Aplicación de algoritmos y métodos de IA sobre un juego real en Internet

Sergio Daniel Fernández Atochero

Resumen—En este proyecto se ha llevado a cabo una implementación funcional de un videojuego online de estrategia con dificultad media para un mínimo de 2 jugadores y un máximo de 4, en donde, como máximo, puede haber un jugador humano, aunque no es necesario que lo haya. El proyecto se compone de dos elementos que forman la Inteligencia Artificial, por un lado en el cliente, con una IA Local que sido dotada de una pequeña autonomía y por otro lado, en el servidor, a través, de técnicas de aprendizaje como un SVM y una red neuronal artificial que permiten que se aprenda de las estrategias llevadas a cabo en el transcurso de una partida ganadora. El proyecto ha sido creado en su totalidad con HTML5 y con JavaScript, tanto del lado del cliente como en el lado del servidor, para poder ser compatible con multitud de dispositivos y Sistemas Operativos.

Palabras clave—IA, algoritmo, juego, videojuego, técnica, estrategia, HTML5, Node.js, Kinetic.js, socket.io, estrella, mochila, estrategia, online, aprendizaje, SVM, red, neuronal, artificial.

Abstract— In this project has conducted a working implementation of an online strategy game with medium difficulty for a minimum of 2 and a maximum of 4 players, wherein a maximum may be a human player, but need not have there. The project consists of two elements forming the Artificial Intelligence, on the one hand, a Local IA, It was provided with a small range of autonomy and secondly, on the server, though, learning techniques as SVM and an artificial neural network that allows learning from the strategies pursued during a winning game. The project has been created entirely with HTML5 and JavaScript, both client side and server side, in order to be compatible with multiple devices and operating systems.

Index Terms— IA, algorithm, gaming, video game, art, strategy, HTML5, Node.js, Kinetic.js, socket.io, star, backpack, strategy, online, learning, SVM, network, neural, artificial.

1 Introducción

Este articulo trata de describir el Proyecto Final de Grado, desgranando en su totalidad el desarrollo del proyecto, su funcionalidad, elementos que lo componen, la metodología utilizada en su desarrollo y los resultados obtenidos .

Aunque el objetivo final del proyecto no es en sí solo el desarrollo del videojuego, merece la pena hacer una breve descripción introductoria de cómo funciona el juego, pues posteriormente se harán muchas referencias a ello.

El videojuego es un juego de estrategia por turnos con un máximo de 4 jugadores y un mínimo de 2, de los cuales uno puede ser humano o no, y donde se pueden comprar objetos y desarrollar tecnologías para intentar llegar a una de las guaridas (posición de salida) de los otros jugadores, situadas en las esquinas del mapa y así ganar la partida; para poder llegar a la situación anteriormente descrita de victoria, se han de atravesar las zonas de recursos, las de agua, así como las defensas, además de evitar los ataques del enemigo que trataran de impedir nuestro objetivo, impidiendo la ejecución de nuestro turno o turnos, también comentar que tal y como se ha planteado el juego, el jugador debe regresar a su guarida para dejar los recursos que ha ido recolectando, y así liberar el espacio de la mochila que le permita seguir recogiendo recursos y por tanto avanzando. Pero el proyecto es mucho más amplio, dándose una visión general del funcionamiento básico en la *Figura 1*, este funcionamiento será el que describamos en los apartados posteriores de este artículo.

- *E-mail de contacte: SergioDaniel.Fernandez@e-campus.uab.cat*
- *Mención realizada: Computación*
- *Trabajo tutorizado por Ramón Baldrich (CVC)*
- *Curso 2013/14*

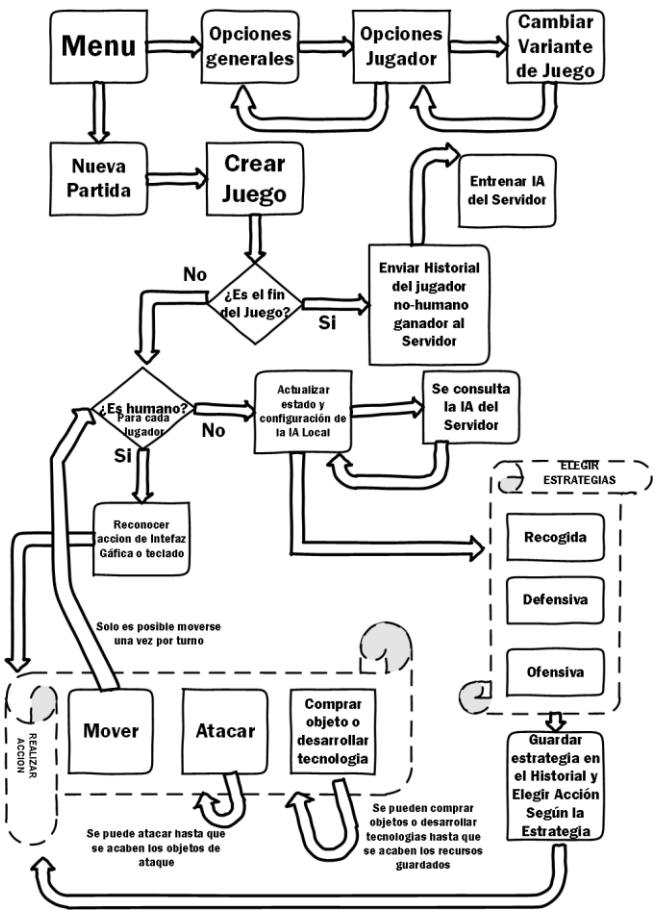


Figura 1

1.1 Inspiración

Si hacemos un repaso a la evolución de la Inteligencia Artificial en los videojuegos, desde los primeros sistemas de IA (años 50) que se aplicaron a juegos de mesa, como las damas(Arthur Samuel) y el ajedrez(Claude Shannon), pasando por los años 60 con juegos como el Pong o Spacewar basados en la lógica, y por los años 70 donde se crearon juegos de 1 jugador que luchaban contra los enemigos, los cuales se movían mediante patrones almacenados, como el juego Space Invaders (1978) que añadió la dificultad creciente y respondía a las acciones del jugador. Más tarde se empezaron a aplicar técnicas de IA más complejas como el Pac-Man (1980) que incorporó algoritmos de búsqueda en laberintos. La evolución prosiguió en los 90 donde se produjo un boom de nuevos géneros y nuevas técnicas de IA (Máquinas de estados finitos, Redes neuronales, Computación evolutiva, Lógica difusa, etc...), por ejemplo Dragon Warrior (1990) fue el primer RPG y permitía variar las rutinas de la IA de los enemigos durante las batallas o Battelcruiser 3000AD (1996) que incorporaba redes neuronales. Más actualmente comentar BotPrize[1] que es un concurso sobre el mundo de “Unreal Tournament 2004, que intenta a través del Test de Turing comprobar la “humanidad” de los bots (jugadores no humanos) que participan.

El avance de las tecnologías web han abierto ante nosotros un mundo de posibilidades que permiten incorporar algoritmos y técnicas de aprendizaje computacional, a la vez que utilizar la web para recopilar información, pero hasta ahora los usos de esta combinación ha calado poco en el conjunto de Internet, hay bastantes ejemplos de esta simbiosis, pero todas o la inmensa mayoría de ellas están dirigidas en gran medida a los sistemas de recomendación, como por ejemplo en las e-commerce como Amazon, donde se recogen tus preferencias para ofrecerte un producto concreto, pero no he encontrado referencias a juegos online que modifiquen su comportamiento a través del aprendizaje de modo online, en todo caso se recogen datos del usuario online y posteriormente se utilizan para entrenar nuevas versiones del juego en offline, pero en ningún caso el juego online evoluciona sin cambiar la versión.

1.1 Estado Del Arte

Nuestro objetivo es conseguir algo diferente a lo anteriormente explicado, básicamente consiste en la realización de un juego de estrategia con varios jugadores, variabilidad de opciones y una complejidad media, para utilizar en él algoritmos y técnicas que le permitan que el juego aprenda, y con este fin, poder utilizar ese conocimiento adquirido como referencia del propio juego, es decir un retro-aprendizaje, pero todo ello sin tener que sacar una nueva versión o aplicar un parche al juego, sino que dicho comportamiento ya vendría incluido en la versión final.

Este proyecto busca conseguir que los dos elementos que conforman nuestra Inteligencia Artificial se comuniquen y aconsejen. Por ejemplo la Inteligencia Artificial Local, que es aquella que utiliza el juego será capaz de consultar a la Inteligencia Artificial No Local, localizada en el servidor y que integra una Red Neuronal y un SVM. Pero a su vez la Inteligencia Artificial No Local se retroalimentara de la información almacenada en la Inteligencia Artificial Local cuando algún jugador (humano o no) consigue ganar una partida.

Este principio se podría utilizar con el fin de modificar la dificultad del juego de forma dinámica, permitiendo que se vaya reentrenando para que cada partida sea más compleja, y así poder ofrecer al usuario una experiencia de juego donde la dificultad sea progresiva.

1.2 Motivaciones Personales

La idea inicial de este proyecto surgió tras la realizar la Mención de Computación en el Grado de Ingeniería Informática de la UAB (Universidad Autónoma de Barcelona), pues en el curse muchas asignaturas orientadas al desarrollo de algoritmos como MIN&MAX, Algoritmo de Estrella, Algoritmo de la Mochila, SVM, ID3, Redes Neuronales, Redes Bayesianas, etc. Pero todas estas técnicas y/o algoritmos se llevaban a cabo en situaciones muy específicas e independientes unas de otras, por ello pensé en realizar un proyecto que integrara todo lo aprendido

y donde combinar muchas de ellas.

También como motivación personal pretendía utilizar tecnologías innovadoras como es el caso de Node.js y con la realización del proyecto asentar la experiencia en estas nuevas tecnologías aplicándolas a un proyecto real.

Por último como motivación personal también indicar que la resolución del proyecto es un gran paso para la finalización del grado, otro gran factor de motivación.

2 METODOLOGÍA

Para la realización del proyecto no me quedo más opción que aplicar una metodología de trabajo iterativa, pues el proyecto está conformado únicamente por una sola persona, por ello se ha llevado a cabo una división en 3 paquetes de trabajo independientes, con el fin de poder hacer entregas funcionales (Hitos) de parte del código, y en el caso de correr algún riesgo tener la posibilidad de abandonar el proyecto dejando alguna parte totalmente funcional y entregada al cliente.

A su vez cada paquete identifica una fase, que finaliza en un Hito, las fases y subfases se describen a continuación, pudiendo llevarse a cabo en paralelo en un equipo compuesto de varias personas.

2.1 Primera Fase. Creación de un videojuego

Como primera fase se desarrollo la parte visual del juego y la lógica básica para los movimientos, sin llegar aún a entrar en la IA Local. En este punto se describe el proceso seguido y una breve explicación de la razones para llevarlo a cabo:

Comenzar comentando que uno de los objetivos básicos que impone el proyecto, como es la utilización de tecnologías web para la ejecución del juego, que es necesaria para conseguir una recopilación de información exitosa, me llevo a considerar diversas opciones para el desarrollo del videojuego, este dato es significativo ya que después de evaluar dichas opciones opte por desarrollar el juego íntegramente en HTML5 y JavaScript, y este factor condiciono totalmente el desarrollo de las partes posteriores.

Seguidamente se planifico cuidadosamente los diferentes componentes del juego y la forma de aplicar patrones de diseño para conseguir una correcta interdependencia entre los diversos componentes de esta primera fase, se optó por el desarrollo del proyecto con el framework MelonJS, el cual permite de un modo sencillo la construcción de videojuegos en 2D visualmente atractivos. Pero tras comenzar el desarrollo, el cliente (profesor) incluyo un nuevo requisito que es totalmente lógico y que entraba en conflicto con el desarrollo llevado a cabo hasta el momento. Este requisito era la generación de mapas con los siguientes parámetros totalmente aleatorios:

-Tamaño de mapa.
-Generación de ríos y lagos (posición y número).
-Generación de recursos.

El requisito era lógico como he mencionado anteriormente, pues el framework que se utilizaba estaba basado en pantallas predefinidas creadas con “tiles”(pequeños elementos gráficos que se combinan en otros de mayor tamaño) que permitirían que la futura IA No-Local se adaptara a dichas pantallas, con el cambio propuesto, al hacerlo totalmente aleatorio la IA No-Local no puede saber a qué se enfrentara, por ejemplo, si el mapa tiene un tamaño fijo, y los recursos están siempre en el mismo lugar, la IA conocerá tras varias ejecuciones la localización de estos recursos y de los límites del mapa, por tanto podrá optimizar las acciones de la IA para que con el mínimo posible de movimientos pueda ganar la partida, dando consecuencia un nivel demasiado elevado para un jugador humano. Este hecho derivo en una nueva búsqueda de un framework que se adaptara mejor para el cumplimiento del requisito y se empezó a utilizar KineticJS, un framework de más bajo nivel para la creación de interfaces HTML5 con canvas, teniendo que desechar la calidad gráfica prevista al principio por una funcionalidad más específica.

Tras este impedimento seguí desarrollando de forma iterativa el resto de componentes en diferentes subfases. Las cuales resumo a continuación:

2.1.1 Crear elementos de la interfaz

Se creó una interfaz donde se muestran los componentes que el usuario y la IA necesitan recibir como información gráfica e interna para el correcto desempeño del juego, por ello se crearon diferentes apartados como la mochila, que contiene los objetos recogidos, una panel de todos los recursos que disponemos, una ventana de turnos inhabilitados, otro panel de información general, además de unos botones que permiten cambiar la información del panel de información general, para poder comprar objetos y/o tecnologías y por último el mapa.

Como se ha comentado anteriormente la mejora de la apariencia gráfica tuvo que dejarse a un lado, pues había un conflicto con los objetivos primordiales y al tenerse que realizar todos los componentes desde 0, el tiempo para la realización de esta parte se salía ampliamente del disponible para la realización del TFG.

2.1.2 Crear movimientos y restricciones del jugador

En esta subfase se definieron los movimientos que el jugador puede realizar y se crearon las restricciones con los diferentes elementos y límites del terreno, necesarias para impedir que el jugador pudiese pasar por el agua, o través de los recursos sin haber realizado la acción de recogerlos previamente, también se definió el mecanismo de inhabilitación, con el fin de bloquear el turno del juga-

dor, que básicamente consiste en invalidar dicho turno, hasta que el resto de jugadores realice su turno, con este procedimiento incluimos un factor de penalización que será utilizado para marcar la diferencia entre los diferentes jugadores, pues el objetivo final del juego consiste en llegar a la base de un jugador enemigo en el menor número de turnos posible y antes que el resto de enemigos.

2.1.3 Crear la lógica básica del juego

Cuando se creó la lógica básica del juego, lo que se hizo fue desarrollar e implementar la lógica básica para que un jugador pudiera comprar objetos para así, ser ofensiva o defensivamente superior a sus oponentes y desarrollar tecnologías, que permiten acelerar la recogida de recursos, una vez realizado este proceso, se amplió la lógica para que pudieran jugar varios jugadores por turnos.

Posteriormente se impuso otro requisito extra por parte del cliente, consistente en implementar más variantes del juego, y así lo desarrolle, fueron las siguientes:

- **Variante de Visibilidad:**
  Permite elegir entre una visibilidad total o parcial, pudiendo ver solo una pequeña parte del mapa como si fuera de noche y el jugador llevara una linterna, o ver el terreno totalmente descubierto.

- **Variante de Memoria:**
  Esta opción permite activar o desactivar la regeneración de la niebla, representada con nubes en el juego, en las zonas que hemos ido pasando.

- **Variante de Regeneración de Recursos:**
  Esta opción permite que en las zonas de niebla se vuelvan a regenerar los recursos de forma aleatoria, pero se comprobó que suponía un gran impedimento en la ejecución correcta del juego, se optó por su eliminación.

2.1.4 Mejorar el rendimiento

Además de la problemática encontrada con la apariencia gráfica, se encontró un gran problema de rendimiento, pues se generaban mapas de gran tamaño, el problema en si era debido a que cada vez que se realizaba un movimiento por parte del jugador humano, era necesario refrescar la pantalla y como se puede comprobar (*Figura 2*), la mayoría de las veces eran demasiados elementos para mostrar, por lo que de nuevo se tuvo que replantear la interfaz, con el fin de realizar una mejora drástica en su rendimiento.

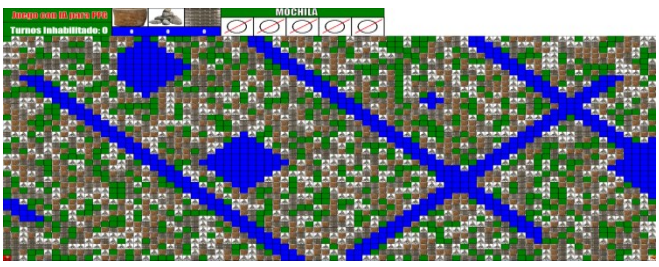


Figura 2

Para ello se separó en dos bloques dicha interfaz y se representó el mapa en la parte izquierda, como se muestra en la *Figura 3*.

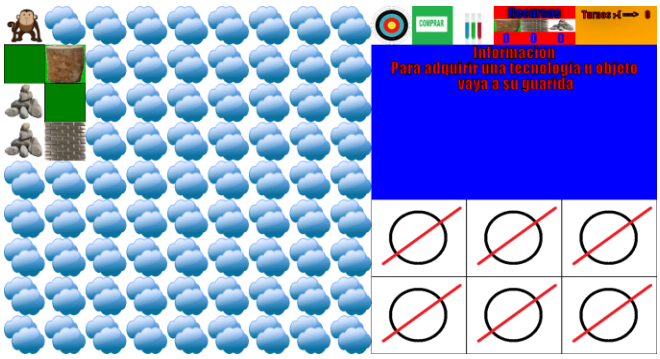


Figura 3

Como se puede apreciar se redujo la ventana que muestra el mapa a un tamaño fijo de 9x9 casillas, permitiendo tener una submatriz del mapa original, el cual es de tamaño nxm, mostrando esta submatriz del mapa se pudo solucionar uno de los principales problemas de rendimiento de esta fase.

2.2 Segunda Fase. Creación de una IA Local

Esta segunda fase fue la más compleja de plantear y desarrollar, en ella se implementó la autonomía de los jugadores no humanos(IA Local), que utilizan la información recopilada por el propio juego para elegir entre las tres estrategias básicas llamadas Recogida, Defensiva y Ofensiva que se han sido definidas, las cuales utilizan un algoritmo de Backtracking(concretamente la configuración del problema de la mochila) para comprar objetos y el algoritmo de estrella para llevar a cabo los movimientos que mejor se adapten a la estrategia elegida, a continuación se hace una breve descripción de estos algoritmos y estrategias.

2.2.1 Métricas estandarizadas

Antes de comenzar a definir los diferentes algoritmos y estrategias, se necesitaba una información que pudiera ser evaluable independientemente de la configuración inicial del juego (mapa, variantes, etc…), y que las estrategias pudieran tomar como parámetros de entrada, para ello se definieron diferentes porcentajes que estandarizan la información del juego, para ejemplificar este punto, a continuación expongo un ejemplo:

- **Ejemplo:** Para un mapa de 320 casillas de anchura, la casilla 20, correspondería al 6.25% de la anchura ((100/320)*20=6.25).

Para ello definimos diferentes porcentajes que estandarizan la información del juego, pero estos porcentajes a su vez se basan en otros dos porcentajes básicos llamados porcentaje de avance ofensivo y defensivo, cuyo código a modo de ejemplo se incluye en la *sección A1 del Apéndice*, estos porcentajes son los siguientes:

- **Porcentaje de Avance Ofensivo (PAO):**
Este porcentaje calcula lo cerca que se está de una de las posibles guaridas (posición de inicio) enemigas, incrementando el porcentaje según nos acercamos.

- **Porcentaje de Avance Defensivo (PAD):**
Este porcentaje calcula lo lejos que se está de la guarida del jugador que ejecuta su turno.

Los porcentajes utilizados por las estrategias y que a su vez utilizan los porcentajes anteriormente descritos son los siguientes:

- **Porcentaje Ofensivo (PO):**
Es el resultado de multiplicar el Porcentaje de Avance Ofensivo (PAO) de la casilla que el jugador ocupa por 0.5, pues supone el 50% del peso total de este porcentaje y luego se le suma un porcentaje calculado como el porcentaje de objetos que ese jugador posee de la totalidad de objetos ofensivos disponibles para comprar, llamado Porcentaje de Objetos Ofensivos (POO) * 0.5 al igual que el parámetro anterior.

PO= (PAO*0.5) + (POO*0.5)

- **Porcentaje Defensivo (PD):**
Es el resultado de multiplicar el Porcentaje de Avance Defensivo (PAD) de la casilla que el jugador ocupa por 0.5, pues supone el 50% del peso total de este porcentaje y luego se le suma un porcentaje calculado en base del número de objetos defensivos colocados por el jugador entre el número total de turnos y todo ello multiplicado por 0.5 como el parámetro anterior.

PD= (PAD*0.5)+ ((Nº de Objetos Defensivos colocados por el jugador / Nº de turnos) * 0.5)

- **Porcentaje de Éxito (PE):**
Depende del Porcentaje Defensivo (PD), ya que si tenemos el problema de la defensa solucionado, es más fácil que podamos ganar y del Porcentaje Ofensivo (PO) pues nos da una perspectiva de lo cerca que podemos estar de una base enemiga y de cómo estamos preparados para realizar la incursión. Cabe destacar que el PAD y PAO se contrarrestan el uno al otro, porque dependen de la posición en el mapa y permiten que no se dé nunca el 100% en el porcentaje de Éxito, ya que es algo que no podemos saber, se calcula como sigue:

PE = (PD+PO)/2

- **Porcentaje de Peligro (PP):**
Depende del número de avistamientos, que es el número de veces que vemos en nuestro campo de visión a un jugador enemigo, también de una variable donde almacenamos la menor distancia con respecto a la base donde hemos visto a un enemigo, que corresponde al valor PAD de la casilla que ocupa el jugador enemigo y por ultimo del número de turnos, se calcula de la siguiente forma:

PP= (Nº de Avistamientos/Nº de turnos) +PAD

2.2.2 Estrategias

Los porcentajes anteriormente citados son tomados como base a la hora de elegir la estrategia que se llevara a cabo en ese turno, comentar que esa estrategia no permanece en el tiempo, si no que es modificada en cada nuevo turno del jugador respondiendo a la situación actual de los porcentajes, también añadir que no se han podido implementar todas las características necesarias para que la IA Local funcione con total normalidad y de una forma totalmente coherente, para llegar a ese objetivo se hubiese necesitado muchísimo más tiempo que el que teníamos disponible para la realización de este proyecto, por lo que se tienen tres tipos de estrategias diferentes muy simples que imponen diferentes comportamientos a la IA Local (jugador no humano), pero que en ningún momento se puede decir que la IA Local tenga un comportamiento realista, comentar además, que para simplificar el ataque las estrategias descargan todo su potencial ofensivo sobre aquellos enemigos que tienen en su campo de visión, los cuales son captados en dicho campo y almacenados en un array junto con la distancia del jugador que posee el turno, posteriormente se recorre el array intentando optimizar los ataques, es decir, empiezan con aquellos objetos que permiten un ataque desde una distancia mayor al jugador o jugadores enemigos más distantes, se pasa al siguiente objeto y se ataca al siguiente jugador que este en el campo de acción de dicho objeto y así hasta terminar con la lista de objetos de la mochila, es decir, en un turno se permite hacer un ataque con cada objeto ofensivo de la mochila.

Las estrategias implementadas son las siguientes:

- **Recogida:**
Esta estrategia es la que se tiene para iniciar el juego, ya que para empezar hay que recoger recursos, que posteriormente utilizaremos en la compra de objetos y tecnologías, para ello a la hora de configurar el mapa se marcan unas zonas de seguridad, con el fin de asegurar que el jugador tiene los recursos necesarios para empezar como muestra la *Figura 4*, esta estrategia es la predefinida hasta que al menos tengamos 10 recursos de cada tipo, y por consiguiente no hacemos uso del algoritmo que rellena la mochila, pues nuestro interés en mantener el máximo posible de espacio libre en la mochila y así poder

recolectar el máximo de recursos. También se reactivara la estrategia sino tenemos al menos los 10 recursos de cada tipo, si esta situación se cumple se pasara de esta estrategia a evaluar si se cumplen las condiciones para la estrategia Defensiva.

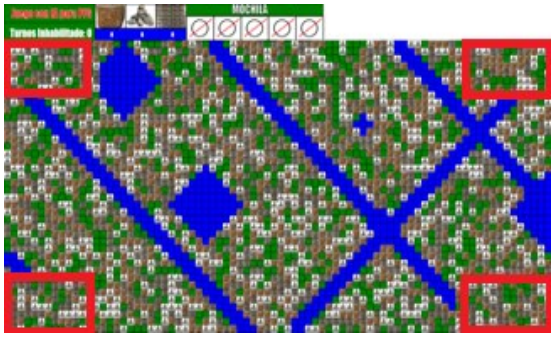


Figura 4

Las siguientes estrategias rellenan solo la mochila hasta la mitad, pues tanto para las estrategias de defensa y ataque se necesitan espacios libres en la mochila con los que ir recogiendo recursos, si no dispusiéramos de estos espacios libres, no podríamos avanzar y por tanto estaríamos bloqueados.

Como se ha comentado anteriormente estas dos estrategias no están correctamente implementadas y este hecho desencadena en un comportamiento errático en las acciones de la IA Local, por lo que, aunque aquí se describa su comportamiento actual, no quiere decir que sea el comportamiento deseado para finalizar el proyecto satisfactoriamente, este apunte se amplía pertinentemente en las dos estrategias que definimos a continuación:

- **Defensiva:**
  Si la puntuación defensiva está por debajo del 65% o la puntuación de peligro es mayor de un 65%, se activa la estrategia, comprando objetos defensivos, como trampas y muros, el problema es que para la colocación correcta de muros y puertas, es necesaria la implementación de unos algoritmos que permitan colocar dichos elementos haciendo recintos cerrados, o aprovechar las fronteras con el agua para impedir correctamente el avance del jugador enemigo, se definieron unos algoritmos para llevar a cabo esta función pero finalmente no se han podido realizar, a modo de información adicional se muestran uno de ellos en la *Sección A2 del Apéndice*, aunque posteriormente en Futuras Mejoras se volverá a hacer un apunte sobre ellos .

- **Ofensiva:**
  Esta estrategia es la que se lleva a cabo cuando no se cumplen las condiciones para la elección de las dos estrategias anteriores y busca ampliar el porcentaje PAO, buscando rutas que permitan recoger los recursos que estén más próximos a las guaridas enemigas, haciendo así que nos aproximemos mas a la base enemiga y que podamos ganar la partida.

Para un juego real lo lógico, sería no ser tan estrictos a la hora de realizar las acciones de la estrategia, y poder combinarlas, en Futuras Mejoras también se abordara este tema.

2.2.3 Historial

Cada vez que un jugador no humano realiza su turno los porcentajes y la estrategia usada se almacenan en un historial, que será utilizado posteriormente por la IA No-Local, esta variable tiene la siguiente estructura:

Historial = {{[PO], [PD], [PE], [PP], [Estrategia Usada]}$_1$,
……
{[PO], [PD], [PE], [PP], [Estrategia Usada]}$_n$}

2.2.4 Algoritmo de Backtracking y el problema de la mochila

Este punto hace referencia al uso del problema que se encontró a la hora de comprar los objetos y que se resolvió con el uso de prioridades para los objetos según su estrategia y el algoritmo de Backtracking, además hemos configurado el algoritmo para que lo pueda hacer en tamaños variables de la mochila, ya que no siempre nos interesara rellenarla en su totalidad, si no que a veces desearemos dejar espacios libres para recoger recursos, pues sino, no podremos avanzar.

El problema[2] se puede expresar matemáticamente como sigue (*Figura 5*):

$$\begin{aligned} \text{maximizar} \quad & \sum_{i=1}^{n} v_i x_i \\ \text{tal que} \quad & \sum_{i=1}^{n} w_i x_i \leq W \\ \text{y} \quad & 1 \leq q_i \leq \infty. \end{aligned}$$

Figura 5

En nuestro caso q_i está acotado a la capacidad de la mochila, aunque como se verá más tarde, este parámetro no es fijo y cada tipo de ítem i tiene un beneficio, designado con v_i, y el peso(w_i) será 1 para un solo objeto, pero como se tienen objetos repetibles, si se repite ese objeto en la mochila, el peso del objeto ira incrementando ya que se ha definido un orden de prioridad que depende de la estrategia que este seleccionada en ese momento, por ejemplo, si se está llevando a cabo una estrategia de-

fensiva, el beneficio de un muro será mayor que el de una catapulta, en la *Sección 3 del Apéndice* se muestra la tabla con las diferentes prioridades de las estrategias.
La resolución de este problema se lleva a cabo ejecutando un algoritmo de Backtracking que intenta optimizar la compra de objetos según los recursos disponibles y la estrategia designada, siguiendo un orden de prioridad, por lo que comienza con los objetos de mayor prioridad y va restando recursos hasta quedarse sin recursos o llenar los espacios que han sido especificados en la mochila, posteriormente se devuelve la lista de objetos que se deberían comprar, los cuales serán comprados por la IA Local.

2.2.5 Algoritmo de Estrella

En este apartado se realiza una breve explicación del algoritmo de Estrella y de aquellos aspectos del proyecto donde da soporte.

El algoritmo de estrella o A* utiliza una función de evaluación $f(n)=g(n)+ h'(n)$, donde $h'(n)$ representa el valor heurístico del nodo a evaluar desde el actual, n, hasta el final, y $g(n)$, el coste real del camino recorrido para llegar a dicho nodo, n, desde el nodo inicial. A* mantiene dos estructuras de datos auxiliares, que podemos denominar **abiertos**, implementado como una cola de prioridad(ordenada por el valor de $f(n)$ de cada nodo), y **cerrados** , donde se guarda la información de los nodos que ya han sido visitados. En cada paso del algoritmo, se expande el nodo que esté primero en abiertos y en caso de que no sea un nodo objetivo, calcula el valor de $f(n)$ de todos sus hijos, los inserta en abiertos y pasa el nodo evaluado a cerrados.

El algoritmo es una combinación entre búsquedas del tipo primero en anchura con primero en profundidad; mientras que $h'(n)$ tiende a primero en profundidad, $g(n)$ tiende a primero en anchura. De este modo, se cambia de camino de búsqueda cada vez que existen nodos más prometedores.

En el caso que nos ocupa, este algoritmo se usa tras evaluar que recursos hay en el campo de visión de la IA Local, para saber si se puede llegar o no al recurso objetivo, si no es posible devolverá una ruta vacía, en caso contrario retornara la ruta hasta dicho recurso, pudiéndose evaluar de los diversos recursos que hay en el campo de visión, cuál es el más próximo que tenga una ruta viable, y pasándosele dicha ruta a la IA Local para ir directamente hacia él, pero si se da el caso de que la mochila ya esté llena, se ha modificado el algoritmo para que nos retorne la ruta óptima(camino más corto evitando recursos) hacia la guarida, con el fin de descargar la mochila.

Además comentar que se ha utilizado una implementación de código totalmente libre y modificable para adaptarlo a nuestro proyecto[3].

2.3 Tercera Fase. Creacion de una IA No Local

Llegados a esta fase y tras crearse la aplicación del cliente(videojuego), se paso al desarrollo del servidor y a la elección de las técnicas que se iban a utilizar en el servidor donde se eligieron del conjunto de modulos disponibles del gestor de modulos(NPM) de Node.jS, el SVM, y la Red Neuronal aunque podria haber utilizado cualquier otra técnica como un ID3(pero necesitaba discretizar los datos) o un K-Means,que es muy rápido, pero opte por seleccionar el SVM y la Red Neuronal porque me gustaba la idea de volver a utilizarlos y correctamente parametrizados pueden llegar a ser técnicas de un gran potencial.

2.3.1 Servidor con Node.js y Express

Para la creación del servidor, se utilizó Node.JS por tres razones principalmente:

La primera por la sencillez a la hora de crear un servidor que se logra con solo 3 líneas de código, y por su gestor de paquetes que me ha permitido utilizar el módulo de SVM y de la Red Neuronal con licencias totalmente libres y que permiten modificar el código[4].

La segunda es el conocimiento previo del entorno y del lenguaje JavaScript que el que utiliza, aunque el conocimiento al principio del proyecto era limitado, me ha permitido realizar todo lo que deseaba en un tiempo muy inferior que si hubiera utilizado PHP, que es el hemos utilizado en el grado, o alguno de los otros lenguajes de servidor disponibles.
La tercera es personal, me gusta cómo funciona Node.js y las posibilidades que ofrece al ser totalmente asíncrono, no como en otros lenguajes donde es necesario Ajax para simular esta funcionalidad, además de su extrema rapidez, también destacar que a primera vista puede parecer que tiene incompatibilidades con el proyecto, pues solo ejecuta un hilo de ejecución por servidor, en aplicaciones con gran cantidad de cálculo como es nuestro caso, puede suponer un problema, pues las peticiones de los clientes se han de quedar esperando en cola mucho tiempo hasta que se complete en entrenamiento de la red neuronal o el SVM(en nuestro caso el entrenamiento se hace cada 24 horas), pero este problema aunque no se ha llegado a implementar en el proyecto por falta de tiempo, sí que había sido contemplado y se había buscado como solucionarlo, principalmente a través de dos estrategias:

> **Hilos de ejecución:**
> Crear un nuevo hilo de ejecución para la red neuronal o SVM, pues no interferiría en las peticiones de los usuarios, y sabríamos que ha terminado, ya que todas las funciones en Node.js llevan un parámetro de callback que permite ejecutar una función cuando finalice su ejecución.

NextTick:
NextTick (), es una función que permite dividir un código en partes, en nuestro caso lo utilizaríamos para ir ejecutando el entrenamiento por rangos, por ejemplo si hacemos 100 iteraciones podríamos dividir en bloques de 5 iteraciones con lo que ni el servidor ni el cliente tendrían constancia de ello ya que distribuiríamos la carga de computo a través del tiempo.

El siguiente punto de este apartado es Express que es un framework para Node.js robusto, rápido, flexible y simple que nos permite implementar de forma muy sencilla aspectos de seguridad, enrutamiento y sesiones.

2.3.2 Websockets con SOCKET.IO

Socket.IO tiene como objetivo hacer posible aplicaciones en tiempo real en todos los navegadores y dispositivos móviles, borrando las diferencias entre los distintos mecanismos de transporte. Además es 100% JavaScript, con ello al enviar datos al servidor no tengo que realizar ningún modificación en la estructura de los datos que envío, pues puedo realizar el envió de variables JavaScript directamente, enviándose en formato JSON y siendo recibidos por el servidor Node.js que también está escrito en el JavaScript. Así es como envío el historial de la IA Local al Servidor, donde será entrenado por la parte de IA No-Local (SVM y Red Neuronal), también lo utilizo para la parte de consulta como se verá en un apartado posterior.

2.3.3 Técnicas de la IA no local

A continuación se hace una muy breve explicación de que consisten las dos técnicas usadas para entrenar la IA no local, estas técnicas son el SVM y la Red Neuronal:

Las **Máquinas de Vectores Soporte(SVM)** son a grandes rasgos una técnica de aprendizaje, concretamente de clasificación, que encuentran una "superficie", con la que intenta separar los ejemplos que le hemos pasado del historial y que almacena como conjunto de entrenamiento, con el margen más grande posible, en nuestro caso al tener 4 atributos (PD, PO, PE, PP), y por tanto 4 dimensiones sería teóricamente una híper-superficie, que buscaría minimizar el error en la separación de los elementos y maximizar a su vez el margen de separación, mejorando por tanto la generalización del clasificador. También comentar que las máquinas de vectores soporte presenta un buen rendimiento al generalizar en problemas de clasificación, pese a no incorporar conocimiento específico sobre el dominio, por lo que la solución no depende de la estructura del planteamiento del problema, como en otras técnicas. Para más información consultar[5]

Una **Red Neuronal Artificial**, se puede definir muy brevemente como un conjunto de neuronas, clasificadas en capas, donde siempre hay una capa inicial y otra capa final, pudiéndose añadir la cantidad de capas ocultas deseada, formando una especie de malla (*Figura 6*), lo que realizamos al entrenarlas, es la adaptación de dicha malla al problema a través de la regulación de los pesos que hay entre ellas. Intentare explicarlo de una forma más mundana, imaginemos un traje de nuestro padre que tenemos y nos está muy grande (red neuronal inicial), en este caso tendríamos que reducir el tamaño de nuestra malla de neuronas, eliminando tela (capas ocultas), consiguiendo así un traje más adaptado a nuestro tamaño, del mismo modo si este fuera pequeño tendríamos que ampliarlo con la adicción de capas ocultas. Por último para adaptar las holguras que quedaran (pesos entre las neuronas) llamaríamos a un sastre, el cual a partir de la experiencia obtenida al visualizar nuestra figura podría adaptarnos más el traje hasta dejarlo como un guante (entrenamiento). Aunque esta explicación puede parecer fuera de lugar he creído conveniente añadirla, para que entiendan el funcionamiento básico de la red neuronal artificial todos aquellos no versados en el tema, para más información consultar[6].

Por último comentar que el código para la implementación de ambas técnicas se ha incluido en la *Sección A4 del Apéndice*.

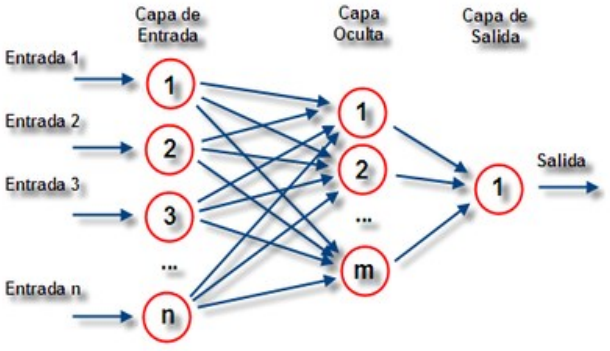


Figura 6

2.3.4 Consulta de la IA no local

La IA local consulta al servidor que estrategia le recomienda según el contexto en que se encuentre, dicho contexto se define por los porcentajes PD, PO, PE y PP que han sido explicados, haciendo una petición específica al SVM o a la red Neuronal, a la que el servidor responde con la estrategia que debería usar, o *null* en el caso de que aún no haya sido entrenado, o no disponga de datos suficientes. Un ejemplo del código que realiza esta funcionalidad, se ha incluido en la *Sección A4 del Apéndice*.

3 RESULTADOS VERSUS REQUISITOS

Nuestro fin primordial era cumplir con los requisitos y llevar a cabo la mayoría de los objetivos propuestos, por ello como resultados se realiza una disección de todos ellos, comentando cual ha sido el resultado de cada uno y haciendo especial hincapié en su grado de importancia. Comencemos por los requisitos:

3.4 Requisitos

En nuestro proyecto tenemos dos tipos de requisitos los funcionales y los no funcionales, por lo que a continuación, los comentaremos por separado:

3.4.1 Requisitos Funcionales

El primer requisito funcional(RF1) consistente en la existencia de un juego sencillo que permita la recogida de recursos, la compra de objetos y tecnologias, y llegar a una base enemiga para completar el juego, se ha completado satisfactoriamente, aunque se ha de comentar que el juego debería de haber sido más sencillo de lo que es en realidad.

El segundo requisito funcional(RF2) se basa en ofrecer al menos un modo de juego que permita evaluar y probar las limitaciones de la IA; en nuestro caso se implementaron dos, comentados anteriormente y definidos con el termino de variantes, por lo que también se ha obtenido un resultado satisfactorio en este requisito.

El tercer requisito funcional(RF3) establece que se debe implementar como mínimo un algoritmo o técnica de IA de forma local, en nuestro caso también han sido dos Backtracking para resolver el problema de la mochila y el algoritmo de estrella para resolver el problema de las ruta, por lo que se ha concluido con un resultado satisfactorio en este punto.

Por último el cuarto requisito funcional(RF4) consistia en la implementación de como mínimo una técnica de IA de forma no local que aprenda de las acciones del usuario. Este punto también ha sido solventado con el uso de un SVM y una red neuronal, y por tanto con resultado satisfactorio.

3.4.2 Requisitos No Funcionales

El primer requisito no funcional(RNF1) consiste en que el juego debia de ser compatible con diferentes navegadores y SO, y asi es, ya que esta implementado integramente en HTML5 y Javascript, por tanto el resultado tambien es satisfactorio.

El segundo requisito no funcional(RNF2) se basa en que se pudiera elegir entre diferentes técnicas para la IA No Local, y asi es, aunque la funcionalidad esta implementada y se puede modificar sencillamente en el código, no se ha establecido una opción para el jugador por consider que debia abstraerse de este elemento, por lo que siendo totalmente critico podriamos concluir en que el resultado en este punto no ha sido lo establecido inicialmente pero si satisfactorio.

El último de los requisitos no funcionales(RNF3) establece que el juego incrementara su dificultad gracias a la experiencia adquirida de forma local. El resultado para este punto ha sido no satisfactorio pues surgió el siguiente problema, al tener una IA Local y otra IA No Local, nos encontramos con el dilema de saber cuál de las dos estrategias recomendadas es mejor, y al final optamos por dejar la IA No Local como un elemento al que se le consulta y no se le termina por hacer caso.

Requisitos	Resultado	
	Satisfactorio	No Satisfactorio
RF1	X	
RF2	X	
RF3	X	
RF4	X	
RNF1	X	
RNF2	X	
RNF3		X
RESULTADO:	6/7 de los Requisitos son Satisfactorios	

Tabla 1

2.4 Objetivos

En nuestro proyecto tenemos una diversidad de objetivos, de los cuales ya hemos dado una descripción especifica de todos ellos en la sección de Metodología y que a su vez forman parte de las 3 fases en las que se dividio el proyecto, aqui se repasan para dar finalmente una valoración de su cumplimiento:

- **Objetivos de la Fase 1** (Crear Videojuego)
  - Crear un Mapa
  - Crear Movimientos del Jugador.
  - Crear Lógica del Juego.
  - Mejorar Apariencia Gráfica.
  - Mejorar Rendimiento.
  - Implementar Variabte Visibilidad.
  - Implementar Variante Memoria.

- **Objetivos de la Fase 2**(Crear IA Local)
  - Implementar Backtracking para resolver el problema de la mochila.
  - Implementar el Algoritmo de Estrella.
  - Implementar varios tipos de Estrategias.

- **Objetivos de la Fase 3**(Crear IA No Local)
  - Diseñar e implementar el servidor.
  - Implementar técnicas de aprendizaje en el servidor.
  - Implementar método de consulta de la IA local para consultar al Servidor.

Como se ha explicado en la sección de Metodología, los únicos puntos no satisfactorios, han sido la mejora de la apariencia gráfica y las implementación de estrategias en la IA Local por lo que obtenemos como resultado 11/13 objetivos propuestos realizados satisfactoriamente.

4 Futuras Mejoras

Las futuras mejoras son muchas y muy variadas, a continuación se da un listado de varias de ellas, pero podrían ser muchas más:

La primera será comenzar por el correcto desarrollo de las estrategias para que la IA Local tenga una autonomía real, además de incluir estrategias más complejas que también incluyan las estrategias básicas con el fin de dar más dinamismo a los jugadores no-humanos.

Mejorar el apartado gráfico necesario para una buena aceptación del público en general.

También sería necesaria una optimización de los parámetros que utilizan el SVM y la Red Neuronal, para conseguir que el entrenamiento fuera más óptimo.

En lo que concierne al juego habría que buscar alguna forma de eliminar la necesidad de regresar a la base para dejar los recursos, pues el juego se hace eterno y da como resultado una experiencia poco agradable para el jugador, además de implementar la posibilidad de que varios jugadores humanos jueguen en la misma partida.

Por último una forma de comparar si la estrategia recomendada por el servidor es mejor en términos de eficiencia y adaptación que la que recomienda la IA Local.

5 Conclusión

Hemos podido comprobar que el objetivo primordial, que es crear una plataforma para el aprendizaje online en un videojuego en la web es posible y que por tanto puede llevarse a cabo un proyecto de un gran interés, en un futuro creo que se verán bastantes más ejemplos de este tipo de proyectos, pues las tecnologías web están

creciendo exponencialmente ofreciendo posibilidades de desarrollo e integración sencillas para las técnicas de IA, que antes eran muy complejas y laboriosas, también destacar que para mí ha supuesto una gran ayuda para centrar todos los conocimientos adquiridos que tenía dispersos y así lograr asentarlos.

Agradecimientos

A Ramon Baldrich por su tiempo y por guiarme en el desarrollo del proyecto y a la UAB (Universidad Autónoma de Barcelona) por haberme dado los conocimientos necesarios para el desarrollo del proyecto.

Bibliografía

[1] "botprize : home." [Online]. Available: http://botprize.org/. [Accessed: 11-Feb-2014].

[2] "Roberto Martínez: Reporte 2: Problema de la mochila." [Online]. Available: http://roberto-mtz.blogspot.com.es/2011/07/reporte-2-problema-de-la-mochila.html. [Accessed: 11-Feb-2014].

[3] "46 Dogs: A Star (A*) path/route finding Javascript code." [Online]. Available: http://46dogs.blogspot.com.es/2009/10/star-pathroute-finding-javascript-code.html.

[4] "Browse by Keyword: 'machine%20learning.'"[Online]. Available: https://www.npmjs.org/browse/keyword/machine learning. [Accessed: 11-Feb-2014].

[5] "Máquinas de vectores de soporte - Wikipedia, la enciclopedia libre." [Online]. Available: http://es.wikipedia.org/wiki/Máquinas_de_vectores_de_soporte.

[6] "Red neuronal artificial - Wikipedia, la enciclopedia libre." [Online]. Available: http://es.wikipedia.org/wiki/Red_neuronal_artificial.

APÈNDICE

A1. EJEMPLO DE UNA IMPLEMENTACIÓN PARA EL CÁLCULO DE PORCENTAJES EN JAVASCRIPT

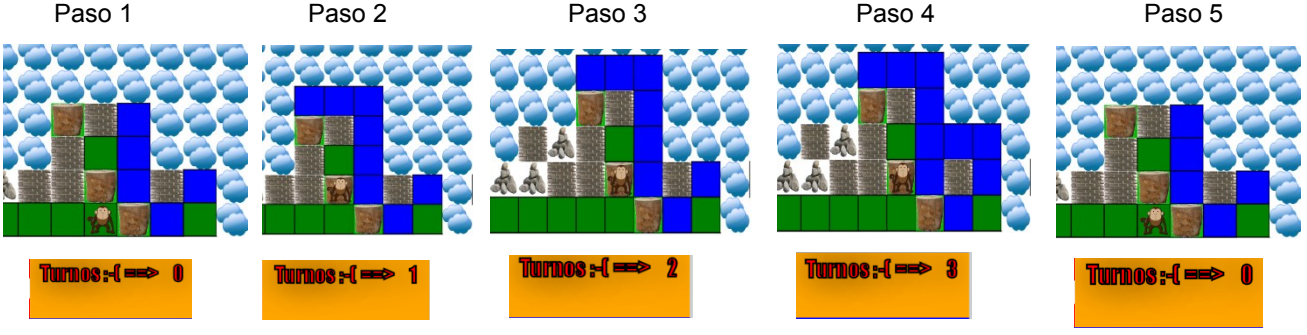
```
var porcentajeUnidadX = (100 / mapaNRectX);
    var porcentajeUnidadY = (100 / mapaNRectY);
    var distanciaDeXaGuarida = Math.abs(posXGuarida - posX);
    var distanciaDeYaGuarida = Math.abs(posYGuarida - posY);

    this.porcentajeAvanceDefensivo=CalcularPorcentajeAvanceDefensivo();
    this.porcentajeAvanceOfensivo=CalcularPorcentajeAvanceOfensivo();
    function CalcularPorcentajeAvanceDefensivo() {
        var porcentajeX = 100 - (porcentajeUnidadX * distanciaDeXaGuarida);
        var porcentajeY = 100 - (porcentajeUnidadY * distanciaDeYaGuarida);
        return porcentajeAvanceDefensivo = (porcentajeX + porcentajeY) / 2;
        console.log("INFO CASILLA-->Porcentaje Defensivo: " + this.porcentajeAvanceDefensivo);

    }
    function CalcularPorcentajeAvanceOfensivo() {
        var porcentajeX = porcentajeUnidadX * distanciaDeXaGuarida;
        var porcentajeY = porcentajeUnidadY * distanciaDeYaGuarida;
        return porcentajeAvanceOfensivo = (porcentajeX + porcentajeUnidadY) / 2;
        console.log("INFO CASILLA-->Porcentaje Ofensivo: " + this.porcentajeAvanceOfensivo);
    }
```

A.2 ALGORITMOS AUXILIARES Y ALGORITMOS TEÓRICOS AUXILIARES

A continuación se da un pequeño ejemplo de un algoritmo auxiliar implementado en el juego y de dos que considero de utilidad pero que no se llegaron a implementar.

Este primer algoritmo se utiliza por las estrategias para optimizar la recogida de recursos como es utilizar el tiempo en el cual está inhabilitado para explorar las zonas adyacentes, a continuación hago una explicación detallada de dicho comportamiento, en usa secuencia de pasos[*Ilustración 5*]:

1. Llegamos desde abajo dirección hacia arriba.
2. Nos quedamos atascados por un recurso, a partir de ahora tenemos 3 turnos para explorar el entorno.
3. Exploramos la izquierda, y quedan 2 turnos de inhabilitación.
4. Exploramos la derecha y queda 1 turno de inhabilitación.
5. Como ya tenemos la información podemos colocarnos hacia la posición que deseamos ir para continuar por allí.



A continuación se describirán dos algoritmos que ayudarían a la estrategia de defensa, pero como se ha comentado solo forman parte de la teoría, pues no han sido implementados en el proyecto:

Uniformidad del campo descubierto:

A la hora de prevenir un futuro ataque se premiara que el rango sea más general, antes que parcial, ya que una vista general nos hará saber que el otro jugador está cerca y monitorizarlo, de la otra forma, no podremos saber de forma concreta el porqué de pasar por esa zona.

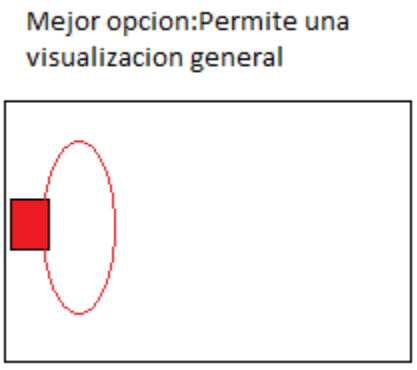


Figura 7

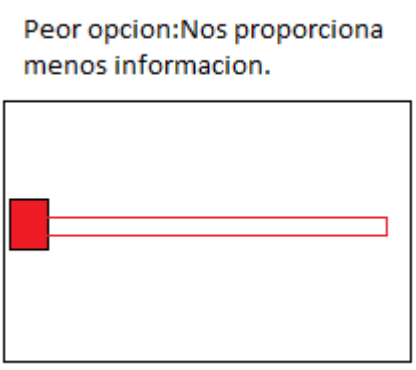


Figura 8

Amurallamiento y Colocación de Puertas:

La estrategia defensiva puede tener dos tipos de amurallamiento y a su vez, diferentes algoritmos para colocar las puertas. La primera es la colocación de una capa o más de murallas, se puede hacer el paralelismo con una cebolla; una vez que hay diferentes capas la colocación de las puertas nos puede dar ventajas defensivas o facilitarnos la recogida de recursos, esto se puede medir calculando la separación entre las diferentes puertas.

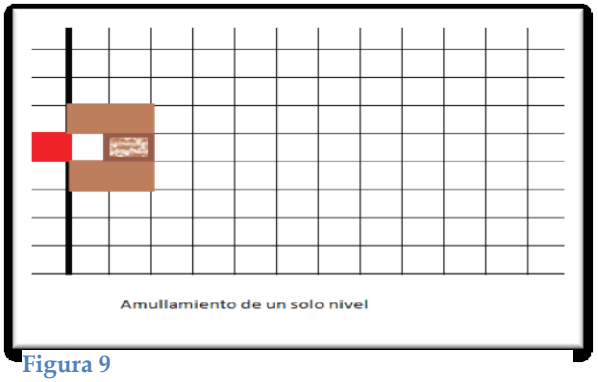


Figura 9

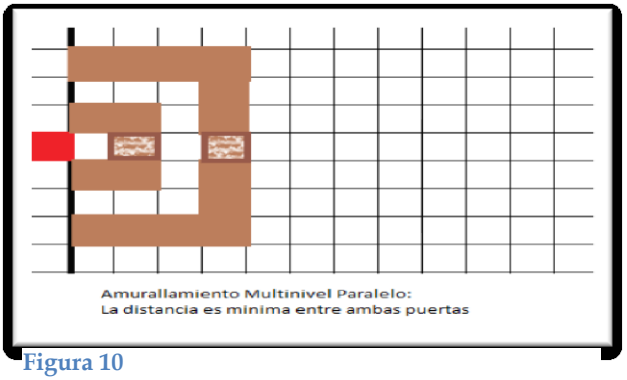


Figura 10

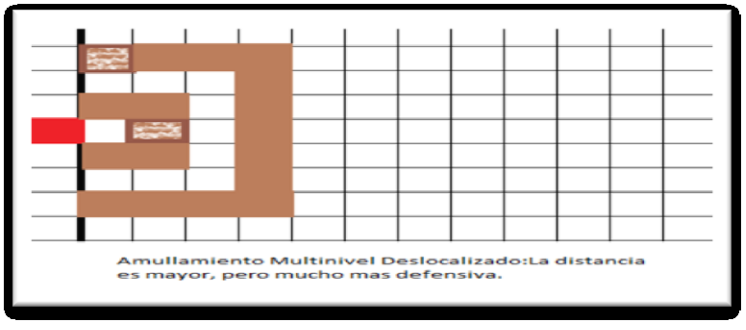


Figura 11

Nota: La inserción de trampas permitiría ampliar en gran medida el retardo del oponente en llegar a nuestra guarida, si se aplica el método deslocalizado, a su vez se puede calcular el porcentaje defensivo que tiene sumando las casillas entre la puerta más interior y la más exterior.

A.3 OBJETOS Y TECNOLOGÍAS

En esta sección se hace un listado de los objetos que se pueden comprar en el juego, además de dar una pequeña definición sobre su funcionalidad cuando se usan:

- Carro:
  - Permite aumentar el tamaño de la mochila para llevar mas objetos.
  - Solo puede haber un objeto Carro en la mochila simultaneamente.
- Pico
  - Permite recolectar los recursos de Piedra y Metal
  - Solo puede haber un objeto Pico en la mochila simultaneamente.
- Hacha
  - Permite recolectar el recurso Madera
  - Solo puede haber un objeto Hacha en la mochila simultaneamente.
- Puente
  - Permite colocar un puente sobre una casilla de agua.
  - Puede haber más de un objeto Puente en la mochila.
- Puerta
  - Permite colocar una puerta que solo puede atravesar el jugador propietario.
  - Puede haber más de un objeto Puerta en la mochila
- Muro
  - Permite colocar un muro que no puede atravesar ningún jugador.
  - Puede haber más de un objeto Muro en la mochila.
- Trampa
  - Permite colocar una trampa que inhabilitara al jugador según el nivel de esta.
  - Con el uso de la tecnologia “Subir Nivel de Trampa” se puede subir el nivel de la Trampa.
  - Hay 5 niveles de Trampa que permiten hasta un maximo de 5 turnos de inhabilitación.
  - Puede haber más de un objeto Trampa en la mochila.
- Cuchillo
  - Permite atacar a un jugador enemigo que se encuentre a una casilla o menos del jugador propietario del objeto.
  - Solo puede haber un objeto Cuchillo en la mochila simultaneamente.
- Honda
  - Permite atacar a un jugador enemigo que se encuentre a dos casillas o menos del jugador propietario del objeto.
  - Solo puede haber un objeto Honda en la mochila simultaneamente.
- Ariete
  - Permite destruir puertas y muros de un jugador enemigo que se encuentren a una casilla de distancia.
  - Solo puede haber un objeto Ariete en la mochila simultaneamente.
- Catapulta
  - Permite destruir puertas y muros de un jugador enemigo que se encuentren a una casilla de distancia.
  - Solo puede haber un objeto Ariete en la mochila simultaneamente.

PRIORIDAD	ESTRATEGIA		
	RECOGIDA	DEFENSIVA	OFENSIVA
1	Carro	Carro	Carro
2	Pico	Pico	Pico
3	Hacha	Hacha	Hacha
4	Puente	Muro	Puente
5	Puerta	Puerta	Catapulta
6	Muro	Trampa	Ariete
7	Trampa	Catapulta	Trampa
8	Cuchillo	Honda	Honda
9	Honda	Cuchillo	Cuchillo
10	Ariete	Ariete	Puerta
11	Catapulta	Puente	Muro

Tabla 2

También se hace un listado de las tecnologias que se pueden obtener en el juego, además de dar una pequeña definición sobre su funcionalidad cuando se usan:

- Subir Nivel Pico
  - Nivel 1: Se tarda 3 turnos en recoger un recurso Metal o Piedra.
  - Nivel 2: Se tarda 2 turnos en recoger un recurso Metal o Piedra.
  - Nivel 3: Se tarda 1 turnos en recoger un recurso Metal o Piedra.
- Subir Nivel Hacha
  - Nivel 1: Se tarda 3 turnos en recoger un recurso Madera.

  - Nivel 2: Se tarda 2 turnos en recoger un recurso Madera.
  - Nivel 3: Se tarda 1 turnos en recoger un recurso Madera.

- Subir Nivel Trampa
  - Nivel 1: Inhabilita al jugador enemigo que la pise 1 turno.
  - Nivel 2: Inhabilita al jugador enemigo que la pise 2 turno.
  - Nivel 3: Inhabilita al jugador enemigo que la pise 3 turno.
  - Nivel 4: Inhabilita al jugador enemigo que la pise 4 turno.
  - Nivel 5: Inhabilita al jugador enemigo que la pise 5 turno.

La prioridad en este caso es estricta, primero van el Pico y el Hacha indistintamente y cuando se ha conseguido el nivel máximo de ambos se comienza a incrementar el nivel de Trampa.

A4. CÓDIGO DE EJEMPLO DE LA IA NO LOCAL

Implementación y configuración del SVM

Se ha utilizado un SVM multiclase, donde como parámetros se introduce el historial (Data), sin la estrategia y en labels la estrategia misma, como sigue a continuación:

```
Data= {{[PO] [PD] [PE] [PP]}1,
       {[PO] [PD] [PE] [PP]}2,
       ……
       {[PO] [PD] [PE] [PP]}n}

Labels = {{[Estrategia Usada]}1,
          {[Estrategia Usada]}2,
          ……
          {[Estrategia Usada]}n}
```

El proceso seguido para entrenarlo ha sido el siguiente:

1. Se crea una nueva instancia de un objeto SVM:
   ```
   svm = new svmjs.SVM();
   ```
2. Entrenamos el SVM:
   a. Se configura, por ejemplo con un kernel gaussiano, con el valor de sigma en 0.5, aunque se podrían haber usado otras configuraciones o parámetros como por ejemplo un kernel lineal
      ```
      svm.train(Data, Labels, { kernel: 'rbf', rbfsigma: 0.5 });
      ```
   b. Se entrena el SVM:
      ```
      svm.train(Data, Labels, { kernel: 'linear' });
      ```
3. Cuando recibo una petición de la IA Local hago una predicción de la estrategia como sigue:
   ```
   var testlabels = svm.predict(Data);
   ```

Implementación y configuración de la red neuronal artificial

Para la red neuronal artificial he adaptado se han adaptado los parámetros de entrada al formato exigido por la librería

```
Data= {input:{ [PO] [PD] [PE] [PP]}1 output: {[Estrategia Usada]}1,
       {input:{ [PO] [PD] [PE] [PP]}2 output: {[Estrategia Usada]}2,
       ……
       {input:{ [PO] [PD] [PE] [PP]}n output: {[Estrategia]}n}
```

El proceso seguido para entrenarlo ha sido el siguiente:

1. Se crea una nueva instancia de un objeto de Red Neuronal Artificial:
   ```
   var net = new brain.NeuralNetwork();
   ```
2. Se entrena el modelo configurando los diversos parámetros:
   ```
   net.train(Data, {
           errorThresh: 0.004, // error threshold to reach
           iterations: 20000, // maximum training iterations
           log: true, // console.log() progress periodically
           logPeriod: 10 // number of iterations between logging
   })
   ```
3. Se hace una predicción de una estrategia con los parámetros obtenidos de la IA Local:
   ```
   var output = net.run({[PO] [PD] [PE] [PP]});
   ```