (as we have seen in this chapter comparing strategies working in the non-copy mode versus working in the copy mode). Finally, we have shown that the *CB-CS justifications* is able to achieve the highest classification accuracy values proving that it achieves an increased individual classification accuracy while keeping a low error correlation.

9.8 Conclusions

In this chapter we have presented several *Case Bartering Collaboration Strategies* (CB-CS), that allow a committee of agents to reach bartering agreements. Specifically, we have presented a common interaction protocol and two sets of

decision policies that the agents in a $\mathcal{M}$AC system can use with CB-CS:

- Strategies with bias based decision policies: focused on diminishing the bias of the individual agents in a $\mathcal{M}$AC system, and thus increasing classification accuracy. Moreover, these policies can work in the non-copy mode (that does not increase redundancy) or in the copy mode (that moderately increases redundancy).

- A strategy with justification based decision policies: that uses the JCU utility assessment function presented in Chapter 8. The purpose of this strategy is that each agent obtains cases with a high JCU utility (i.e. cases that can prevent them from making errors in the future), thus increasing their classification accuracy.

The experimental evaluation presented in this chapter allows us to conclude that case bartering is useful since both the individual and the committee classification accuracy increase with case bartering. The experiments have shown that the bias based strategies succeed in increasing classification accuracy by diminishing case base bias. Moreover, the experimental results have shown the moderate degree of redundancy introduced when performing case bartering in copy mode increases the classification accuracy with respect to the non-copy mode. Thus, the classification accuracy increase achieved by the bias based strategy in the copy mode is the effect of both diminishing case base bias and of slightly increasing redundancy.

The justification based strategy achieves the highest classification accuracy, and this fact cannot be explained only as a sum of the reduction of bias and in the increase of redundancy. In fact, justifications allow the agents to obtain information suitable to correctly decide which specific cases are effectively traded with whom. Moreover, the decisions taken are right in the sense that they achieve a higher individual agent classification accuracy while maintaining a low error correlation. The fact that the classification accuracy achieved using justifications is higher than the accuracy that could be expected by the degree of bias and of redundancy of the system let us conclude that the analysis made using justifications provides useful information that can be exploited by the agents (in this case to generate interesting bartering offers). Moreover, this reinforces the results obtained in Chapter 8 where a justification based retention strategy achieved higher accuracy than expected by the ensemble space analysis.

In general, we can view case bartering as a decentralized search process in the space of all the possible distributions of cases among the agents in a $\mathcal{M}$AC system (i.e. a search in the ensemble space in which the number of agents is fixed). Each time two agents barter cases, the distribution of cases in the $\mathcal{M}$AC system changes. Moreover, case bartering performs a search with the goal of maximizing/minimizing some criterion. For instance, the bias based strategies have as the goal minimizing the case base bias of the agents, and the justification based strategy has as the goal minimizing the number of individual errors that the agents make while solving exams during the information gathering step (i.e. maximizing their individual classification accuracy) and keeping the error

correlation as low as possible. However, case bartering is not limited to diminishing bias or to exchange high utility cases; new criteria could be defined other than the ones presented in this chapter, and new decision policies developed that optimize such criteria.

Notice that bias based case bartering is very related to classification tasks (that are our main focus in this work). The reason is that it is based on the result found in Chapter 4 stating that a high case base bias with respect to the partition induced by the solution classes leads to a reduced classification accuracy. Thus, it is not obvious to adapt the bias based bartering to other domains such as regression or planning, where there is no straightforward way to define a partition with which to compute case base bias. However, this limitation does not apply to the justification based case bartering. Justification case bartering relies only on the concept of "erroneous solution": for any domain in which we can define a criterion of what is an erroneous solution, it will be possible to compute the refutation sets, and thus to use the justification based decision policies. For instance, in a planning domain, an erroneous solution can be defined as a solution that involves an execution cost higher than a certain value, or that has a higher cost than a given known solution. Therefore, the justification based case bartering is a more general strategy than the bias based one in the sense that it can be applied to a wider range of domains.

Finally, there are two open issues in the case bartering strategies presented in this chapter. The first issue is to define bartering decision policies that automatically find the optimal degree of redundancy (so that the committee achieves the maximum classification accuracy). Current decision policies introduce a moderate degree of redundancy, that improves classification accuracy, but they do not search the optimal redundancy degree. The second open issue is to define case bartering decision policies inspired on justifications using the non-copy mode. The problem to work in the non-copy mode is that the agents need two criteria: one to decide which cases are interesting to be obtained and another one to decide to which cases to do without. As future work we think it would be interesting to integrate JUST(see Chapter 8) into the justifications based case bartering so that it determines a set of cases that an agent can do without in exchange of cases with higher utility.

Chapter 10

Conclusions

This chapter presents a brief summary of the thesis, presenting a complete list of the techniques and strategies introduced, discussing then the main contributions of our work, and finally outlining some future lines of research.

10.1 Summary

In this section we are going to present a brief summary of the work presented in this thesis with the purpose of having a global view of all the thesis.

With the purpose of studying the effect of distributed data and distributed decision making in learning problems, in this work we have focused on *Multi-Agent Case Based Reasoning Systems* ($\mathcal{M}$AC), that are multi-agent systems where each individual agent uses CBR to solve problems. Moreover, the work on $\mathcal{M}$AC systems presented in this thesis can be grouped in three main areas, as Figure 10.1 shows:

- Work on *committees*, focusing on how agents can coordinate to act as ensembles that achieve better performance than working individually.
- Work on *retention*, focusing on retention strategies specifically designed for a multi-agent setting instead of single CBR systems.
- Work on *redistribution of cases*, focusing on how the agents in a $\mathcal{M}$AC system can coordinately exchange cases in their case bases in order to improve their individual and committee performance.

Figure 10.1 shows these three areas of work, together with the specific techniques proposed in this thesis for each area.

Committees

Concerning the work presented in committees, our research has focused in three sub-areas, namely *committee formation*, *prediction aggregation* and *committee*

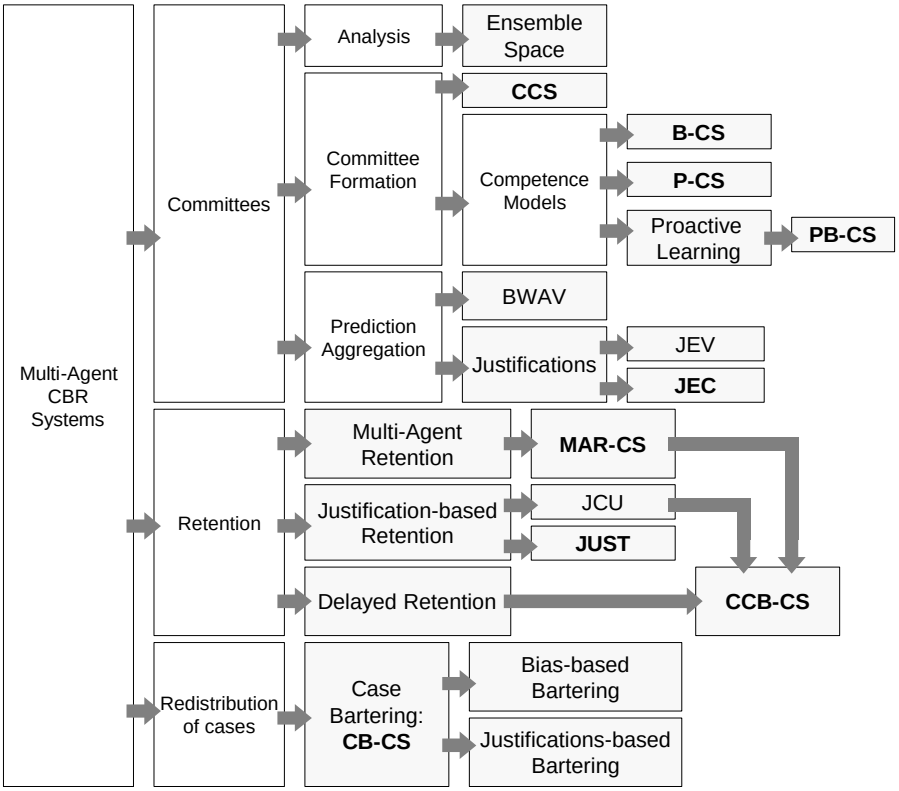


Figure 10.1: The different techniques presented in this thesis, grouped by areas.

analysis (see Figure 10.1). Specifically, committee formation involves designing collaboration strategies and decision policies that allow individual agents to decide when to convene committees and which agents to convene. The *Committee Collaboration Strategy* (CCS) has been presented as the basic way to convene committees and has been shown to have a higher classification accuracy than when the agents solve problems individually. Therefore, CCS has shown the feasibility of attaining the ensemble effect over distributed data.

CCS uses the static strategy of convening a committee with all the agents in a $\mathcal{M}$AC system for any problem, but it is not obvious that this is the best strategy in all situations. Therefore, we proposed the *dynamic committee* collaboration strategies, where an ad hoc committee is convened for each specific problem. For this purpose, we defined *competence models*, used to assess the confidence that an agent or group of agents will predict the correct solution for a given problem. Moreover, competence models can also be used for self-evaluation. Using competence models agents can determine when a committee should be convened or not (assessing its own confidence) and which agents should be convened (assessing the confidence of the other agents in the system). Notice that

there are problems for which dynamic committees will not convene a committee at all (when the competence model predicts that an individual agent is competent enough to individually solve the problem). Thus, dynamic committees are a lazy approach to convening committees, where committees are only convened when needed, and where no more agents than required are convened. Specifically, the *Bounded Counsel Collaboration* (B-CCS) and the *Peer Counsel Collaboration* (P-CCS) are strategies that use competence models to convene dynamic committees.

Moreover, we have also presented a proactive learning technique with which agents can autonomously learn competence models. Combining the proactive learning technique with B-CCS, we defined the *Proactive Bounded Counsel Collaboration Strategy* (PB-CCS), that achieved higher classification accuracy than B-CCS and PB-CCS, and even outperformed CCS in some scenarios. Moreover, PB-CCS showed to be a very robust way of creating dynamic committees, since it was much less sensitive to the distribution of cases among the agents than B-CCS, PB-CCS or CCS. This shows that competence models can be learnt, and that learning competence models gives $\mathcal{MAC}$ systems a larger flexibility (since it allows agents to perform well in a larger variety of scenarios) and a better performance than using predefined competence models.

Let us now focus on prediction aggregation. Prediction aggregation determines a global prediction from the set of individual predictions made by the agent members of a committee. We proposed the *Bounded Weighted Approval Voting* (BWAV) as a voting system specifically designed for CBR agents. BWAV uses the number of retrieved cases of each solution class to determine how many votes each agent casts for each solution class.

BWAV assumes that every agent is similarly competent. However, since there are situations where this assumption does not hold, it would be desirable to design techniques such that more competent agents have more relevance in the voting system than less competent agents. For this purpose we introduced the notion of *justifications*. A justification is the explanation given by an agent of why it has considered the solution of a specific problem to be correct. Justifications provide a way in which agents can examine the rationale of predictions made by other agents, and thus assess the confidence on the prediction made by any agent in a committee. Thus, while competence models offer an eager way to assess the confidence on agents, justifications provide us with a problem-centered and lazy way of assessing that confidence. Using this idea, we defined an alternative voting system called *Justification Endorsed Voting* (JEV), where the assessed confidence values for member agents are used to weight their respective predictions in the voting system. Then, we defined the *Justification-based Committee Collaboration Strategy* (JE-CS) that uses JEV in order to aggregate the predictions made by the individual agents in a committee. This new way of aggregating solutions has proved to be much more robust, since less competent agents are detected and their weight in the voting system are diminished accordingly.

Finally, we have also worked in committee analysis. For that purpose, we defined the *ensemble space*, a characterization of committees based on three fea-

tures: *completeness*, *bias*, and *redundance*. Using those features, we performed an analysis of several committees with different properties, and determined that a committee should have high completeness, low bias and moderate redundance in order to achieve high performance.

Case Retention

Concerning Case Retention, we introduced three new ideas: *multi-agent retention*, *justification-based case utility* assessment, and *delayed retention* (Figure 10.1).

The idea of multi-agent retention is that when an agent discards a case, is may be worthwhile to offer it to another agent in the $\mathcal{MAC}$ system, since that can improve the individual and committee performance. The *Multi-Agent Case Retention Collaboration Strategy* (MAR-CS) has been presented as a general way to implement that idea. Moreover, MAR-CS is a family of strategies and not a single strategy. Thus, several MAR-CS strategies have been defined, both inspired in active learning and in CBR case retention ideas, namely: NR-NO, AR-NO, OFR-NO, ID-NO, OFR-AO, ID-AO, OFR-AOC, and ID-AOC. Specifically, two strategies, OFR-AOC and ID-AOC (that allowed multiples copies of the same case to be retained by several agents) outperformed the others.

Another idea introduced for case retention is the use of justifications to assess the utility of a case. Thus, we defined the *Justification-based Case Utility* (JCU), that uses justifications to predict whether new cases can improve the performance of a given CBR agent if the cases are added to its case base. To evaluate the effectiveness of JCU, we define the *Justification-based Selection of Training Examples* technique (JUST), that uses JCU in order to reduce the size of a case base without increasing its classification error.

A main issue in retention techniques is that they are usually sensitive to the order in which new cases are considered. In order to mitigate this problem, we introduce the idea of delayed retention. With delayed retention, new cases are stored in a pool, and only when the pool is full, the cases are considered for retention. This allows the agents to use retention strategies that are less eager at discarding or retaining cases. Finally, the three ideas presented introduced for case retention are combined in the *Collaborative Case Bargaining Collaboration Strategy* (CCB-CS), that allows a group of agents to decide which agents should retain each case using a bargaining mechanism. Moreover, experimental results have shown that CCB-CS is a very robust retention strategy, and outperforms all the different versions of MAR-CS.

Case Redistribution

Chapter 9 proposed the idea of *case bartering* to deal with the general problem of case redistribution among the agents in a committee. Case bartering involves the exchange of cases among agents with the purpose of achieving a distribution of cases that improved both individual and committee performance. Moreover,

recall that we are dealing with autonomous agents, and thus two agents will barter cases only if both agents benefit from that barter. We have proposed two alternative strategies to perform case bartering: *bias based case bartering* and *justification based case bartering*.

The bias based case bartering is inspired on the results obtained using the ensemble space analysis, namely that reducing the bias of the agents the performance of the committee tends to increase. Thus, we proposed two collaboration strategies, *CB-CS bias copy* and *CB-CS bias no-copy*. The first one involves bartering of copies of cases and not the cases themselves, and thus increases the redundancy in the system; the second one does not allow copies of cases, and thus does not increase the degree of redundancy in the system. Experimental results have shown that both strategies can improve both individual and committee performance (and that *CB-CS bias copy* achieves better results, since, in addition to decreasing bias, it introduces a moderate degree of redundancy). These experimental results a) reinforce the results obtained with the ensemble space analysis and b) prove that is is feasible to perform case redistribution using decentralized decision making in a way that performance improves.

In justification based case bartering, an agent uses JCU to determine which of its own cases are likely to be useful for another agent. Thanks to JCU, two agents barter cases with high utility for both. Specifically, we have presented the *CB-CS justifications* collaboration strategy, that has shown to greatly improve the individual and committee performance of the agents, even improving the results achieved with *CB-CS bias copy*. This shows that the additional information provided by justifications is useful and that the agents can benefit from it to achieve a more adequate distribution of cases.

10.2 Contributions

In this section we are going to summarily present the contributions of this thesis to the fields of multi-agent learning, Case Based Reasoning and ensemble learning:

- *A framework to study learning processes over distributed data with decentralized decision making.* This framework is based on multi-agent systems, where coordination is achieved by means of collaboration strategies (composed of interaction protocols and individual decision policies), and where data is distributed among the different agents of the system.

- *An analysis of the ensemble effect over distributed data.* The ensemble effect has been studied in ensemble learning for centralized data. However, we study the ensemble effect in a new scenario where data is distributed over a number of predictors and where no assumptions can be made about how data is distributed or even about the number of those predictors.

- *The notion of committee.* A committee is the organizational form of an "ensemble of agents" from the point of view of multi-agent systems. Com-

mittees are ensembles of agents because data is distributed and because the control is decentralized.

- *The definition of the dynamic committees.* Dynamic committees are a lazy and problem-centered approach to "agent ensemble" creation. They are lazy because committees are only convened on demand, and they are problem-centered because ad hoc committees are convened depending on the specific problem. For these two reasons, dynamic committees are an approach well adapted to the dynamic and open nature of multi-agent systems.

- *A framework for agents to learn how to collaborate based on competence models.* In this framework, the task of learning *when* to collaborate and with *whom* to collaborate can be achieved by learning competence models. Competence models can be individually learnt by interacting with other agents, and those competence models can then be used in the agents' decision policies to determine when and with whom to collaborate. Moreover, experimental results show that learnt competence models allow agents to achieve higher performance than predefined competence models.

- *The definition of the ensemble space.* The ensemble space is an analytical tool for characterizing agent ensembles and their performance. We have proposed a set of properties for characterizing agent ensembles, namely *completeness*, *bias*, and *redundancy*. These three properties define the space of possible ensembles, the *ensemble space*.

- *Identification of the directly measurable preconditions that an agent ensemble must satisfy in order to achieve high performance.* Using the ensemble space analysis, we have determined the preconditions that an ensemble must satisfy to achieve a high performance, namely a high completeness, a low bias and a moderate redundancy. Notice that the ensemble space analysis is based only on the properties of the distribution of cases among the agents. Thus, these properties are directly measurable and do not require empirical evaluation. In contrast, the existing formulations of the preconditions of the ensemble effect are based on properties that are not directly measurable, namely that the members of an ensemble must be minimally competent and have low error correlation error (both properties that require empirical evaluation to be measured).

- *An framework for distributed CBR using multi-agent systems ($\mathcal{MAC}$).* $\mathcal{MAC}$ systems use a social agents approach based on electronic institutions, where each individual agent owns a case base. Moreover, our approach preserves the autonomy of the agents and the confidentiality of the individual case bases (in the sense that a specific piece of data can be revealed, but only if the responsible agent decides to do so).

- *The introduction of distributed reuse.* In distributed reuse, each agent individually retrieves cases from its case base, gathering relevant information to solve the problem at hand. Then, the information gathered

by each individual agent is aggregated to build a final prediction about the problem. Moreover, classic aggregation schemes consider only individual predictions aggregation, while distributed reuse is a more general technique where any information that the agents can provide can be considered to generate the final prediction. Thus distributed reuse offers an alternative to distributed retrieval, with the property that each individual agent only retrieves cases from its own case base and thus no disclosure of cases is required, preserving confidentiality of data.

- *The introduction of distributed retain.* Distributed retain improves the performance of committees of agents with respect to performing individual retain. The improvement achieved is due to two main facts. First, because collaboration during retain allows the individual agents to have access to a larger sample of cases during the retain process. And second, because collaboration allows agents to achieve a more adequate distribution of cases among them resulting in a performance improvement when they solve problems individually or acting as an ensemble.

- *The notion of delayed retention.* Delayed retention allows the retain processes to use batches of cases instead of considering cases one by one. Delayed retention strategies achieve higher performance than one by one strategies because delayed retention is less eager and less sensitive to the order in which cases are considered.

- *A bartering framework for data redistribution.* This framework provides a decentralized way in which agents can redistribute cases. The redistribution process in case bartering is the result of individual decision making with respect to which cases are offered and accepted in specific barters. Thus, agent autonomy is preserved, resulting in a suitable process for multi-agent systems.

- *The interpretation of explanations as justifications.* Explanations can be seen as justifications endorsing agent predictions that can be communicated to other agents in order to improve their collaboration and performance. This is a novel idea with respect to the traditional treatment of explanations in CBR, where explanations are simply be provided to human users.

- *The use of justifications to assess the confidence on predictions.* Examination of justifications provides the agents with a problem-centered way to evaluate the confidence on the predictions made by other agents in a committee. Moreover, we have developed a weighted voting system based on justifications where those predictions endorsed by more confident justifications have higher weights.

- *The use of justifications to evaluate the usefulness of cases.* Justifications can be used to identify weak points in the knowledge of lazy learners, and

thus evaluate the usefulness of new cases for them. Moreover, the justification based case utility (JCU) has been successfully applied to improve both case retention and case redistribution strategies.

10.3 Future Work

There are several issues in our research that will be subject of future research. First of all, we would like to perform a complete evaluation of the *ensemble space*, with a larger number of data sets, with the goal of obtaining clues on how to find the optimal degree of redundancy for an agent ensemble. This result is expected to be very useful in both retention and redistribution techniques. We also plan to improve the ensemble space analysis by trying to identify other characteristics (in addition to completeness, bias and redundancy) that can be used to analyze agent ensembles. Moreover, determining the optimal ensemble space point in terms of classification accuracy could lead to the definition of new retention and redistribution strategies that exploit this new information in order to further increase the performance of the agents in a $\mathcal{M}$AC system.

Competence models play an important role in committee collaboration strategies. Specifically, we have seen that agents make decisions using individual decision policies, and that decision policies use competence models and the current situation in order to make decisions. Thus, a future line of research is developing better ways to define and to learn competence models with the goal of improving agent decision making. For instance, better competence models could lead to improvements in the dynamic committee collaboration strategies presented in this work.

The generation of justifications is also a subject for future research. During our work, we have used LID and decision trees algorithms to generate justifications. However, identifying the kind of justifications that other machine learning methods can provide would be interesting. In this way, the justification based techniques that we have presented in this thesis could be applied to a broader range of machine learning techniques.

We have seen that justifications can be examined by other agents. Moreover, these justifications can be seen as *arguments* that endorse agents' predictions. In fact, the Justification Endorsed Voting (JEV) presented in this work can be seen as a basic way of argumentation, where each agent expresses its arguments and the rest of agents examine those arguments in order to assess a confidence for the predictions that those arguments endorse. Thus, we plan to develop further committee aggregation techniques based on more complex argumentation processes, where the members of a committee tries to convince the other agents by expressing their arguments in favor of their individual predictions.

In this work we have been interested on analyzing the ensemble effect in distributed data scenarios. For this purpose, we have experimented with homogeneous $\mathcal{M}$AC systems i.e. systems where all the agents used the same learning methods. In that way, we ensured that the experimental results reflected the improvement on the ensemble effect due only to our collaboration strategies and

not due to the heterogeneity of the learning methods. Indeed, further increase of performance can be achieved using $\mathcal{MAC}$ systems where agents use different learning methods. However, heterogeneous $\mathcal{MAC}$ systems pose new problems. For instance, the ensemble space analysis should be extended in order to consider the heterogeneity of the system. Moreover, it would be interesting to work with justifications in heterogeneous $\mathcal{MAC}$ systems since those justifications may not be directly comparable.

We are also interested in applying our techniques to other domains than classification, such as regression or planning. These new domains raise several new challenges. First of all, it is not obvious how to aggregate solutions in a planning domain, thus new ways to aggregate individual solutions have to be investigated. Some possible ways to do this could be plan merging or the division of the creation of a plan in several classification sub-tasks where voting can be applied.

Another problem is that bias is measured in classification domains with respect to the partition induced by the solution classes; this partition, however, is not available in regression or planning domains. Therefore, a way to define a reference partition with which to compute bias or a completely new way to compute bias should be developed. Possible ways to do this could be related to the definition of similarity relations in the solution space.

Another problem is how to generate and examine justifications to endorse predictions in regression or planning domains. The main problem with justifications is that most of the justification based techniques proposed in this work rely on the idea that the *correctness* of a solution can be assessed in a boolean way, however this is not true for regression nor planning. In fact, in planning domains, it is likely that several plans are correct, but that some of them are better than others. For that reason, correctness predicates should take into account these preference relations among solutions.

Finally, more complex argumentation processes will surely be needed in order to aggregate justified predictions in a planning domain, that would involve plan critiquing, alternative option discussion, preferences negotiation or plan revision.

Appendix A

Notation

This appendix summarizes all the notation used throughout this thesis.

A.1 General Notation

- $\mathcal{M}$: denotes a $\mathcal{M}$AC system.
- A_i: is used to denote an agent.
- $\mathcal{A}$: denotes a set of agents.
- C_i: denotes the case base of the agent A_i.
- $c = \langle P, S \rangle$: denotes a specific case, composed of a problem description and of a solution class.
- $\mathcal{S}$: denotes the set of possible solution classes in a domain.
- S_i: denotes a specific solution class.
- K: is the number of possible solution classes in a domain.
- D_i: is used to denote a decision policy.
- I_{CS}: is used to denote an interaction protocol for a specific collaboration strategy CS.
- w_i: is used to denote a specific state of an interaction protocol.
- Γ: is used to denote the set of sorts that can be used in the feature terms.
- γ_i: denotes a specific sort.
- $\perp$: denotes the more general sort *any*.
- ψ: is used to denote a feature term.

- $\tau(\psi)$: is used to denote the root node of a feature term.
- $\sqsubseteq$: denotes the *subsumption* relation among feature terms.

A.2 Ensemble Space Notation

- $\mathcal{P}$: denotes a problem space.
- $\mathcal{D}$: denotes a data-set.
- $\mathbb{C}$: denotes the *completeness* of a case base or of a committee in the terms of the *ensemble space analysis*.
- $\mathbb{B}_{\Pi}$: denotes the *bias* of a case base (or the average bias the case bases of the agents in a committee) with respect to a partition of the problem space Π.
- $\mathbb{B}_{\mathcal{S}}$: denotes the *bias* with respect to the partition induced by the set of solution classes $\mathcal{S}$.
- $\mathbb{B}$: is sometimes used as $\mathbb{B}_{\mathcal{S}}$ for short.
- $\mathbb{R}$: denotes the *redundancy* in a committee of agents.

A.3 Committees Notation

- A_c: denotes the *convener agent* of a committee.
- $\mathcal{A}^c$: denotes a committee of agents convened by a convener agent A_c.
- $\mathcal{A}^c_t$: denotes a committee of agents convened at a round t of a dynamic committee collaboration strategy.
- $\mathcal{A}^r_t$: denotes the candidate agents to be invited to join a committee at the round $t+1$ of a dynamic committee collaboration strategy.
- $\mathbf{R} = \langle S, E, P, A \rangle$: denotes a *solution endorsement record* (SER), composed of a solution class, the number of endorsing cases, a problem, and the agent who has generated the record.
- $\mathcal{R}_{A_i}$: is used to denote the set of SERs generated by an agent A_i while solving a specific problem.
- $\mathcal{R}_{\mathcal{A}}$: is used to denote the set of SERs generated by a set of agents $\mathcal{A}$ while solving a specific problem (this is also called a *voting situation*).
- $\mathcal{R}^c$: is used to denote the set of all the SERs generated by all the agents in a committee $\mathcal{A}^c$.

A.4 Voting Systems

- $Vote(S_k, P, A_i)$: is used to denote the votes of an agent A_i for a solution class S_k to be the correct solution of problem P.
- $Ballot(S_k, P, \mathcal{A}^c)$: the sum of all the votes for solution class S_k casted by the agents in $\mathcal{A}^c$ for the problem P.
- $Sol(\mathcal{S}, P, \mathcal{R})$: is used to denote the solution class resulting from applying the voting system to the set of SERs $\mathcal{R}$.

A.5 Proactive Learning Notation

- $\mathbb{V}(A_i)$: is used to denote the set of all the possible valid voting situations for an agent A_i.
- $\mathbb{A}(A_i)$: is used to denote the set of all the possible subsets of agents of the current committee $\mathcal{A}^c$ that at least contain A_i.
- M_{A_i}: denotes a competence model of an agent A_i.
- $M_{\mathcal{A}}$: denotes a competence model of a set of agents $\mathcal{A}$.
- $m_i = \langle A_1, ..., A_n, S^c, V^c, V^r, \rho \rangle$: is denotes an *M-example*, i.e. an example to learn a competence model M.
- T_M: denotes a training set composed of *M-examples* that can be used to learn a competence model M.
- l: denotes a specific leaf in a confidence tree.
- p_l: is used to denote the confidence of a voting situation classified in a leaf l of a confidence tree.
- p_l^-: is the pessimistic estimation of p_l.
- p_l^+: is the optimistic estimation of p_l.

A.6 Justifications Notation

- J: is used to denote a justification.
- $\mathbf{J} = \langle S, J, P, A \rangle$: denotes a *justification endorsement record* (JER), composed of a solution class, a justification, a problem, and the agent who has generated the record.
- $EC(\mathbf{J}, C)$: denotes the set of endorsing cases of a JER $\mathbf{J}$ contained in a case base C.

- $CE(\mathbf{J}, C)$: denotes the set of counterexamples of a JER $\mathbf{J}$ contained in a case base C.
- $VCE(\mathbf{J}, C)$: denotes the set of valid counterexamples of a JER $\mathbf{J}$ contained in a case base C.
- $\mathbf{X} = \langle \mathbf{J}, Y, N, A \rangle$: denotes an *examination record* (XER), composed of a JER, the number of endorsing cases and of counterexamples and the agent who has generated it.
- $\mathcal{J}_{A_i}$: is used to denote the set of JERs generated by an agent A_i while solving a specific problem.
- $\mathcal{J}_{\mathcal{A}}$: is used to denote the set of JERs generated by a set of agents $\mathcal{A}$ while solving a specific problem.
- $C_{\mathcal{A}^c}(\mathbf{J})$: denotes the overall confidence of a JER computed by a committee of agents $\mathcal{A}^c$.
- $JVote(S_k, P, A_i)$: is used to denote the justification endorsed votes of an agent A_i for a solution class S_k to be the correct solution of problem P.
- $JBallot(S_k, P, \mathcal{A}^c)$: the sum of all the justification endorsed votes for solution class S_k casted by the agents in $\mathcal{A}^c$ for the problem P.
- $JSol(\mathcal{S}, P, \mathcal{R})$: is used to denote the solution class resulting from applying the justification endorsed voting system to the set of SERs $\mathcal{R}$.

A.7 Retention Notation

- E: is used to denote an exam.
- C_E: denotes the set of cases from which an exam E has been generated.
- C_t^r: is used to denote the reduced case base at a round t of JUST.
- C_t^u: is used to denote set of unseen cases at a round t of JUST.
- $JCU(C, C_E)$: is used to denote the JCU utility of a set of cases C computed with respect to a set of examination cases C_E.
- $\mathcal{T}$: is used to denote the termination criterion of the JUST method.
- $R_t^{\mathbf{J}}$: denotes the refutation set of a JER $\mathbf{J}$ at a round t of JUST.
- B_t: denotes the belying set at a round t of JUST.
- B_i: is used to denote the pool of delayed retention cases of an agent A_i.
- $\widehat{B}$: denotes the union of the pools of delayed retention cases of all the agents in a committee.

- $\widehat{B}_t$: denotes the cases remaining to the cases remaining to bargain for at a round t of the CCB-CS collaboration strategy.
- $\mathbf{U} = \langle A, C, V \rangle$: denotes an utility record containing the agent who has computed the record, the case and the utility value.
- $\mathcal{U}_t^{A_i}$: is used to denote the set of utility records generated by an agent A_i at a round t of the CCB-CS collaboration strategy.
- $\overline{U}_t$: denotes the utility record with maximum utility value at a round t of the CCB-CS collaboration strategy.

A.8 Bartering Notation

- $d_{j,k}$: denotes the fraction of cases in the case base of agent A_j that belong to the solution class S_k.
- d_k: denotes the expected fraction of classes with solution S_k in the application domain.
- $\mathbf{O} = \langle S_O, S_R, A_O, S_R \rangle$: is used to denote a bartering offer in the bias based case bartering.
- $\mathcal{R}_{A_j}^t$: denotes the collection of refutation sets computed for an agent A_j in a round t of the justification based bartering.
- $\mathbb{R}_{A_i}^t$: denotes the set of the collections of refutation sets computed by an agent A_i in a round t of the justification based bartering.
- $\mathbf{Q} = \langle D_{R \to O}, A_R, A_O \rangle$: is used to denote a bartering request in the justification based case bartering.
- $\mathbf{O} = \langle D_{O \to R}, D_{R \to O}, A_O, A_R \rangle$: is used to denote a bartering offer in the justification based case bartering.
- $B_{i \to j}^t$: the set of cases that an agent A_i will send to an agent A_j in round t the justification based bartering.
- $D_{i \to j}^t$: denotes the case descriptions of all the cases in $B_{i \to j}^t$.

Appendix B

The NOOS Agent Platform

In this appendix we are going to briefly describe the NOOS Agent Platform (NAP), with which we have performed all the experiments reported in this work and that has been developed at the Artificial Intelligence Research Institute (IIIA) of the Spanish Council for Scientific Research (CSIC).

B.1 Overview

The NOOS Agent Platform (NAP), is a LISP based development environment specifically designed to create knowledge rich Case Based Reasoning agents in the NOOS representation language based that can communicate and cooperate using the technology of *Agent Mediated Institutions*.

The NOOS representation language was developed in the Analog Project and in the Massive Memory Project, that aimed at developing systems that integrate problem-solving methods with learning from experience, and has been successfully used to develop AI applications related to music [5], and to medical domains [8] among others. Later, NOOS was extended to support multi-agent systems using the technology of agent mediated institutions [27], constituting the NOOS Agent Platform, that was developed in the COMRIS, IBROW, E-institutor and Tabasco projects.

Specifically, NAP is mainly composed of two elements:

- The NOOS representation language, based on *feature terms* and that provides tools to create CBR agents.
- A FIPA compliant agent platform, that allows the user to define NOOS agents that can communicate and coordinate among them and with external agents.

Let us briefly describe the NOOS representation language and the agent platform.

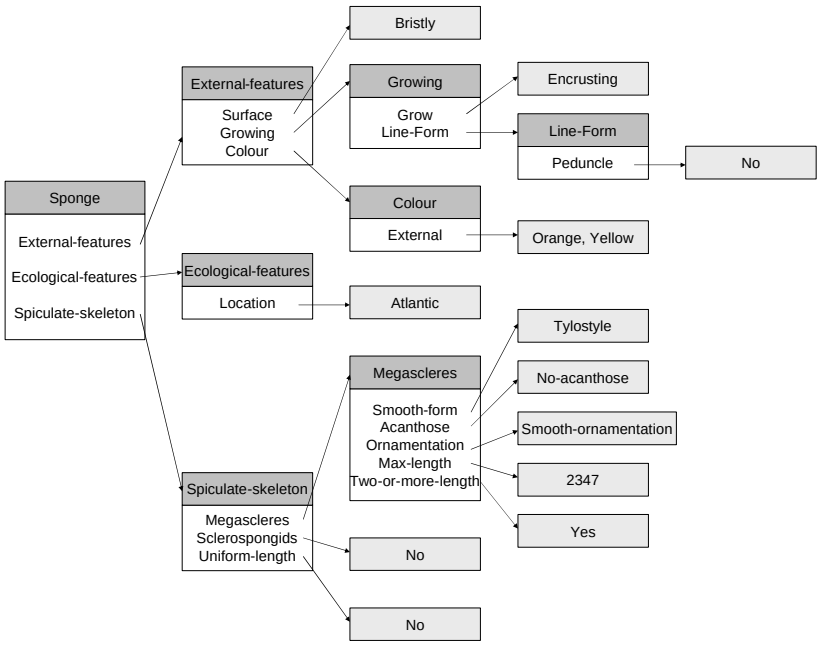


Figure B.1: Example of a simplified sponge represented using feature terms.

B.2 Knowledge Representation

NOOS uses the feature term formalism to represent knowledge (see Section 3.2.1) and provides a LISP-like language to specify feature terms.

Specifically, three elements constitute the knowledge representation language of NOOS: *ontologies*, *domain model* and *terms*:

- An *ontology* is a hierarchy of *sorts* that describes a specific domain.
- The *domain model* of a specific domain is constituted in NOOS by a set of predefined instances of the sorts specified in the corresponding ontology.
- Finally, a *term* describe an individual in a specific domain.

Let us illustrate this with an example in the sponges domain used in our experiments. Figure B.1 shows a simplified sponge of the sponge data set (in the sponge data set, the problem is to classify a new sponge in its correct biological order: astrophorida, hadromerida or axinellida). Figure B.1 shows a series of boxes. Each box represents a node of the feature term. Each node has a *sort* (that in the figure is represented with a gray background on the top of the boxes) and a set of features (on the lower part of the boxes). The root node of the feature term is the left-most node in the figure. Specifically, Figure B.1 shows a sponge that has three features (`external-features`, `ecological-features`

```
(define-sort (sponge)
   (external-features external-features)
   (ecological-features ecological-features)
   (spiculate-skeleton spiculate-skeleton)
   (fibrous-skeleton fibrous-skeleton)
   (tracts-skeleton tracts-skeleton)
   (anatomy anatomic-features))
(define-sort (spiculate-skeleton)
   (chemical chemical)
   (architecture architecture)
   (megascleres megascleres)
   (microscleres micros)
   (sclerospongids boolean)
   (hexactinellids hexactinellids)
   (uniform-length boolean))
(define-sort (micros))
(define-sort (micros aster))
(define-sort (aster sterraster)
   (type form)
   (deformed boolean)
   (sterr-diameter number))
(define-sort (aster oxyaster)
   (deformation boolean)
   (max-diameter number)
   (two-length-categories boolean))
...
```

Figure B.2: NOOS definition of the ontology needed to define the feature term depicted in Figure B.1.

and `spiculate-skeleton`); each feature value has its own features, etc. For instance, the feature `ecological-features` has as a value a node that is of the sort `ecological-features`, that has a single feature, `location`, that has as value `atlantic`, that is defined in the domain model.

Now, let us see how to define that feature term in NOOS. First of all, the ontology has to be defined. The ontology required for the sponge data set is huge, and thus only a fragment will be presented here. Figure B.2 shows the specification of a part of the sponge ontology in the NOOS representation language. Initially, the sort `sponge` is defined, as having 6 features (`external-features`, `ecological-features`, `spiculate-skeleton`, `tracts-skeleton` and `anatomy`). Each feature is specified as a pair, for instance, the pair `(anatomy anatomic-features)`, defines a feature named `anatomy` and that takes as values terms of the sort `anatomic-features`. After that, the `spiculate-skeleton` sort is defined. Then, the `micros` sort, and after that, the `micros` sort is specialized in the

```
...
(define (aster :id spheraster))
(define (aster :id chiaster))
(define (aster :id strongylaster))
(define (aster :id selenaster))
(define (aster :id spiraster))
(define (aster :id diplaster))
(define (aster :id amphiaster))
(define (aster :id sanidaster))
(define (aster :id metaster))
(define (aster :id plesiaster))
...
```

Figure B.3: Part of the NOOS definition of the domain model for the marine sponge identification domain.

`aster`, `sterraster` and `oxiaster`. Notice that when a sort is specialized, further features can be defined. For instance, the sort `oxiaster` defines three extra features not present in the sort `aster`, namely, `deformation`, `max-diameter`, and `two-length-categories`.

Once the ontology is defined, the next step is to define the domain model. The domain model in NOOS is constituted by a set of predefined instances of the sorts defined in the ontology. For instance, Figure B.3 defines a lot of different instances of the sort `aster` (defined in the ontology). Notice that the instances in the domain model had an identifier. The identifier is used to make reference to them when defining terms. For instance, the first line defines an instance called `spheraster` of the sort `aster`.

Finally, once the ontology and the domain model are defined, terms can be defined. Figure B.4 shows the NOOS expression to define the sponge depicted in Figure B.1. All the sorts used in the definition of terms must have been defined in the ontology. Moreover, notice that the term makes also reference to the instances defined in the domain model. For instance, the value `bristly` found in `external-features.surface` must have been defined in the domain model as an instance of the proper sort. Notice also that the sort `sponge` had 6 features defined in the ontology, and that the definition of Figure B.4 only defines three of them (`external-features`, `ecological-features`, and `spiculate-skeleton`); the non specified features are assumed to have the value `any` (that represents the minimum information in NOOS).

Moreover, NOOS gives support to define case bases (that are called *episodic memories* in NOOS). Each individual agent defined in NOOS has its own episodic memory, and can retrieve cases (expressed as terms) from it. Moreover, NOOS also implements the subsumption relation (that defines an information order among terms) and many other tools to easily visualize, create and manipulate terms. Thus, the NOOS representation language, provides an excellent framework to develop knowledge intensive applications that combine learning and

```
(define (sponge)
   (external-features
      (define (external-features)
         (surface bristly)
         (growing
            (define (growing)
               (grow encrusting)
               (line-form
                  (define (line-form)
                     (peduncle no)))))
         (colour
            (define (colour)
               (external
                  (define (set) orange yellow))))))
   (ecological-features
      (define (ecological-features)
         (location atlantic)))
   (spiculate-skeleton
      (define (spiculate-skeleton)
         (megascleres
            (define (megascleres})
               (smooth-form tylostyle)
               (acanthose no-acanthose)
               (ornamentation smooth-ornamentation)
               (max-length 2347)
               (two-or-more-length yes)))
               (sclerospongids no)
               (uniform-length no))))
```

Figure B.4: NOOS definition of the feature term depicted in Figure B.1.

problem solving.

B.3 Agent Platform

The NOOS Agent Platform gives support for agent communication and coordination using the framework of *agent mediated institutions* [27]. Agents are implemented in NAP by means of four elements: *activities*, *scenes*, *behaviors*, and *problem solving methods* (PSM).

- Problem Solving Methods (PSMs): are the primitive inference units. For instance, CBR methods that solve new problems are PSMs.
- Scenes: define the interaction among a group of agents to perform a certain task. A scene is fully specified by an interaction protocol that defines a

```
(define-noos-scene BARTER-CASES-SCENE
  :description "Scene for two agents that barter cases"
  :roles (AI AJ)
  :states (W0 W1 W2)
  :initial-state W0
  :final-states (W2)
  :connections ((W0 W1 (Inform (?I AI) (?J AJ) b-message))
                (W1 W2 (Inform (!J AJ) (!I AI) b-message))))
```

Figure B.5: NOOS definition of the interaction protocol that two agents (AI and AJ) follow to barter two cases.

set of *roles* that the agents may play, a set of *interaction states* that the agents playing different roles may be in, and a set of *messages* that each agent is allowed to send in each interaction state.

- Behaviors: define how a specific agent behaves in a specific protocol I. A behavior is defined by a *behavior-method* for each possible interaction state in I. The behavior methods decide the actions that an agent will perform at each interaction state, including sending messages. Moreover, behavior-methods use PSMs to make decisions.

- Activities: defines the tasks that an agent is able to manage. An activity specifies which protocols an agent knows, and which behaviors will an agent use to follow each known protocol playing a specific role.

Let us illustrate, all the previous elements with a simple example. We will show the implementation of the interaction protocol followed by two agents that want to barter two cases (recall the case bartering strategy presented in Chapter 9):

Figure B.5 presents the definition in NAP of the scene for two agents that want to barter two cases, the `BARTER-CASES-SCENE`. Two roles (`AI` and `AJ`), and three interaction states (`W0`, `W1`, and `W2`) are defined. Then, state `W0` is defined as the initial state (that state in which each agent will be at the beginning of the protocol), and state `W2` is regarded as a final state (i.e. the protocol will end when reaches that state). Finally, `connections` defines the set of messages that can be send in each of the interaction states by each of the roles in the protocol. Specifically, this simple protocol only allows two messages, one in state `W0` from `AI` to `AJ` (that will contain the case that `AI` gives to `AJ`), and another in state `W1` from `AJ` to `AI` (containing the case that `AJ` gives to `AI`). Intuitively, the protocol is designed to work as follows: first the agent playing role `AI` will send a case to the agent playing the role `AJ`; after that, `AJ` will answer to `AI` with another case and the protocol will end.

The next step is to define the *activities*. Recall that an activity defines which behavior should be used by an agent to play a specific role in an interaction protocol. Thus, activity definition is simple in NOOS, as can be seen

```
(define-Activity AI-BARTER-CASE-ACTIVITY
   :implements ((AI BARTER-CASES-SCENE)))
(define-Activity-Method AI-BARTER-CASE-ACTIVITY (W args)
  (use-behavior 'AI-Barter-Cases-Behavior args))

(define-Activity AJ-BARTER-CASE-ACTIVITY
   :implements ((AJ BARTER-CASES-SCENE)))
(define-Activity-Method AJ-BARTER-CASE-ACTIVITY (W)
  (use-behavior 'AJ-Barter-Cases-Behavior))
```

Figure B.6: NOOS definition of the two activities required to play each of the two roles defined in the protocol specified in Figure B.5

in Figure B.6, where the two activities for playing roles `AI` and `AJ` are specified. The first activity `AI-BARTER-CASE-ACTIVITY` specifies that the behavior `AI-Barter-Cases-Behavior` will be used as the behavior to play the role `AI` in the scene `BARTER-CASES-SCENE`. The second activity `AJ-BARTER-CASE-ACTIVITY` specifies that the behavior `AJ-Barter-Cases-Behavior` will be used as the behavior to play the role `AJ` in the scene `BARTER-CASES-SCENE`.

Figure B.7 shows the NOOS definition of the two behaviors referenced in the activities. Namely, `AI-Barter-Cases-Behavior` and `AJ-Barter-Cases-Behavior`. Notice, that the behavior for the role `AI` defines a behavior-method for two states (`W0` and `W2`) while the behavior for the role `AJ` only defines a behavior-method for the state `W1`. This is because the agent playing the role `AI` has to perform actions in states `W0` and `W2` and the agent playing the role `AJ` only has to perform any action in the state `W1`.

Figure B.7 shows that the agent playing the role `AI` in state `W0` will choose a case to be sent to `AJ`, delete it from its case base if the bartering is performed in the *non-copy* mode, and send it to `AJ`. In state `W2`, the agent playing the role `AJ` only has to retain into its case base the case received from `AJ`. In state `W1` the agent playing the role `AI` does not have to perform any action, just wait for `AJ`. Moreover, the agent playing the role `AJ` in state `W1` has to retain the case received from `AI` to its case base, choose a case to send to `AI`, remove it from its case base if the bartering is performed in the *non-copy* mode, and finally send the selected case to `AI`.

The methods referred to in the behavior methods (such as `Choose-case-to-send`, etc.) are the problem solving methods, that are basically LISP functions that solve the problems that an agent has to solve in order to perform a certain task with a given behavior. For the behaviors defined in this example, just three PSMs need to be defined: `choose-case-to-send`, that chooses a case from the agents case base to be sent to another agent; `add-to-cb`, that retains a given case into the agent's case base; and `remove-from-cb` that deletes a given case form the agent's case base.

```
(define-behavior AI-Barter-Cases-Behavior
  :implements (AI BARTER-CASES-SCENE))

(define-behavior-Method AI-Barter-Cases-Behavior ((W W0) args)
  (let* ((aj (first args))
         (case (Choose-case-to-send aj)))
    (when (eq bartering-mode 'non-copy) (remove-from-cb case))
    (send-acl 'INFORM aj (define (b-message) (case-offered case)))))

(define-behavior-Method AI-Barter-Cases-Behavior ((W W2) (m message))
  (let* ((case-received m.content.case-offered)))
    (add-to-cb case-received)))

(define-behavior AJ-Barter-Cases-Behavior
  :implements (AJ BARTER-CASES-SCENE))

(define-behavior-Method AJ-Barter-Cases-Behavior ((W W1) (m message))
  (let* ((ai m.sender)
         (case-received m.content.case-offered))
         (case (Choose-case-to-send ai))
    (add-to-cb case-received)
    (when (eq bartering-mode 'non-copy) (remove-from-cb case))
    (send-acl 'INFORM ai (define (b-message) (case-offered case)))))
```

Figure B.7: NOOS definition of the behaviors to play the roles AI and AJ in the protocol specified in Figure B.5

Finally, once all the scenes, activities, behaviors and problem solving methods are defined, we can define the agents. Figure B.8 shows the definition in NAP of a simple agent that can use the two activities defined in this exemplification, and that is able to understand the sponge ontology.

Figure B.9 shows a screenshot of the the NAP development environment. On the upper part of the screen you can see the different tabs related to agents, protocols, activities, behaviors, methods, ontologies, and domain knowledge. Specifically, the screenshot shows the the protocol view, where the scenes and protocols that have been defined can be graphically viewed.

```
(define-Agent CBR-Agent
  (activities AJ-Barter-Case-Activity
              AI-Barter-Case-Activity)
  (ontologies Sponge-Ontology )
  (Domain-Models Sponge-DM))
```

Figure B.8: NOOS definition of an agent that is able to deal with the sponge domain and to perform some activities.

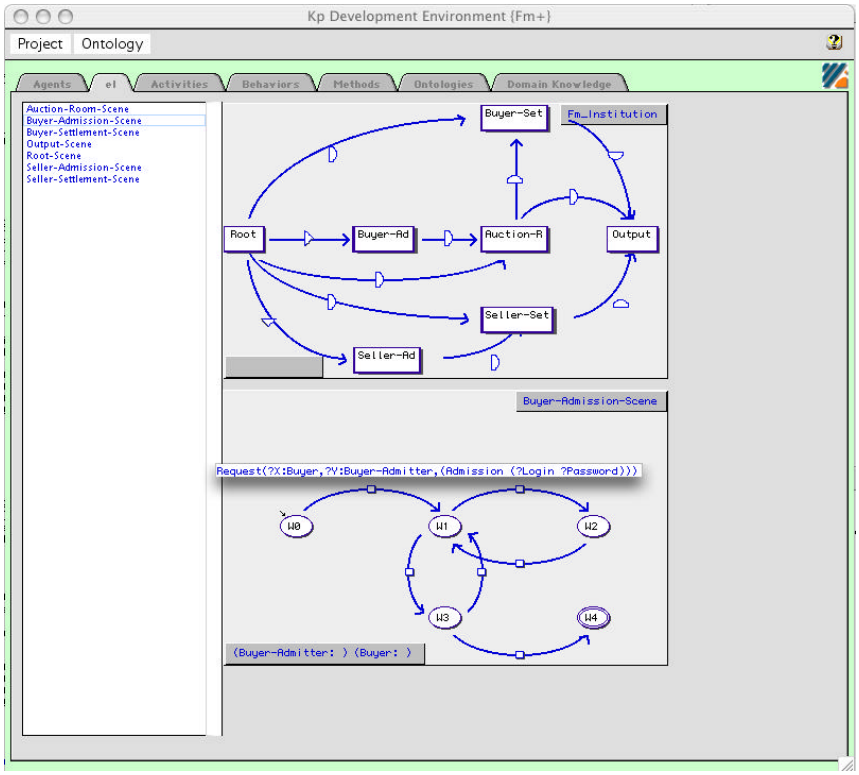


Figure B.9: Screenshot of the NOOS Agent Platform development environment.

Appendix C

Probability Estimation

This appendix will explain in more detail the technique used in chapters 6 and 8 to estimate probabilities. Specifically, Chapter 6 uses probability estimation to learn competence models of individual agents and of committees of agents for the PB-CCS collaboration strategy, and Chapter 8 uses probability estimation to assess the classification accuracy of a CBR system in the JUST method.

Let us formalize the problem of probability estimation. Let V be a variable taking values in a set $\{x_1, ..., x_n\}$. Moreover, we have a set of observations $O = \{o_1, ..., o_m\}$ of the values that V has taken in the past. The goal is to compute the probability of each of the n possible values that V can take. We will call θ_i to the probability of the value x_i, and thus the goal is to compute the set $\theta = \{\theta_1, ..., \theta_n\}$ (notice that $\theta_n = 1 - \sum_{i=1...n-1} \theta_i$, and thus we have only $n - 1$ parameters to estimate).

A first solution to the previous problem can be found by simply performing a frequency count of the different values x_i on the observations. However, it would be desirable that a measure of how confident are the estimations made on θ_i could be provided. Therefore, a more complex approach has to be taken.

In order to provide an estimation of how confident our probability estimation is, the solution is to compute a *likelihood* function $p(\theta|D)$ instead of computing θ directly. Such likelihood function will predict how probable is a concrete instantiation of θ given the observed data D.

To compute $p(\theta|D)$ we will simply use the Bayes' Formula. We will start with an initial distribution $p(\theta|\xi)$, where ξ represents our prior knowledge (if we have no prior knowledge about θ, then $p(\theta|\xi)$ will simply consist on a uniform density probability function). Then, we will use the Bayes' Formula to update $p(\theta|\xi)$ given the observed data.

Thus, if D consists of a single observation $o_1 = \langle V = x_i \rangle$, using the Bayes' Formula we would obtain the following likelihood function:

$$p(\theta|V = x_i, \xi) = \frac{p(V = x_i|\theta, \xi)p(\theta|\xi)}{p(V = x_i|\xi)}$$

by definition $p(V = x_i|\theta, \xi) = \theta_i$, and thus:

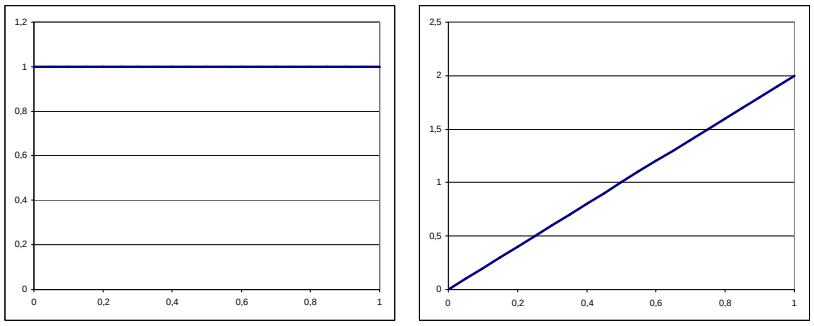


Figure C.1: Left, $p(\theta_1|\xi)$ as a uniform density probability function; Right, $p(\theta_1|D_1, \xi)$ for a single observation where $V = correct$.

$$p(\theta|V = x_i, \xi) = \frac{\theta_i p(\theta|\xi)}{p(V = x_i|\xi)}$$

Now, since $p(V = x_i|\xi)$ does not depend on θ, it can be considered a normalization constant c, and obtain:

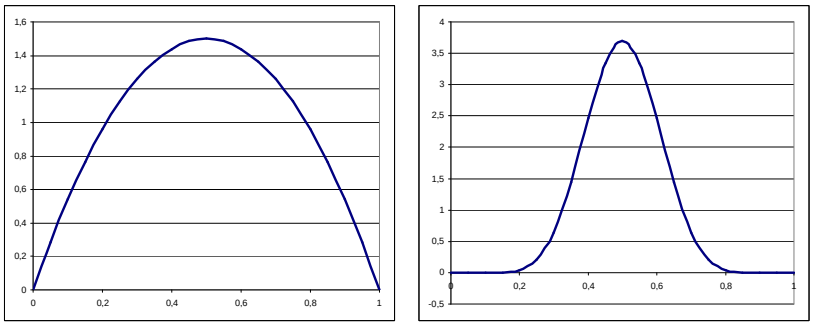
$$p(\theta|V = x_i, \xi) = c\theta_i p(\theta|\xi)$$

It is now easy to derive the general formula for the case that D consists of a set of m observations (where each value x_i has been observed α_i times):

$$p(\theta|D, \xi) = c' \prod_{i=1}^{n} \theta_i^{\alpha_i}$$

Let us illustrate the derived formula with an exemplification before proceeding to explain how to compute the estimations of the confidence. Assume a binary variable V that can take the values $\{correct, incorrect\}$, such variable corresponds to a CBR agent, and takes the value *correct* when the CBR agent correctly answers a problem, and *incorrect* otherwise. Thus, we have to estimate $\theta = \{\theta_1, \theta_2\}$, and since $\theta_2 = 1 - \theta_1$, we just have to estimate θ_1, that corresponds to the probability of the value *correct* (i.e. the classification accuracy of the CBR agent). Moreover, let us assume that we have no prior knowledge of the CBR agent, thus $p(\theta|\xi)$ is a uniform density probability function. Moreover, since we have just to estimate θ_1, we will consider only $p(\theta_1|\xi)$ and $p(\theta_1|D, \xi)$ in this example.

The left plot of Figure C.1 shows $p(\theta_1|\xi)$ as a uniform density probability function. The right plot of Figure C.1 shows $p(\theta_1|D_1, \xi)$ where D_1 consists of one observation $o_1 = \langle V = correct \rangle$. Figure C.1 shows that with a single observation of $V = correct$, the most probable value for θ_1 is 1 (that corresponds



1 observation of correct
1 observation of incorrect

10 observations of correct
10 observations of incorrect

Figure C.2: Left, $p(\theta_1|D_2, \xi)$ for two observations;Right, $p(\theta_1|D_3, \xi)$ for twenty observations.

to a classification accuracy of 100%), however, there are many other values with probability higher than 0.

The left plot of Figure C.2 shows $p(\theta_1|D_2, \xi)$ where D_2 consists of two observations $o_1 = \langle V = correct \rangle$ and $o_2 = \langle V = incorrect \rangle$. The most probable now is that $\theta_1 = 0.5$ (corresponding to a classification accuracy of 50%). Moreover, the right plot of Figure C.2 shows $p(\theta_1|D_3, \xi)$ where D_3 consists of 20 observations, in 10 of them V has taken the value *correct*, and in 10 of them V has taken the value *incorrect*. Notice that again the most probable is that $\theta_1 = 0.5$ (corresponding to a classification accuracy of 50%), but now the plot shown for 20 observations (right hand side) is much narrower than the plot shown for 2 observations (left hand side).

The probability that θ_1 takes value in a certain interval $[a, b]$ can be obtained by computing the area below the plot in the interval $[a, b]$:

$$P(\theta_1 \in [a, b]|D, \xi) = \int_a^b p(\theta_1 = x|D, \xi)dx$$

For instance, if we compute the probability that θ_1 lies in the interval $[0.4, 0.5]$ in both plots (corresponding the the probability that the classification accuracy is between 40% and 50%), we obtain that in the left hand side plot, the probability is $P(\theta_1 \in [0.4, 0.6]|D_2, \xi) = 0.2945$, while in the right hand side plot it is $P(\theta_1 \in [0.4, 0.6]|D_3, \xi) = 0.6423$. Therefore, if we have 20 observations, we can be more confident that θ_1 will lie in the interval $[0.4, 0.6]$ than with just 2 observations. Thus, we can say that "with 10 *correct* observations and 10 *incorrect* observations we have a 64.23% of certainty that the classification accuracy will be between 40% and 60%, and the most probable value is 50%".

Thus, we now have the tools to mathematically prove the statement made in Chapter 8: "for estimating accuracy values around $\alpha = 90\%$ (having a certainty of 66% of having an error lower than the 4%) at least 60 answers are required".

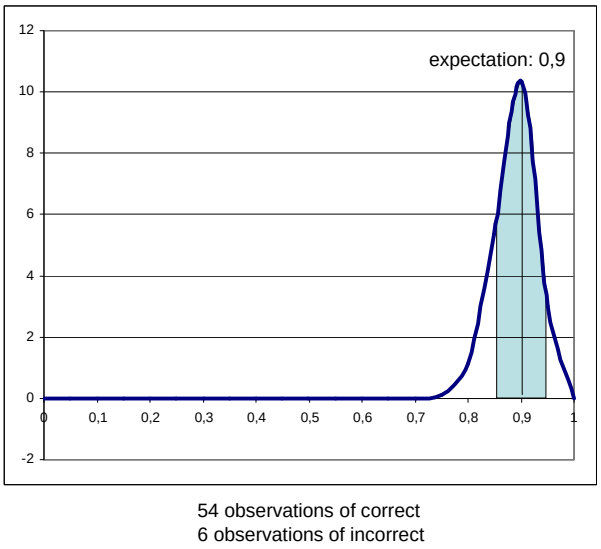


Figure C.3: $p(\theta_1|D_4, \xi)$ for a 60 observations where $V = correct$ in the 90% of the observations. The blue area corresponds to the area below the plot in the interval $[0.86, 0.94]$ (see the text).

Figure C.3 shows the likelihood function $p(\theta_1|D_4, \xi)$ where D_4 consists of 60 observations (where the 90% of them where *correct* and the 10% of them where *incorrect*). The blue area represents the area of the interval $[0.86, 0.94]$ (that corresponds to having an error of $\pm 4\%$ in the estimation of the classification accuracy). This area can be easily computed numerically, and is approximately 0.6864. Thus, we can conclude that the probability that θ_1 lies in the interval $[0.86, 0.94]$ is 0.6864, and therefore we have a certainty higher than the 66% (in fact, it is of the 68.64%) of having an error lower than the 4% if we estimate the accuracy to be 90%.

Moreover, similar computations are used in Chapter 6 to compute the confidence intervals $[p_l^-, p_l^+]$ at the leaves of the confidence tree. Specifically, the interval $[p_l^-, p_l^+]$ is computed so that the area below the curve in the interval is 0.66 (i.e. having a certainty of 66% than the estimated value lies in that interval).

Bibliography

[1] Agnar Aamodt and Enric Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *Artificial Intelligence Communications*, 7(1):39–59, 1994. online at <url:http://www.iiia.csic.es/People/enric/AICom_ToC.html>.

[2] David W. Aha. Case-based learning algorithms. In *DARPA Case-Based Reasoning Workshop*, pages 147–158.

[3] David W. Aha, Dennis Kibler, and Marc K. Albert. Instance-based learning algorithms. *Machine Learning*, 6(1):37–66, 1991.

[4] Josep Lluís Arcos. *The Noos representation language*. PhD thesis, Universitat Politècnica de Catalunya, 1997.

[5] Josep Lluís Arcos, Ramon López de Mántaras, and Xavier Serra. Saxex : a case-based reasoning system for generating expressive musical performances. *Journal of New Music Research*, 27 (3):194–210, 1998.

[6] Josep Lluís Arcos and Enric Plaza. Inference and reflection in the object-centered representation language Noos. *Journal of Future Generation Computer Systems*, 12:173–188, 1996.

[7] Shlomo Argamon-Engelson and Ido Dagan. Committee-based sample selection for probabilistic classifiers. *Journal of Artificial Intelligence Research*, 11:335–360, 1999.

[8] E. Armengol and E. Plaza. Individual prognosis of diabetes long-term risks: A CBR approach. *Methods of Information in Medicine*, page to appear, 2001.

[9] E. Armengol and E. Plaza. Lazy induction of descriptions for relational case-based learning. In *Submitted*, 2001.

[10] Bikramjit Banerjee and Jing Peng. Adaptive policy gradient in multiagent learning. In *Int. Conf. Autonomous Agents and Multiagent Systems AAMAS'03*, pages 686–692, 2003.

[11] Bikramjit Banerjee, Sandip Sen, and Jing Peng. Fast concurrent reinforcement learners. In *IJCAI 2001*, pages 825–830, 2001.

[12] Stephen D. Bay. Combining nearest neighbor classifiers through multiple feature subsets. In *Proc. 15th International Conf. on Machine Learning*, pages 37–45. Morgan Kaufmann, San Francisco, CA, 1998.

[13] Michael Bowling and Manuela Veloso. Simultaneous adversarial multi-robot learning. In *IJCAI 2003*, 2003.

[14] Michael H. Bowling and Manuela M. Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136(2):215–250, 2002.

[15] Steven J. Brams and Peter C. Fishburn. *Approval Voting*. Birkhauser, Boston, 1983.

[16] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[17] D. Caragea, A. Silvescu, and V. Honavar. Decision tree induction from distributed heterogeneous autonomous data sources, 2003.

[18] Alison Cawsey. Explanation and interaction. the computer generation of explanatory dialogues., 1992.

[19] Philip K. Chan and Salvatore J. Stolfo. A comparative evaluation of voting and meta-learning on partitioned data. In *Proc. 12th International Conference on Machine Learning*, pages 90–98. Morgan Kaufmann, 1995.

[20] David A. Cohn, Les Atlas, and Richard E. Ladner. Improving generalization with active learning. *Machine Learning*, 15(2):201–221, 1994.

[21] David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. Active learning with statistical models. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 705–712. The MIT Press, 1995.

[22] Gerald Dejong and Raymond Mooney. Explanation-based learning: An alternative view. *Machine Learning*, 1(2):145–176, 1986.

[23] T.G. Dietterich. Ensemble methods in machine learning. In J. Kittler and F. Roli, editors, *First International Workshop on Multiple Classifier Systems*, Lecture Notes in Computer Science, pages 1 – 15. Springer Verlag, 2000.

[24] Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

[25] E. H. Durfee and V. R. Lesser. Negotiating task decomposition and allocation using partial global planning. In Les Gasser and Michael N. Huhns, editors, *Distributed Artificial Intelligence*, volume II, pages 229 – 243. Morgan Kaufmann Publishers, 1989.

[26] E. H. Durfee and J. Rosenschein. Distributed problem solving and multiagent systems: Comparisons and examples. In M. Klein, editor, *Proceedings of the 13th International Workshop on DAI*, pages 94–104, Lake Quinalt, WA, USA, 1994.

[27] M. Esteva, J. A. Rodriguez-Aguilar, C. Sierra, P.Garcia, and J. L. Arcos. On the formal specification of electronic institutions. In *Agent Mediated Electronic Commerce*, volume 1991 of *LNAI*. Springer-Verlag, 2001.

[28] Marc Esteva, Julian Padget, and Carles Sierra. Formalising a language for institutions and norms. In *Intelligent Agents VIII, Proceedings ATAL'01*, LNAI. Springer Verlag, To appear.

[29] Dayne Freitag. Multistrategy learning for information extraction. In *Proc. 15th International Conf. on Machine Learning*, pages 161–169. Morgan Kaufmann, San Francisco, CA, 1998.

[30] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *European Conference on Computational Learning Theory*, pages 23–37, 1995.

[31] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *Proc. 13th ICML*, pages 148–156. Morgan Kaufmann, 1996.

[32] Johannes Fürnkranz. Pairwise classification as an ensemble technique. In *Proceedings of the 13th European Conference on Machine Learning, ECML'2002*, volume 2430 of *LNAI*, pages 97–110. Springer Verlag, 2002.

[33] P. Cunningham G. Zenobi. Using diversity in preparing ensemble of classifiers based on different subsets to minimize generalization error. In *12th European Conference on Machine Learning*, 2001.

[34] João Gama and Pavel Brazdil. Cascade generalization. *Machine Learning*, 41(3):315–343, 2000.

[35] Ashok K. Goel, Andrs Gmez de Silva Garza, Nathalie Gru, J. William Murdock, and Margaret M. Recker. Functional explanations in design. In *IJCAI-97 Workshop on Modeling and Reasoning about Function*, 1997.

[36] L. K. Hansen and P. Salamon. Neural networks ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:993–1001, 1990.

[37] P. Hart. The condensed nearest neighbor rule. *IEEE Transactions on Information Theory*, 14:515–516, 1967.

[38] Thomas Haynes, Kit Lau, and Sandip Sen. Learning cases to compliment rules for conflict resolution in multiagent systems. In Sandip Sen, editor, *AAAI Symposium on Adaptation, Co-evolution and Learning in Multiagent Systems*, pages 51–56, 1998.

[39] Thomas Haynes and Sandip Sen. Evolving behavioral strategies in predators and prey. In Sandip Sen, editor, *IJCAI-95 Workshop on Adaptation and Learning in Multiagent Systems*, pages 32–37, Montreal, Quebec, Canada, 20-25 1995. Morgan Kaufmann.

[40] Thomas Haynes, Roger Wainwright, Sandip Sen, and Dale Schoenefeld. Strongly typed genetic programming in evolving cooperation strategies. In L. Eshelman, editor, *Genetic Algorithms: Proceedings of the Sixth International Conference (ICGA95)*, pages 271–278, Pittsburgh, PA, USA, 15-19 1995. Morgan Kaufmann.

[41] Tin Kam Ho. Adaptive coordination of multiple classifiers. In J.J. Hull and S.L. Taylor, editors, *Document Analysis Systems II*, pages 371–384. World Scientific Publishing Co., 1997.

[42] Junling Hu and Michael P. Wellman. Multiagent reinforcement learning: theoretical framework and an algorithm. In *Proc. 15th International Conf. on Machine Learning*, pages 242–250. Morgan Kaufmann, San Francisco, CA, 1998.

[43] N. R. Jennings. Commitments and conventions: The foundation of coordination in multi-agent systems. *The Knowledge Engineering Review*, 8(3):223–250, 1993.

[44] L. Karsenty and P. Brzillon. Cooperative problem solving and explanation, 1994.

[45] Leen kiat Soh and Juan Luo. Combining individual and cooperative learning for multiagent negotiations. In *Int. Conf. Autonomous Agents and Multiagent Systems AAMAS'03*, pages 686–692, 2003.

[46] Ron Kohavi and David H. Wolpert. Bias plus variance decomposition for zero-one loss functions. In Lorenza Saitta, editor, *Machine Learning: Proceedings of the Thirteenth International Conference*, pages 275–283. Morgan Kaufmann, 1996.

[47] V. Koltchinskii, D. Panchenko, and F. Lozano. Some new bounds on the generlization error of combined classiers. In T.Dietterich, editor, *Advances in Neural Information Processing Systems*, volume 14. MIT Press., 2001.

[48] M. Koppel and S. Engelson. Integrating multiple classifiers by finding their areas of expertise. In *AAAI-96 Workshop On Integrating Multiple Learned Models*, 1996.

[49] Anders Krogh and Jesper Vedelsby. Neural network ensembles, cross validation, and active learning. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 231–238. The MIT Press, 1995.

[50] D. Leake and R. Sooriamurthi. Automatically selecting strategies for multi-case-base reasoning. In S. Craw and A. Preece, editors, *Advances in Case-Based Reasoning: Proceedings of the Fifth European Conference on Case-Based Reasoning*, pages 204–219, Berlin, 2002. Springer Verlag.

[51] David B. Leake and Raja Sooriamurthi. When two case bases are better than one: Exploiting multiple case bases. In *ICCBR*, pages 321–335, 2001.

[52] David B. Leake and Raja Sooriamurthi. Managing multiple case bases: Dimensions and issues. In *Proceedings of the Fifteenth International Florida Artificial Intelligence Research Society (FLAIRS)*, pages 106–110. AAAI Press, 2002.

[53] David B. Leake and Raja Sooriamurthi. Dispatching cases versus merging case-bases: When mcbr matters. In *Proceedings of the Sixteenth International Florida Artificial Intelligence Research Society Conference (FLAIRS)*, pages 129–133. AAAI Press, 2003.

[54] David B. Leake and David C. Wilson. Remembering why to remember: Performance-guided case-base maintenance. In *EWCBR*, pages 161–172, 2000.

[55] Michael Lindenbaum, Shaul Markovitch, and Dmitry Rusakov. Selective sampling for nearest neighbor classifiers. In *AAAI/IAAI*, pages 366–371, 1999.

[56] Michael L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the 11th International Conference on Machine Learning (ML-94)*, pages 157–163, New Brunswick, NJ, 1994. Morgan Kaufmann.

[57] Elisabet Golobardes Maria Salamó. Hybrid deletion policies for case base maintenance. In *FLAIRS'2003*, pages 1150–155, 2003.

[58] Ofer Matan. On voting ensembles of classifiers (extended abstract). In *AAAI 96 - Workshop in Induction of Multiple Learning Models*, 1996.

[59] M. Mataric. Learning to behave socially, 1994.

[60] N. Matos, C. Sierra, and N. R. Jennings. Determining successful negotiation strategies: an evolutionary approach. In Y. Demazeau, editor, *Proceedings of the 3rd International Conference on Multi-Agent Systems (ICMAS-98)*, pages 182–189, Paris, France, 1998. IEEE Press.

[61] Lorraine McGinty and Barry smyth. Collaborative case-based reasoning: Applications in personalized route planning. In *ICCBR*, pages 362–376, 2001.

[62] E. McKenna and B. Smyth. Competence models and the maintenance problem. *Computational Intelligence: Special Issue on Maintaining Case-Based Reasoning Systems*, 17(2):235–249, 2001.

[63] Tom Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[64] Pragnesh Jay Modi and Wei-Min Shen. Collaborative multiagent learning for classification tasks. In Jörg P. Müller, Elisabeth Andre, Sandip Sen, and Claude Frasson, editors, *Proceedings of the Fifth International Conference on Autonomous Agents*, pages 37–38, Montreal, Canada, 2001. ACM Press.

[65] John F. Nash. Non-cooperative games. *Annals of Mathematics*, 2(54):286–295, 1951.

[66] Santi Ontañón and Enric Plaza. Learning when to collaborate among learning agents. In *12th European Conference on Machine Learning*, pages 394–405, 2001.

[67] M. P. Perrone and L. N. Cooper. When networks disagree: Ensemble methods for hydrid neural networks. In *Artificial Neural Networks for Speech and Vision*. Chapman-Hall, 1993.

[68] E. Plaza, J. L. Arcos, P. Noriega, and C. Sierra. Competing agents in agent-mediated institutions. *Journal of Personal Technologies*, 2:212–220, 1998.

[69] Enric Plaza, Josep Lluís Arcos, and Francisco Martín. Cooperative case-based reasoning. In Gerhard Weiss, editor, *Distributed Artificial Intelligence Meets Machine Learning. Learning in Multi-Agent Environments*, number 1221 in Lecture Notes in Artificial Intelligence, pages 180–201. Springer-Verlag, 1997.

[70] Enric Plaza and Santiago Ontañón. Ensemble case-based reasoning: Collaboration policies for multiagent cooperative cbr. In I. Watson and Q. Yang, editors, *In Case-Based Reasoning Research and Development: ICCBR-2001*, number 2080 in LNAI, pages 437–451. Springer-Verlag, 2001.

[71] M V Nagendra Prassad, Victor R Lesser, and Susan Lander. Retrieval and reasoning in distributed case bases. Technical report, UMass Computer Science Department, 1995.

[72] Christopher D. Rosin. *Coevolutionary search among adversaries*. PhD thesis, San Diego, CA, 1997.

[73] T. W. Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135:1–54, 2002.

[74] Robert E. Schapire and Yoram Singer. BoosTexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168, 2000.

[75] Searle. *Speech Acts: An Essay on the Philisophy of Language*. Cambridge University Press, 1969.

[76] H. S. Seung, Manfred Opper, and Haim Sompolinsky. Query by committee. In *Computational Learing Theory*, pages 287–294, 1992.

[77] B. Smyth. The utility problem analysed: A case-based reasoning persepctive. In *Third European Workshop on Case-Based Reasoning EWCBR-96*, Lecture Notes in Artificial Intelligence, pages 234–248. Springer Verlag, 1996.

[78] Barry Smyth and Mark T. Keane. Remenbering to forget: A competence-preserving case delection policy for case-based reasoning systems. In *Proceedings of IJCAI-95*, pages 377–382, 1995.

[79] Barry Smyth and Elizabeth McKenna. Building compact competent case-bases. *Lecture Notes in Computer Science*, 1650:329–??, 1999.

[80] Luc Steels. Emergent functionality in robotic agents through on-line evolution. pages 8–16.

[81] Peter Stone and Manuela M. Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, 2000.

[82] Nobuo Suematsu and Akira Hayashi. A multiagent reinforcement learning algorithm using extended optimal response. In *1st International Joint Conference in Autonomous Agents and Multiagent Systems*, pages 370–377, 2002.

[83] Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative learning. In *Proc. 10th International Conf. on Machine Learning*, pages 330–337. Morgan Kaufmann, San Francisco, CA, 1993.

[84] K. Ting. The characterisation of predictive accuracy and decision combination. In *Thirteenth International Conference on Machine Learning*, pages 498–506. Morgan Kaufmann, 1996.

[85] K. Tumer and J. Ghosh. Classifier combining: analytical results and implications. In *AAAI 96 - Workshop in Induction of Multiple Learning Models*, 1996.

[86] Kagan Tumer, Adrian K. Agogino, and David H. Wolpert. Learning sequences of actions in collectives of autonomous agents. In *1st International Joint Conference in Autonomous Agents and Multiagent Systems*, pages 378–385, 2002.

[87] D. H. Wolpert. Stacked generalization. Technical Report LA-UR-90-3460, Los Alamos, NM, 1990.

[88] Bruce A. Wooley. Explanation component of software systems. *ACM CrossRoads*, 1998.

[89] Michael Woolridge. *The logical modelling of computational multi-agent systems*. PhD thesis, University of Manchester, U.K., 1992.

[90] Jun Zhu and Qiang Yang. Remembering to add: Competence-preserving case-addition policies for case base maintenance. In *IJCAI*, pages 234–241, 1999.