Received 4 December 2023, accepted 5 January 2024, date of publication 17 January 2024, date of current version 1 February 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3354714

RESEARCH ARTICLE

A Metaheuristic Search Algorithm Based on Sampling and Clustering

MARIA HARITA, ALVARO WONG, REMO SUPPI, DOLORES REXACHS, AND EMILIO LUQUE
Department of Computer Architecture and Operating Systems, Universitat Autònoma de Barcelona, 08193 Bellaterra, Spain

Corresponding author: Maria Harita (MariaDeLosAngeles.Harita@autonoma.cat)

This work was supported in part by the Spanish "Agencia Estatal de Investigacion (AEI)", Spain, and the "Fondo Europeo de Desarrollo Regional (FEDER), Unión Europea", under Contract PID2020-112496GB-I00; and in part by the "Fundacion Escuelas Universitarias Gimbernat (EUG)".

ABSTRACT As optimization problems become more complicated and extensive, parameterization becomes complex, resulting in a difficult task requiring significant amounts of time and resources. In this paper we propose a heuristic search algorithm we call MCSA (Montecarlo-Clustering Search Algorithm), which is based on Montecarlo sampling, and a clustering strategy involving two techniques. Our objective is to apply MCSA, an inherently stochastic method, to address optimization problems. To assess its performance, we conducted an evaluation using classical benchmark optimization functions. Additionally, we leveraged the CEC2017 benchmark suite to comprehensively evaluate the algorithm, highlighting the pivotal role of the Exploration stage in our methodology. Subsequently, we extended our methodology to tackle a practical combinatorial problem, the Knapsack problem. This NP-Hard problem holds significant real-world applications in resource allocation, scheduling, planning, logistics, and more. Our contributions lie in parameterizing the Knapsack Problem to align with MCSA's parameters for reference indicators adjustment and achieving high-quality solutions, surpassing 90% in comparison to exhaustive methods such as branch and bound.

INDEX TERMS Benchmarks, heuristic methods, Knapsack problem, Montecarlo and clustering methods, optimization.

I. INTRODUCTION

Recently, optimization problems are becoming more complicated and extensive, hence the inability of accurate optimization methods to solve complex and multidimensional problems. Therefore, approximation algorithms have been proposed as a new approach to solving such problems. Each type of optimization has some challenges. Single-objective optimization, however, can be included in many other types of optimization, like constrained optimization, multi-modal optimization, multi-objective optimization, etc. [1], [2]. Regarding the dimensionality of an optimization problem, it is challenging, since the search space becomes enormous when the dimensions of a given problem in-crease [3]. The domains of the variables determine a search space (lower and upper bounds). However, due to the complexity of continuous optimization problems, that cover a wide range of different issues, even mathematical methods (used to solve scientific and engineering problems) have had difficulties in regards to the extensive calculation needed [4]. Therefore, the use of meta-heuristic algorithms has grown, and now they are a reliable tool for solving various search problems, both continuous and discrete.

Meta-heuristic algorithms benefit from the Exploration process, since they can produce new solutions while visiting the search space. Gradually, the algorithms transit to Exploitation, focusing on the accuracy improvement of the solutions obtained in the Exploration phase. On the other hand, the Exploitation process generally produces a new solution based on the best solution available [5], [6], [7], [8]. As a result, the challenge that meta-heuristic algorithms face lies in using essential Exploration and Exploitation processes to avoid getting trapped in the local best solution and converging towards the target [9], [10].

The associate editor coordinating the review of this manuscript and approving it for publication was Utku Kose.

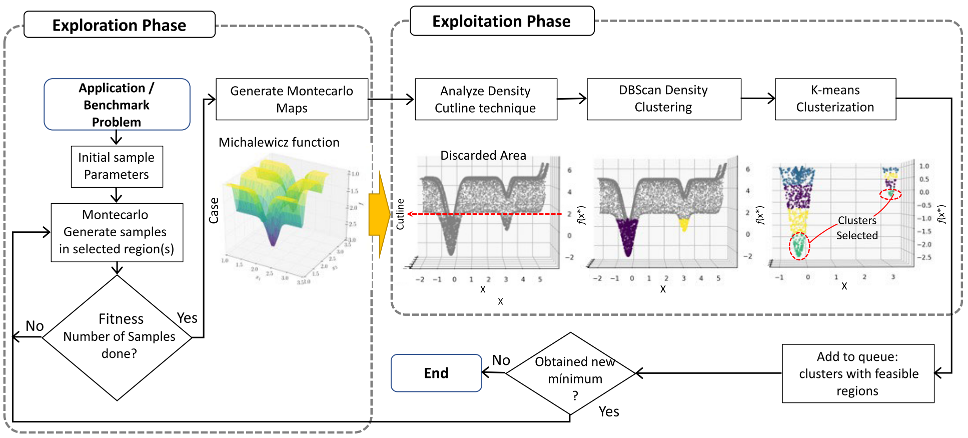

Exploration Phase
Application / Benchmark Problem
Initial sample Parameters
Montecarlo Generate samples in selected region(s)
Fitness Number of Samples done?
No
Yes
Case
Generate Montecarlo Maps
Michalewicz function
Exploitation Phase
Analyze Density Cutline technique
DBScan Density Clustering
K-means Clusterization
Discarded Area
Cutline
Clusters Selected
Add to queue: clusters with feasible regions
Obtained new minimum ?
No
Yes
End

FIGURE 1. Overview of MCSA heuristic method [44].

In this work, we present a heuristic methodology that features the Montecarlo Clustering Search Algorithm (MCSA) to solve optimization problems. MCSA consists on two phases: Exploration and Exploitation, as seen in Figure 1. Exploration is based on random uniform sampling by Montecarlo techniques, and in Exploitation, we perform an unsupervised clustering strategy to analyze density, allowing for a centroid-based clustering afterwards via K-means [11]. The clustering process, at the heart of MCSA's Exploitation phase, accomplishes two key objectives: it effectively reduces the search space and efficiently identifies feasible regions containing optimal values.

Furthermore, MCSA conducts an iterative search process. Once it identifies a K-cluster (or multiple K-clusters) containing promising solutions, it initiates a subsequent exploration phase. This phase revolves around these viable regions, where new random samples are generated and leveraged to enhance the solution, bringing it closer to the optimum.

In order to test the scope of the methodology, we set out to solve a combinatorial optimization problem. To this extent, the Knapsack Problem is part of a historical list of NP-Complete problems elaborated by Richard Karp [12], and it also appeared in [13]. The binary Knapsack Problem (KP01) has been treated with a multitude of methods, some more efficient than others in terms of quality of the solution (such as greedy algorithms). As we have reviewed, a problem is classified as inherently difficult if its solution requires onerous computational resources, regardless of the algorithm used. This has also been carried out through heuristic methods [14], sometimes trading solution quality for a reduction of the search time, or sometimes seeking a reduction of the search space [15].

Therefore, we present a novel approach to tackle KP01. Our strategy involves organizing the input data-set within a discretized search space, structured in up to $n - dimensions$. We set the problem size as being equal to the power-set of the $n - elements$. The core aspect of our approach lies in the distribution of elements along with their combinations, across the $n - dimensions$. Consequently, as the number of dimensions increases, the number of elements to be arranged within each dimension decreases. Notably, our approach has demonstrated solution qualities exceeding 90% on average when addressing instances containing up to 2000 elements achieving near-optimal solutions.

Since we are aware that quality evaluation is important, MCSA was evaluated through the benchmark functions for optimization [16], [17], [18], achieving promising results, optimal in 60% of the test cases, and near optimal in the remaining 40% given the complexity of the chosen functions.

The paper is organized as follows: Section II describes the Related Works, introducing meta-heuristic algorithms. Section III explains the Background of MCSA Methodology, and its evolution into a more robust algorithm along with the Evaluation tests that we carried out with benchmark functions. Section IV presents how we adapted the method to solve the Knapsack Problem, and brought its input parameters to-wards the parameters of MCSA. Also in this Section, we provide a proposal to solve the multi-objective variant of the Knapsack Problem with our MCSA method. We discuss the results in detail on Section V, and finally present the conclusions in Section VI.

II. RELATED WORKS

As optimization problems become more complicated, optimization algorithms play an essential role in this transformation, which usually at-tempts to characterize the type of search strategy through an improvement on simple local search algorithms. However, in cases where the search space is enormous, an exhaustive search, iterative methods, or simple heuristics are impractical. In contrast, meta-heuristic ideas [19], some-times classified as global search algorithms, can often find reasonable solutions with less computational effort.

Probabilistic distribution methods, such as Montecarlo, offer flexible forms of approximation with some advantages regarding cost. There are useful statistical techniques available, but the Montecarlo methods use randomness to solve problems, which is useful when it becomes difficult to apply other approaches. The algorithms based on random searches are often globally convergent. Thus, we find the probabilistic methods such as the stochastic method [20], Andradottir's random search methods [21], the stochastic comparison method [22], and nested partitions with specific implementations as in [23], among other algorithms. Any globally convergent algorithm requires examining each feasible solution often to ensure convergence, because, a part of an optimization algorithm uses the algorithm's information to help decide which solution between the candidates should be tested.

In this sense, other approaches use similarity or meta-heuristic algorithms to solve high-dimensional optimization problems, which are validated using large-scale functions [24]. However, they are prone to fall into local optimum values. To solve global optimization problems, the most recognized approaches that make use of global exploration are meta-heuristic algorithms. For example, many researchers have developed evolutionary-type algorithms [25], since they provide a harmonious balance between Exploration and Exploitation [26], [27], [28].

Evolutionary algorithms are inspired by natural biological evolution, such as reproduction, mutation, recombination, and selection. Some examples are the Genetic Algorithm [13], which has been used with the most essential combinatorial and mutational operators and is considered a successful algorithm, accounting for its many variations. In the last decade, many optimization algorithms have been proposed, generally consisting of swarm-based, physics-based, genetics, etc [29]. Evolution Strategy [30] or Evolutionary Programming [31] are other common naturally inspired approaches to global optimization problems. Further details regarding the vast state of the art on optimization algorithms are reviewed in [32] and [33].

While multiple approaches develop continuously, we found that, among the combinatorial optimization methods, as Cabrera et al. [34] argument, the efficiency gains regarding the application of sampling and grouping techniques should be explored to solve problems of a complex nature, that sometimes involve dynamic and strongly human-dependent systems.

Cluster analysis algorithms are a key element of exploratory data analysis. In a broad sense, clustering algorithms can be categorized into partition-based, hierarchy-based, density-based, and grid-based methods. Notably, the most widely used clustering algorithms are K-means [35] and DBSCAN [36], [37], [38], the latter being a popular clustering algorithm rooted in density-based principles. It identifies clusters by detecting high-density areas separated by low-density areas relying on the concept of cluster density. DBSCAN is suitable for finding clusters of any shape in a spatial database and connecting adjacent regions with corresponding density. Its robustness extends to handling outlier data, making it particularly valuable for spatial data clustering [39].

III. MCSA HEURISTIC METHODOLOGY

In this section, we present MCSA which is based on random sampling by Montecarlo techniques and a set of clustering strategies [44].

The method was initially developed to solve a particular type of optimization problem that featured a double-objective scheme [40], and whose solution was ultimately within the limits of the Pareto optimal. Solving a multi-objective optimization problem is approximating a representative set of Pareto optimal solutions. Cabrera et al. approach was an interesting starting point because two-objective optimization brings out the essential features of multi-objective optimization that pursued the best scenario to manage a Medical Emergency Department, and their methodology was validated through an exhaustive method, or using brute force, which helped to prove the results' quality when using a combined heuristic.

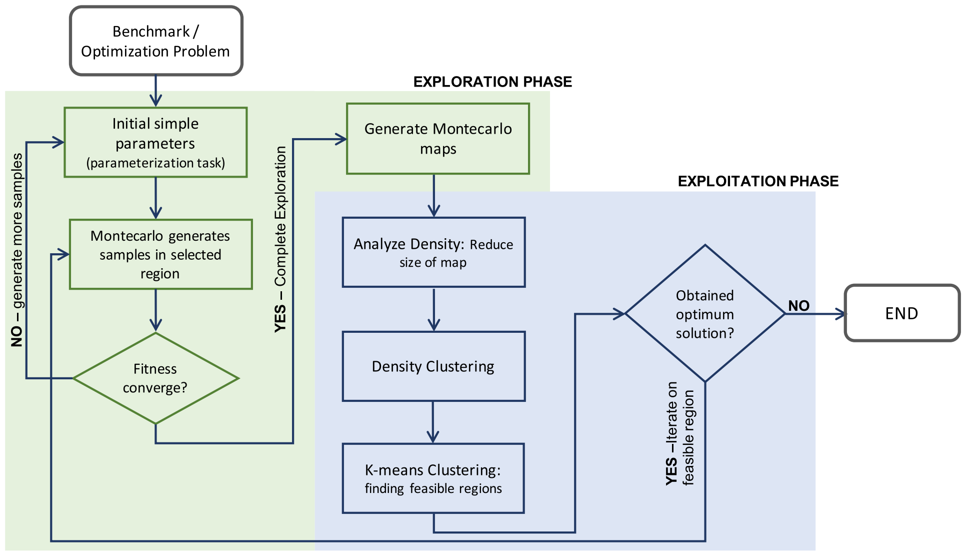
We have devised an enhanced methodology that builds upon the Montecarlo sampling approach. This methodology incorporates a novel clustering strategy that synergizes two techniques to conduct a more precise search, con-sequently reducing the search space. The resulting regions are meticulously analyzed as sample concentrations, offering valuable insights that can lead us to the optimal solution or a highly promising one.

In Figure 2, we show MCSA flowchart. The base method consists of two phases. The first phase, Exploration, includes a coarse-grained approach, which carries out a global exploration of the discrete ordered $n - dimensional$ search space to find promising regions identified on the neighborhood structure of the problem. This phase uses Montecarlo-based heuristics, returning a collection of fit samples or a set of in-dependent and homogeneously distributed variables that can be represented in the form of a map or in a landscape. At this stage, one of our main objectives is to obtain a uniform sample as homogeneous as possible. Thus, we ensure that the search process will be even.

Regarding the convergence criterion, we employ a fitness function that relies on calculating the mean of each Montecarlo sample set with a fixed precision. These samples ultimately form the solution map. This map comprises all the fit samples that meet specific criteria or restrictions.

At this juncture, the Exploration Phase of the methodology concludes, marking the gradual transition into the Exploitation Phase. To facilitate this transition, we introduce a crucial step involving map reduction. This entails making a cut procedure along the objective function Axis, thereby allowing us to seamlessly transition into the subsequent clustering analysis. This analysis constitutes the primary focus of the Exploitation phase.

The iterative behavior of MCSA, allows to re-explore feasible regions, improving the solution, and contributing to



Benchmark / Optimization Problem
EXPLORATION PHASE
Initial simple parameters (parameterization task)
Montecarlo generates samples in selected region
Fitness converge?
NO – generate more samples
YES – Complete Exploration
Generate Montecarlo maps
EXPLOITATION PHASE
Analyze Density: Reduce size of map
Density Clustering
K-means Clustering: finding feasible regions
Obtained optimum solution?
NO
END
YES –Iterate on feasible region

FIGURE 2. **Flowchart of the MCSA heuristic methodology [44].**

progress while approaching to the global minimum, hence increasing the quality of the results. With high accuracy, the user can expect to obtain the value closest to the optimum. In addition, it may contribute to save both time and human resources, considering the enormous amount of data to be calculated and resources that sometimes are needed to reach a solution [40].

A. EXPLORATION PHASE

Montecarlo techniques initiate by generating random samples that uniformly cover the discretized search space, creating a representative sample set for the problem to be evaluated within the objective function $f(x)$. The fitness function converges with a fixed precision of $E-02$ based on the average of previous values. This convergence criterion necessitates at least two iterations of the Montecarlo process for comparison. Once an initial sample is obtained, it can be visualized as a logical map, particularly evident in plots when dealing with low-dimensional problems. For instance, in an optimization problem with just one decision variable, the resulting fitness landscape typically takes the form of a nonlinear line. Similarly, for problems with two decision variables, the fitness landscape becomes a two-dimensional surface. In contrast, for higher-dimensional problems featuring three or more decision variables, the fitness landscape becomes a hyperplane, which poses challenges for easy visualization. We want to emphasize that we have imposed a strict upper limit on the percentage of random samples generated, which is fixed at 15%. Our analysis confirms that this limit was never surpassed in any scenario.

In the quest for the globally optimal solution, success depends on factors such as the size of the area to be searched, the landscape smoothness, and the behavior of the search method. In the context of optimization problems, we define the objective function as $f(x)$, representing a real-valued function linked to one or more decision variables, aiming for maximization or minimization. It is worth noting these aspects because the search strategy we propose is capable of analyzing wider regions and cross valleys in the fitness landscape to achieve better results. Clearly, the methodology is scalable to problems of larger dimensions.

After the initial execution, MCSA enters a series of iterations aimed at refining the solution. During these iterations, it returns to the Exploration phase, with a focus on feasible regions delineated by clusters containing the most favorable values. The purpose of these iterations is to achieve increasingly accurate results. However, the algorithm limits the number of iterations, and the stopping criterion is determined by variations in solution values. If these variations become negligible, the algorithm considers the current solution as the best possible.

B. EXPLOITATION PHASE

In this phase, we delve into an investigation of the algorithm's efficiency. The transition into this phase is gradual and occurs after the cutting process has been executed. As previously mentioned, one of the challenges encountered during the adaptation of the density clustering method to the MCSA methodology was related to the expanding problem size. To successfully complete our two-step Exploitation process using clustering methods, we cannot initiate the search for densities directly on the initial Montecarlo map. Directly applying DBSCAN, for instance, would result in a single cluster, necessitating an analysis of the entire sample.

Hence, it is advantageous to reduce the sample map, a task achieved by cutting the sample along the objective function $f(x)$ Axis.

To provide a clearer insight into the cutting process, we present a Flowchart in Figure 3, where we have decomposed the steps that would lead us to implement this cutting proposal on the sample map in order to reduce it and enhance the analysis and clustering of the data. It also details the procedure for cutting through the sample map and eliminating irrelevant samples.

Initially, we determine the lower and upper bounds based on the values of $f(x)$ obtained from the resulting samples (*Distance* and *Range* in the flowchart). This division results in ten fixed sections ($MaxSteps = 10$), each representing a heuristic 'Step' based on $f(x)$. We then commence the sweep from the lower bound of each section until one of two stopping criteria is met. Either the heuristic 'Step' is successfully completed, or we identify multiple regions with a high concentration of samples.

In the first scenario, after completing a heuristic step (initial cut proposal), we calculate the average of the values within the resulting sample concentration to assess the potential for grouping concentrations into clusters. These calculations are updated at each step ($Step++$).

If, during one step, more than one cluster is identified ($CurrNumClusters >= PrevNumClusters$), and in subsequent steps ($Step <= 10$), the number of clusters either remains unchanged or decreases ($PrevNumClusters = CurrNumClusters$), the alternative stopping criterion is triggered. This leads to a return to the previous step, where the algorithm divides the data into two separate regions that can be treated as distinct clusters. This approach enables a more in-depth exploration of regions with potential values.

We say we are entering the Exploitation Phase once that the resultant areas (with concentrations of samples) are delimited. As depicted in Figure 4, systematically sweeping the data along the $f(x)$ Axis starting from its lower bound (the minimum sample value) allows for the exclusion of samples that do not need to be analyzed.

The same Figure provides information about the number of potential clusters discovered at each step of the heuristic process. The Y-axis corresponds to the range of function values, while the X-axis represents the function's domain. Furthermore, the arrow in the Figure signifies the direction of this sweeping operation, involving a partial analysis of the sample. Lastly, the dashed line denotes the section where the proposed cut will be executed.

After this step, the Montecarlo map is clustered to pinpoint regions predominantly characterized by sample concentrations, with adherence to DBSCAN-configured parameters, such as the minimum number of samples per cluster and the inter-sample distance. In this specific case, two clusters will be identified and formed.

The second clustering process utilizes K-means. This particular sequencing enables us to discern feasible regions with greater precision. Additionally, it addresses the limitations we encountered with distance criteria, upon which many centroid-based clustering methods, including K-means, rely. These criteria did not facilitate an accurate analysis of the Monte Carlo map for our specific objectives. Consequently, we have devised this dual clustering strategy that incorporates a density analysis.

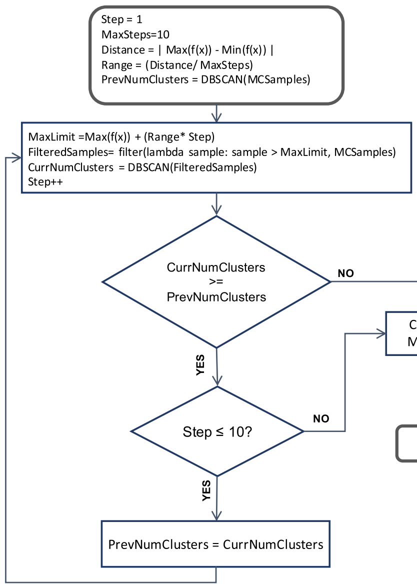

Step = 1
MaxSteps=10
Distance = | Max(f(x)) - Min(f(x)) |
Range = (Distance/ MaxSteps)
PrevNumClusters = DBSCAN(MCSamples)
MaxLimit =Max(f(x)) + (Range* Step)
FilteredSamples= filter(lambda sample: sample > MaxLimit, MCSamples)
CurrNumClusters = DBSCAN(FilteredSamples)
Step++
CurrNumClusters >= PrevNumClusters
NO
YES
CutLine= MaxLimit
Step ≤ 10?
NO
YES
END
PrevNumClusters = CurrNumClusters

FIGURE 3. Flowchart illustrating the cutting process for sample reduction.

Figure 5 offers an illustrative example demonstrating how the implementation of the cutting process results in the removal of redundant samples. Consequently, it presents the advantage of obviating the need for an exhaustive analysis of every single sample.

In this scenario, two distinct regions were identified and subsequently selected as the final clusters. At this point, the remaining portion of the sample is discarded, and we proceed to initiate the k-means clustering process. The clusters are formed around four centroids to classify the values based on their distance to the centroid.

This allows for the selection of the cluster (or clusters) containing the most promising values for further analysis. The initial selection often provides a solution, which in some cases can be deemed feasible. In other instances, it may be necessary to iterate a specific number of times to refine the solution. Consequently, conducting further exploration within the bounded regions and exploiting the new values becomes crucial for enhancing the solution. In cases where multiple K-clusters contain numerous identical values that can be considered feasible solutions, one might infer that the problem resembles a function with multiple global minima.

Thus far, the proposed MCSA methodology enables us to operate within reduced regions of the discrete search space. This is achieved through the combined clustering techniques that constitute a successful Exploitation phase. Moreover, the iterative nature of MCSA helps prevent being trapped in local minima, thereby offering the opportunity to discover better values compared to the ones found initially.

C. QUALITY EVALUATION OF MCSA HEURISTIC METHOD

To evaluate the heuristic MCSA thoroughly, we selected a set of different benchmarks, seeking further for more complicated multi-dimensional functions to increase the difficulty in optimization and stress the algorithm. A balance between

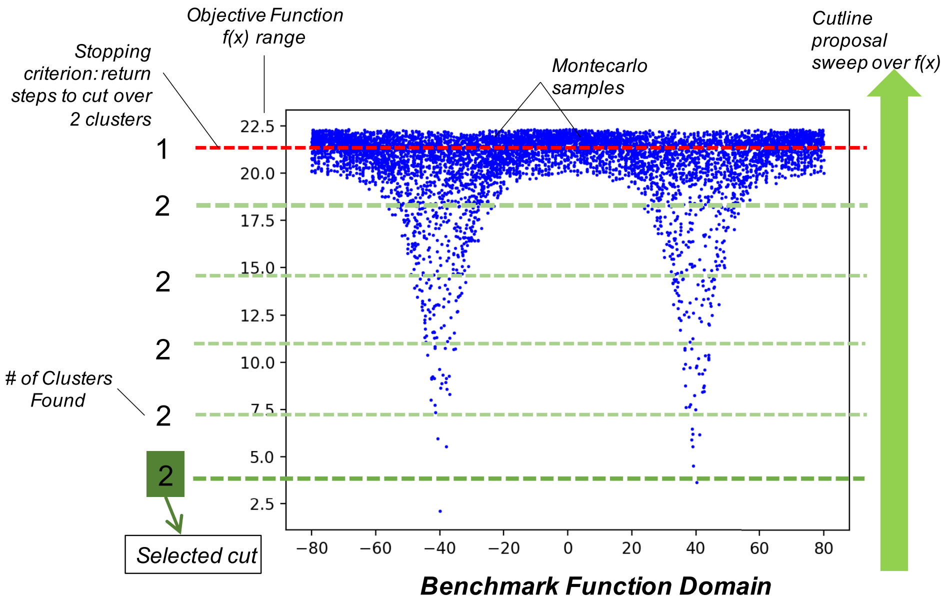


FIGURE 4. Cutline proposal over a Montecarlo map to reduce the search space into dense regions in a random benchmark function.

the number of samples needed to reach a solution, against the problem size is necessary to perform the evaluation.

The experiments were conducted using classical optimization benchmark functions whose number of objectives is known. Among the most common limitations of these functions is that some of them are defined with respect to only one or two parameters. Others are also non-separable, or non-linear functions, etc. Regarding the benchmarks, as listed in Table 1, they are seen as minimization problems in our matter. The quality of the solutions are overall above 97% with respect to the optimal, which is reported in the Objective Function Value $f(x)$ column. As evident from our observations, the ratio between the number of samples required to attain a solution and the problem size is exceptionally low. In the majority of instances we were able to achieve high-quality solutions, and in some cases, even the optimal solution. Nevertheless, out of all the cases we tested, only one in which reached the limit of 15% of samples regarding the problem size. This specific case pertains to the XinSheYang function, where the solution quality reached a remarkably 100%.

In addition to these tests, we have evaluated MCSA using the CEC17 benchmark problems for single-objective, real-parameter numerical optimization [10], which has helped us under-stand how to compare the performance of different meta-heuristics, as appears in [41], [42], and [43]. Table 2 reports the tested benchmark problems based on shifted, rotated, non-separable, highly ill-conditioned, and complex optimization benchmark functions and show the best results obtained from an average of 200 executions. Most of the functions we tested in two and ten dimensions and in an ample domain.

As seen in the boxplots in Figure 6 and Figure 7, the results obtained by MCSA were relevant overall. While in some functions, the optimum was found, in other functions promising results were achieved, indicating that the result values are more or less symmetrical. The difficulty of the functions is coupled with the size of the search space which scales exponentially when the number of dimensions is increased. Therefore, as the problem size grows exponentially, the possibilities of obtaining a homogeneously distributed random sample is incredibly difficult, causing the fitness function values to converge too soon, giving rise to inefficiencies and low opportunities of finding promising solutions.

Based on these results, we believe there is potential for enhancement. As a result, we are eager to pursue further testing with the aim of exploring opportunities to reduce the search space. This will enhance the algorithm's exploration processes, particularly when dealing with substantial problem sizes.

IV. SOLVING THE KNAPSACK PROBLEM THROUGH MCSA HEURISTIC METHOD

The Knapsack Problem (KP) is considered a combinatorial optimization problem, because it fulfills a series of properties that also make it particularly suitable to study. Binary KP or KP01 is an important modality of the classical Knapsack Problem and one of the most intensively analyzed discrete programming problems.

The reason for such interest derives from the fact that it may represent many practical situations. For example, in cutting stock, where you have to cut a steel plate into different pieces, or you have to determine the items that a warehouse can store to maximize its total value or maximize

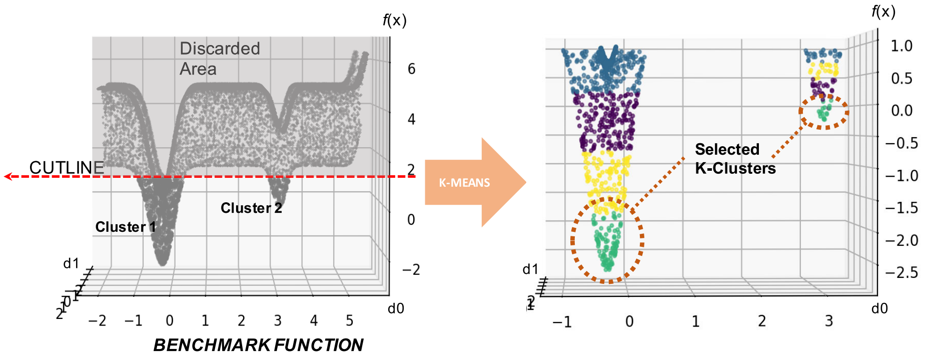


FIGURE 5. Clustered regions that are subsequently consolidated using K-means clustering, over a selected benchmark.

the benefit in investment allocation when there is only one restriction. In addition to this, in situations where there are load distribution problems (physical, electrical, etc.) or in the allocation of processors and data in distributed systems.

The Knapsack Problem also turns out to be an interesting choice to evaluate our MCSA methodology because of its intuitive and easy-to-understand statement and its importance among real-life decision-making processes in various fields. In addition, adapting this type of problem to our methodology adds another step. The previous work needed to adapt the parameters of the problem into the parameters of MCSA required to develop the strategy that included the decomposition of the problem in a mathematical formulation, comprising variables and parameters.

Our aim to solve the Knapsack Problem is to find a quality solution, while making an efficient use of the resources, especially regarding compute, which is directly related to the Exploration processes that involve sampling methods like MCSA, designed to re-explore feasible regions to improve the solution by performing iterations. As we know, the solution is contained within some combination of elements, for which, we must know the power-set, based on the number of elements. Thus, any combination within this set is a possible solution. To deal with this problem, we distribute the elements and its power-set along the search space that can be $n - dimensional$ and perform an ordering process. This type of arranging, does take into consideration all the possible combinations of elements.

The Knapsack Problem has the following statement: *Given a set of items, we determine the number of items included in the Knapsack so that the total weight is less than or equal to a given capacity. Each item has a weight and provides a benefit (or profit), regarding the problem itself.* The objective is for a supposed person to choose the elements that will allow him to maximize the benefit without exceeding the allowed capacity of the Knapsack.

Mathematically, we have:

$$W(x) = \sum_{i=1}^{n} x_i w_i = W \tag{1}$$

where W denotes the maximum capacity of the backpack, x would be the elements whose index numbering can vary from 1 to n. Concerning w_i and p_i, they represent the weight and value of element number 1, meaning that the sum of the weights, or the function $W(x)$ of the objects that we put in the backpack must not exceed its capacity, which is W. Regarding the restriction, it is given by:

$$Z(x) = \sum_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{x_i p_i} \tag{2}$$

where $Z(x)$ is the objective function (to maximize or minimize). In addition, a vector x that meets the restriction W in the formula is feasible if we have a maximum result in $Z(x)$. Then X is optimal, and other constraints can be added depending on the case, being able to find singular cases.

One of the essential steps in our methodology is to analyze how to adapt any problems data to the parameters of MCSA. Find the limitation rules, and how to sort data to be represented through a Montecarlo sample map. If we make a good parameterization of any problem, then the results will be better interpreted, and will enable us to find the relationship between the variables and be able to reuse what was previously learned.

Figure 8 shows a graphic representation of MCSA methodology applied to the Knapsack Problem. The Exploration phase comprises the sampling process, which is made throughout the search space. The ranges of such space are bounded by the calculus of the power-set. This is, $2^6 = 64$ Suppose we arrange a $2 - Dimensional$ search space. How do we distribute the elements?

To answer this question, we should keep in mind the cardinality of a set. If we distribute evenly, then we will put 3 elements per dimension, this is $|S| = 3$. Therefore, its power-set is 2^3, which is equal to 8 possible combinations. As seen in the same Figure 8, for this instance, the problem size remains equal to 64, due to the correspondence between the sets.

To enable identification of any element or a subset, a binary ID is given to every combination of elements in each

TABLE 1. Evaluation results using the classical benchmarks for optimization.

Benchmark properties				MCSA Results				
Function	Objective Function Value f(x)	Location of Global Optimum	Problem size (Psize)	# of Samples	# of Samples vs Psize (%)	Best result	Location of Best result	Quality of Solution (%)
Ackley	*f(x)=0*	(0, ..., 0)	1,048,576	87,000	8.3	0.0044	(0.00104, 0.00114)	100.00
Sphere	*f(x)=0*	(0, ..., 0)	1,048,576	24,300	2.3	0.0238	(0.11025, -0.10818)	98.97
Holder Table	*f(x)= -19.85*	(8.05502, 9.66459), (8.05502, -9.66459), (-8.05502, 9.66459), (-8.05502, -9.66459)	1,048,576	116,800	11.1	-19.2084	(8.05521, -9.66407), (8.05539, 9.66502), (-8.0563, -9.6647), (-8.05450, 9.66626)	96.72
Levi N.13	*f(x)=0*	(1, 1)	1,048,576	11,200	1.1	0.0194	(1.00392, 0.88739)	98.98
Griewank	*f(x)=0*	(0, ..., 0)	1,440,000	48,605	3.4	0.0235	(0.0, -9.0)	98.97
XinSheYang	*f(x)=0*	(0, ..., 0)	250,000	37,601	15.0	0.0000	(0.0, 0.0)	100.00

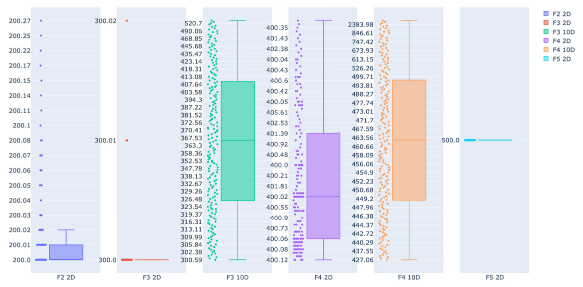


FIGURE 6. **Boxplot showing the results of CEC2017 benchmark tests.**

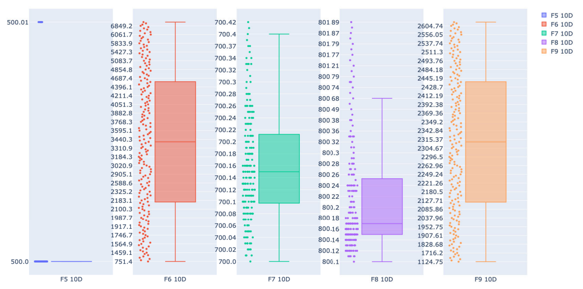


FIGURE 7. **Boxplot showing different set of results of CEC2017 benchmark tests.**

dimension, and its values 0 and 1 indicate whether an element enters the Knapsack or not. Thus, it is possible to know exactly which elements make up each combination, making it easier to interpret the solution. Once we have given every binary ID, we sort the elements according to the objective function $Z(x)$, and use the Montecarlo Index to place them in the Axes. This step is important, since it will provide continuity to the sampling process. Every sample will be surrounded by similar valued samples, ensuring MCSA not to analyze the entire map.

For larger instances, or when the number of elements increase, say we have to deal with a large input that involves many elements and variables. In this case, we use an $n - dimensional$ distribution approach to solve it. As it appears in Table 3, the input data can be distributed through the $n - dimensions$, for which, we can think of n-arrays that will contain subsets of elements, while the problem size remains the same. Here, we show a multi-dimensional arrangement of the elements from a specific Knapsack Problem instance.

After completing the ordering and sorting of elements along the Axes, we initiate the Exploration phase, as illustrated in Figure 9. This phase involves generating random samples, with the aim of achieving a uniform distribution through-out the search space. Each set corresponds to a sample, where the value is determined by the sum of the weights of elements selected from the respective subsets. It's crucial to emphasize that for a sample to be considered, the total weight sum must adhere to the problem's weight restriction (or not exceed the Knapsack's capacity). Any sample failing this condition will be discarded.

The Exploration phase transitions smoothly into Exploitation, employing a reduction strategy through the previously described "cutting process" along the $f(x)$ axis. This process eliminates redundant samples and facilitates density analysis, resulting in at least one density cluster, which is then grouped using K-means to identify feasible regions.

To illustrate the methodology, we applied it to solve the $EX01$ instance from Kreher and Stinson (available online). Before entering the Exploration phase, we experimented with different dimensional distributions. For instance, distributing 12 elements per dimension results in a staggering 4096 possible combinations for each dimension. However, when using a four-dimensional distribution with 6 elements per dimension, there are only 64 combinations per dimension. This approach significantly reduces the elements to be sorted in each dimension. Extending this to a 24-dimensional distribution, where each dimension contains only one element, can lead to a notable reduction in axis sorting time. This reduction can be advantageous in terms of resource utilization, especially when dealing with a large number of samples to be computed.

To tackle the "$EX01$" instance of the Knapsack problem in an 8-dimensional space, where we position 3 elements in each dimension, we end up with 8 combinations for each dimension. The overall problem size remains unchanged at $1.676\ E + 07$. Before transitioning into the Exploration Phase, a necessary step is to arrange the elements and their combinations along the axes. This setup enables MCSA to generate random samples and construct the Montecarlo map. Keep in mind the weight restriction, denoted as W. Samples failing to meet this restriction are discarded. Consequently, the map consists only of valid, weight-compliant samples.

TABLE 2. Results of the tests with the CEC2017 benchmark functions.

Benchmark	Number of dimensions	Domain	Problem Size	Average Number of samples	Percentage respect of Problem Size	Optimal solution	Best Heuristic Result
F2: Shifted and Rotated Zakharov Function	2	[-100,100]^2	4.00E+08	33,188	0.01	200	200.00
F3: Shifted and Rotated Rosenbrock's Function	2	[-100,100]^2	4.00E+08	34,790	0.01	300	300.00
F3: Shifted and Rotated Rosenbrock's Function	10	[-100,100]^10	1.02E+43	44,130	0.00	300	300.59
F4: Shifted and Rotated Rastrigin's Function	2	[-100,100]^2	4.00E+08	25,630	0.01	400	400.12
F4: Shifted and Rotated Rastrigin's Function	10	[-100,100]^10	1.02E+43	50,348	0.00	400	427.06
F5: Shifted and Rotated Schaffer's F7 Function	2	[-100,100]^2	4.00E+08	4,000	0.00	500	500.00
F5: Shifted and Rotated Schaffer's F7 Function	10	[-50,50]^10	1.00E+40	4,510	0.00	500	500.00
F6: Shifted and Rotated Lunacek Bi-Rastrigin's Function	10	[-100,100]^10	1.02E+33	32,360	0.00	600	751.40
F6: Shifted and Rotated Lunacek Bi-Rastrigin's Function	10	[-50,50]^10	1.00E+40	14,746	0.00	600	791.20
F7: Shifted and Rotated Non-Continuous Rastrigin's Function	10	[-50,50]^10	1.00E+40	6,629	0.00	700	700.00
F8: Shifted and Rotated Levy Function	10	[-50,50]^10	1.00E+40	15,800	0.00	800	800.10
F9: Shifted and Rotated Schwefel's Function	10	[-50,50]^10	1.00E+40	9,046	0.00	900	1,124.75

Table 4, shows how MCSA identifies and evaluates each combination. The binary ID plays a crucial role in computing the total Weight and Profit. In the same Table, we depict all the element combinations in Dimension one of the $EX01$ instance. The remaining elements are distributed across seven other Dimensions, each undergoing the same evaluation process. Subsequently, the combination that maximizes $Z(x)$ while adhering to the weight restriction W is chosen.

Figure 10 provides a visual representation of the process involved in ascending profit ordering and sorting of all subsets by the Montecarlo index. This arrangement is crucial for placing them along the axes. The Figure illustrates the final configuration of each dimension.

The sampling process, denoting the start of the Exploration Phase, leads to the construction of the initial sample map. Subsequently, focusing on reducing the sample map through the "cutting process" along $f(x)$. In our experience, the Knapsack Problem exhibits characteristics akin to a continuous function. As the process unfolds, the algorithm gradually transitions into the Exploitation Phase. Figure 11 visually represents this shift into the Exploitation phase, where clustering procedures are deployed to identify feasible regions.

Iterations help improve the solution by performing re-sampling over the feasible regions. In Figure 12, we represent how MCSA obtains the final solution. A single sample is made up from a selection of different combinations in each dimension. In the table, the marked squares represent the combinations, which are: $C3$ from Dim1, $C5$ from Dim2, $C4$ from Dim3, and so on until $C2$ from Dim8. Notice that dimension 7 returned $C0$, meaning that the empty set was selected.

The numerical results are presented in Table 5. In general, we have achieved an efficient solution from a computational perspective. The ratio of the number of samples required to obtain a solution to the problem size is approximately 1%. When compared to a published solution obtained using a branch and bound algorithm, our solution demonstrates high-quality, as discussed in Section V.

A. THE MULTI-OBJECTIVE KNAPSACK

To broaden the applicability of our methodology to other NP-type combinatorial optimization problems, we consider the Multi-objective Knapsack Problem, which serves as an extension of the original Knapsack Problem. In this variant, our objective is to simultaneously optimize two distinct objective functions: firstly, to maximize the aggregate weight of the elements selected for inclusion in the Knapsack, and secondly, to maximize the profit generated by these chosen elements, all while adhering to the weight restriction.

While the fundamental problem statement of the Knapsack remains consistent, our approach introduces a notable departure in terms of problem-solving strategy. In this context, we treat the weight restriction as both a constraint and an objective function, thereby transforming the problem into a bi-objective Knapsack Problem. Of course, the incorporation of supplementary constraints further amplifies the inherent complexity of the Knapsack Problem within this framework.

The field of multi-objective optimization is concerned with optimization problems involving multiple objectives, which, in general, are conflicting in the sense that no feasible solution exists that is optimal for each of them. Therefore, compromise solutions have to be found. A compromise solution is a

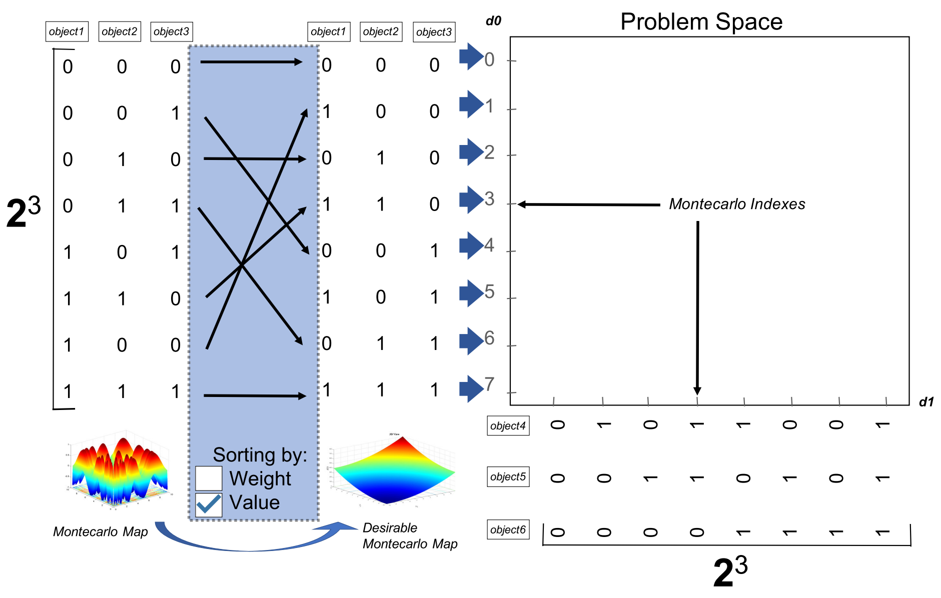


FIGURE 8. Graphic representation of MCSA heuristic methodology to solve the Knapsack Problem: The ordering of the elements in the search space.

solution that cannot be improved concerning one objective without reducing at least one value from the other objective function.

To tackle the Multi-objective Knapsack Problem, we must adapt our methodology to accommodate its specific parameters. As demonstrated earlier, by calculating the power-set, much like in the binary Knapsack case, we can determine the problem's size and subsequently define the boundaries of the $n - dimensional$ search space. The unique approach here involves arranging the elements, each with its corresponding weights and values, within the search space. However, the arrangement is based on the weighted sum of the objectives.

The weighted sum offers the possibility to weigh and combine multiple inputs, in order to create an integrated analysis and account for multiple factors by incorporating an *importancefactor*, which indicates the elements relative importance. In addition, it admits integer and floating-point values, allowing solving instances of the Knapsack problem in which objects can be split, which is another challenging variant.

The formal definition of the binary Multi-objective Knapsack problem with a single constraint has the following mathematical formulation:

$$max\ Z(x) = (z_1(x), z_2(x), \ldots, z_i(x), \ldots, z_m(x))$$
$$\textbf{Subject to}: \sum\nolimits_{j=1}^{n} w_j x_j \leq W$$
$$x_j \in [0, 1], j \in [1, 2, \ldots, n], i \in [1, 2, \ldots, m] \tag{3}$$

where $z_1(x) = \sum_{j=1}^{n} c_j^i x_j$ represent the i-th objective function; n is the number of elements; m is the number of objective function; c^i represents the profit of item j on criterion i; w_j is the weight/cost of item j; and W represents the overall Knapsack capacity. A decision variable $x_j = 1$, if and only if the element j is included in the Knapsack, otherwise $x_j = 0$. In this problem, we assume that W, c^i and w_j are positive integers and $w_j \leq W$ with $\sum_{j=1}^{n} w_j > W$ for all $j \in [1,2, \ldots, n]$ and $i \in [1,2, \ldots, m]$.

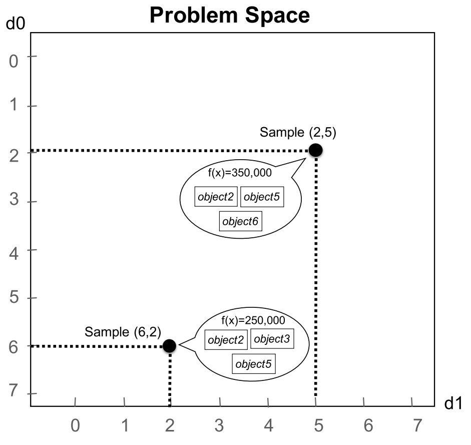


FIGURE 9. Sampling Process: Each sample undergoes evaluation in the function *f(x)* and must satisfy the restriction *W*.

The distribution of elements in the search space is made following the same idea of using an $n - dimensional$ array. Afterward, ordering is carried out according to the weighted sum of the elements. The user will decide whether to maximize one objective or another; or set fixed importance factors to each objective, in an attempt to optimize both objectives at the same time.

To solve this instance, we aimed at a compromise solution, or a trade-off, in which two objectives were considered and

TABLE 3. **Multi-dimensional distribution of elements in the search space for the Knapsack *EX*01 instance with 24 elements [44].**

Dimensional array for the Knapsack EX01 Problem												
ID element (n)	1	2	3	4	5	6	7	8	9	10	11	12
weight (i)	382,745	799,601	909,247	729,069	467,902	44,328	34,610	698,150	823,460	903,959	853,665	551,830
profit (i)	825,594	1,677,009	1,676,628	1,523,970	943,972	97,426	69,666	1,296,457	1,679,693	1,902,996	1,844,992	1,049,289
2-Dimensional	1st dim											
3-Dimensional	1st dim								2nd dim			
4-Dimensional	1st dim						2nd dim					
6-Dimensional	1st dim				2nd dim				3rd dim			
8-Dimensional	1st dim			2nd dim			3rd dim			4th dim		
12-Dimensional	1st dim		2nd dim		3rd dim		4th dim		5th dim		6th dim	
24-Dimensional	1st dim	2nd dim	3rd dim	4th dim	5th dim	6th dim	7th dim	8th dim	9th dim	10th dim	11th dim	12th dim
ID element (n)	13	14	15	16	17	18	19	20	21	22	23	24
weight (i)	610,856	670,702	488,960	951,111	323,046	446,298	931,161	31,385	496,951	264,724	224,916	169,684
profit (i)	1,252,836	1,319,836	953,277	2,067,538	675,367	853,655	1,826,027	65,731	901,489	577,243	466,257	369,261
2-Dimensional	2nd dim											
3-Dimensional	2nd dim				3rd dim							
4-Dimensional	3rd dim						4th dim					
6-Dimensional	4th dim				5th dim				6th dim			
8-Dimensional	5th dim			6th dim			7th dim			8th dim		
12-Dimensional	7th dim		8th dim		9th dim		10th dim		11th dim		12th dim	
24-Dimensional	13th dim	14th dim	15th dim	16th dim	17th dim	18th dim	19th dim	20th dim	21th dim	22th dim	23th dim	24th dim

TABLE 4. **Solving *EX*01: Combinations of the elements of dimension 1 (out of 8).**

Combinations (Cn)		Element	Elements Weight			Weight Combination	Profit Combination
Id	Binary representation		1	2	3		
C0	000	-	-	-	-	-	0
C1	001	1	382,745	-	-	382,745	825,594
C2	010	2	-	799,601	-	799,601	1,677,009
C3	011	1,2	382,745	799,601	-	1,182,346	2,502,603
C4	100	3	-	-	909,247	909,247	1,676,628
C5	101	1,3	382,745	-	909,247	1,291,992	2,502,222
C6	110	2,3	-	799,601	909,247	1,708,848	3,353,637
C7	111	1,2,3	382,745	799,601	909,247	2,091,593	4,179,231

were given different importance factors. The term of the objectives is normalized on the same scale, same as (4):

In multi-objective optimization, the concept of the Pareto front is commonly used since it allows making trade-offs within a set rather than considering the full range of every parameter. This way, we suppose that one of the objectives is to be minimized. The smaller the value of *objective*1 on the formula, the smaller the difference and the smaller the trade-off. Ordering with this method guarantees continuity.

MCSA attained a solution that complies with two objectives while fulfilling the restriction that will, in some parts, depend on the user's requirements, since the importance of the objectives will be defined by the user. The following section shows the discussion of the results that were obtained by MCSA when we solved three instances of the Knapsack Problem and a different variant of the Knapsack problem, where we pursue to optimize two objectives at the same time, reaching a trade-off, for which we applied its weighted sum. The detailed results were the best from 200 executions considering different importance factor for each one of the objectives.

V. DISCUSSION OF THE RESULTS

The evaluation of the MCSA heuristic method through the benchmark functions allowed us to have a better knowledge of the problems that we were going to deal with, making us capable of recognizing the patterns of the feasible regions. This was especially the case when we dealt with

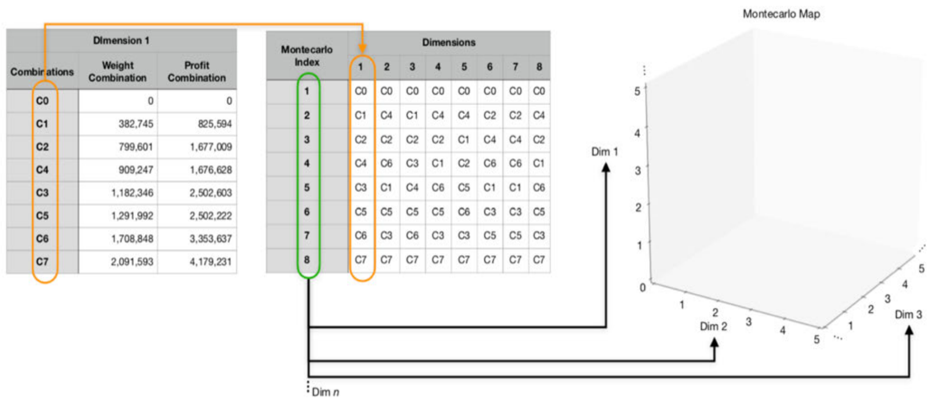
$$W1\frac{(Objective1 - Lower1)}{(Upper1 - Lower1)} + W2\frac{(Upper2 - Objective2)}{(Upper2 - Lower2)} + W3\frac{(Objective3 - Lower3)}{(Upper3 - Lower3)} \quad (4)$$

the Knapsack Problem, since the surface of the problem in which the parameters are unwrapped turned out to be similar to those functions with which we had validated.

The classical Knapsack Problem was fully adapted to our parameters through a heuristic approach. This approach involves distributing the elements across dimensions, calculating the power-set to obtain all combinations, and sorting them within the $n - dimensional$ space. The ordering of elements in this $n - dimensional$ search space ensures continuity, which is crucial for the sampling process. This initial step is essential for enabling MCSA to effectively solve the Knapsack Problem.

In Table 7, we present the details of the different Knapsack Problem instances that we solved. The Optimal Profit column shows the known optimal results for *EX*01 and *EX*02, which are open-source problems and are available online. In the case of all the synthetic *EX*'s, the optimal values were calculated following the method described in [45]. As these are randomly generated Knapsack instances with a vast number of elements, we set an optimal profit to evaluate the performance of the MCSA heuristic.

Overall, we obtained high-quality results for all instances. Table 8, showcases the optimal out-comes in the headers, contrasted with the results obtained through the MCSA heuristic method in the inner cells. Notably, our method demonstrated an impressively minimal ratio between the required number



Dimension 1		
Combinations	Weight Combination	Profit Combination
C0	0	0
C1	382,745	825,594
C2	799,601	1,677,009
C4	909,247	1,676,628
C3	1,182,346	2,502,603
C5	1,291,992	2,502,222
C6	1,708,848	3,353,637
C7	2,091,593	4,179,231

Montecarlo Index	Dimensions							
	1	2	3	4	5	6	7	8
1	C0	C0	C0	C0	C0	C0	C0	C0
2	C1	C4	C1	C4	C4	C2	C2	C4
3	C2	C2	C2	C2	C1	C4	C4	C2
4	C4	C6	C3	C1	C2	C6	C6	C1
5	C3	C1	C4	C6	C5	C1	C1	C6
6	C5	C5	C5	C5	C6	C3	C3	C5
7	C6	C3	C6	C3	C3	C5	C5	C3
8	C7	C7	C7	C7	C7	C7	C7	C7

FIGURE 10. Ordering the elements and sorting by the Montecarlo Index.

TABLE 5. MCSA results for "*EX*01" in 8 dimensions. Showing the best solution achieved.

Dimension	Combination		Elements	Elements Weight			Weight Combination	Profit Combination
	Id	Binary representation		1	2	3		
1	**C3**	011	1,2	382,745	799,601	-	1,182,346	2,502,603
2	**C5**	101	4,6	729,069	-	44,328	773,397	1,621,396
3	**C4**	001	9	823,460	-	-	823,460	1,679,693
4	**C3**	011	10,11	903,959	853,665	-	1,757,624	3,747,988
5	**C2**	010	14	-	670,702	-	670,702	1,319,836
6	**C1**	001	16	951,111	-	-	951,111	2,067,538
7	**C0**	000	-	-	-	-	0	0
8	**C2**	100	23	-	-	224,916	224,916	466,257
						Total Weight	6,383,556	
							Total Profit	13,405,311

TABLE 6. Results obtained for *EX*01 instance adapted to a multi-objective type Problem. Showing different importance factors.

Problem: EX01 Knapsack capacity (W) = 6,404,180 n = 24						
Problem size (Psize)	**# of Dimensions**	**Avg. # of Samples**	**# of Samples vs Psize (%)**	**Best Profit**	**Associated Weight sum**	**Importance Factor {0 -1}**
1.677E+7	2	12,902	0.08	13,524,340	6,397,001	Weight = 0 Profit = 1
1.677E+7	2	13,476	0.08	12,980,893	6,404,180	Weight = 1 Profit = 0
1.677E+7	12	12,174	0.07	13,518,963	6,398,128	Weight = 0 Profit = 1
1.677E+7	2	12,928	0.08	13,517,399	6,389,617	Weight = 0.5 Profit = 0.5
1.677E+7	12	11,994	0.07	13,509,694	6,390,248	Weight = 0.5 Profit = 0.5

TABLE 7. Detail of instances *EX*01 and *EX*02, and synthetic *EX*03 to *EX*06.

KP Instance (solved by branch bound)	#of Elements	Problem Size (Psize)	Capacity (W)	Optimal Profit (P)
EX01	24	1.676E+07	6,404,163	13,549,094
EX02	50	1.125E+15	850	7,534
EX03 (synthetic)	240	1.766E+72	64,041,630	135,789,553
EX04 (synthetic)	500	3.272E+150	2,394,376	2,477,959
EX05 (synthetic)	1000	2^1000	4,864,967	5,052,984
EX06 (synthetic)	2000	2^2000	9,860,929	10,173,777

of samples and the problem size. Furthermore, when examining the Axis sort time, it becomes evident that fewer elements per dimension result in reduced time consumption. This aspect can significantly impact the total execution time and, consequently, resource utilization, as shorter processing times typically indicate more efficient management of computational resources.

The quality of the solutions out of 30 full executions for each dimensional approach were in summary: above 98%

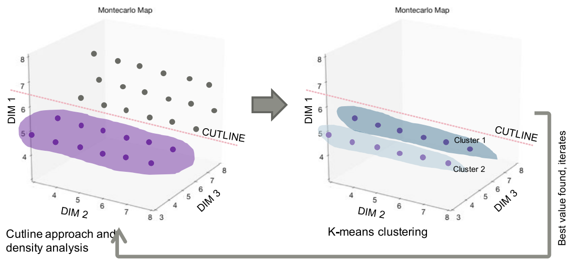


FIGURE 11. Entering Exploitation phase. a)Cut-line to enhance density analysis, b) K-means clustering and iterations.

for *EX*01, averaged 86% in *EX*02, and above 96% regarding *EX*03. As for *EX*04, *EX*05 and *EX*06 the quality reached 92.77%, 92.23% and 94.17% respectively. Additionally, it is worth emphasizing that MCSA demonstrates its efficacy in tackling high-dimensional challenges, despite the inherent complexity associated with such problems. This is apparent not only from the results achieved in benchmark function tests but also from its performance when addressing combinatorial optimization problems such as the Knapsack problem within a discretized sample space.

These results confer a distinct advantage when compared to other search algorithms that achieve optimal solutions through brute force, albeit at a substantial computational cost in terms of execution time.

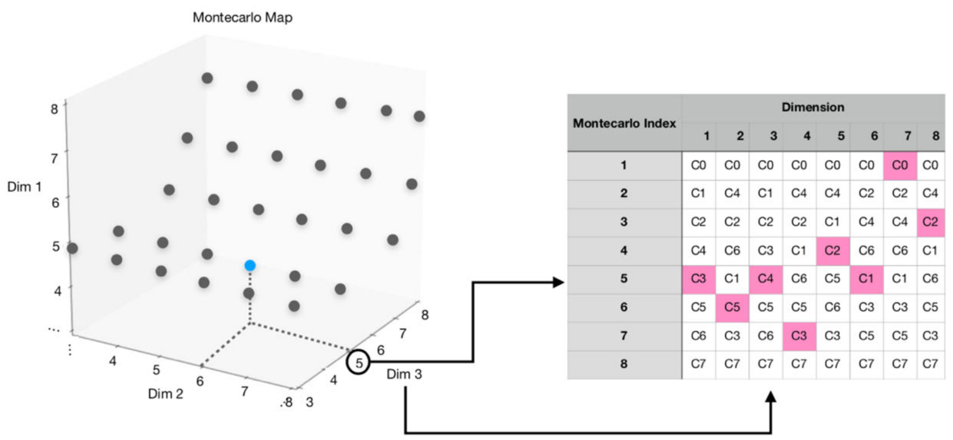
The case of the results obtained for the *EX*01 instance in 8 dimensions, they represent one of the best outcomes. We achieved a maximum profit of 13,436,707, which translates to a solution quality of 99.17% compared to the optimum. Additionally, this solution satisfies the weight constraint *W* with a total weight of 6,399,256, comfortably below the limit. From a computational perspective, it demonstrates high efficiency, as the ratio of required samples to problem size doesn't even reach 1%.

In the context of the *EX*'s synthetic instances, we would like to underscore two key considerations that underpin the rationale behind proposing these problems. Firstly, we sought to enhance problem complexity by augmenting the number

TABLE 8. Results obtained from the different Knapsack problem instances solved by MCSA.

EX01 - Knapsack capacity (W) = 6,404,163, n = 24 Optimal Profit (P) = 13,549,094							
# of Dimensions	**Problem size (Psize)**	**# of Samples**	**# of Samples vs Psize (%)**	**Axis Sort Time (sec)**	**Max Profit (P)**	**Associated Weight**	**Quality (%)**
2	1.68E+07	1,727	0.0102	0.0992	13,463,657	6,403,149	99.36
4	1.68E+07	1,644	0.0097	0.0030	13,461,329	6,402,653	99.35
6	1.68E+07	3,147	0.0187	0.0011	13,437,475	6,375,863	99.17
8	1.68E+07	7,004	0.0417	0.0009	13,436,707	6,399,256	99.17
12	1.68E+07	4,429	0.0263	0.0006	13,385,615	6,389,719	98.79
24	1.68E+07	146	0.0008	0.0000	13,300,418	6,392,566	98.16
EX02 - Knapsack capacity (W) = 850, n = 50, Optimal Profit (P) = 7,534							
# of Dimensions	**Problem size (Psize)**	**# of Samples**	**# of Samples vs Psize (%)**	**Axis Sort Time(sec)**	**Max Profit (P)**	**Associated Weight**	**Quality (%)**
2	1.13E+15	2,084	0.0000	1276.6851	6,242	805	82.85
5	1.13E+15	1,417	0.0000	0.0489	6,643	818	88.17
10	1.13E+15	2,856	0.0000	0.0023	6,962	810	92.40
25	1.13E+15	5,044	0.0000	0.0016	6,233	849	82.73
50	1.13E+15	524	0.0000	0.0011	6,445	772	85.54
EX03- Synthetic Knapsack capacity (W) = 64,041,630, n = 240, Optimal Profit (P) = 135,789,553							
# of Dimensions	**Problem size (Psize)**	**# of Samples**	**# of Samples vs Psize (%)**	**Axis Sort Time(sec)**	**Max Profit (P)**	**Associated Weight**	**Quality (%)**
20	1.77E+72	557	0.0000	1.1231	130,279,429	63,791,741	95.94
30	1.77E+72	1,622	0.0000	0.0868	130,589,231	63,981,629	96.17
40	1.77E+72	899	0.0000	0.0226	130,448,449	63,941,709	96.06
48	1.77E+72	1,148	0.0000	0.0162	130,824,332	63,923,428	96.34
60	1.77E+72	3,170	0.0000	0.0089	130,805,991	63,987,106	96.32
80	1.77E+72	172	0.0000	0.0081	130,011,304	63,741,250	95.74
EX04- Synthetic Knapsack capacity (W) = 2,394,376, n = 500, Optimal Profit (P) = 2,477,959							
# of Dimensions	**Problem size (Psize)**	**# of Samples**	**# of Samples vs Psize (%)**	**Axis Sort Time(sec)**	**Max Profit (P)**	**Associated Weight**	**Quality (%)**
50	3.272E+150	4,000	0.0000	1.7433	2,298,986	1,962,308	92.77
EX04- Synthetic Knapsack capacity (W) = 4,864,967, n = 1000, Optimal Profit (P) = 5,052,984							
# of Dimensions	**Problem size (Psize)**	**# of Samples**	**# of Samples vs Psize (%)**	**Axis Sort Time(sec)**	**Max Profit (P)**	**Associated Weight**	**Quality (%)**
100	2^1000	7,000	0.0000	3.5074	4,660,746	3,947,911	92.23
EX04- Synthetic Knapsack capacity (W) = 9,860,929, n = 2000, Optimal Profit (P) = 10,173,777							
# of Dimensions	**Problem size (Psize)**	**# of Samples**	**# of Samples vs Psize (%)**	**Axis Sort Time(sec)**	**Max Profit (P)**	**Associated Weight**	**Quality (%)**
200	2^2000	4,000	0.0000	6.9587	9,581,024	8,455,373	94.17

of elements, leading to a substantial growth in problem size. Given that the solution space encompasses the power-set of these elements, the distribution and organization of these elements across multiple dimensions become inherently challenging, thereby warranting the application of heuristic methods. Secondly, beyond the challenges posed by



Montecarlo Index	Dimension							
	1	2	3	4	5	6	7	8
1	C0	C0	C0	C0	C0	C0	C0	C0
2	C1	C4	C1	C4	C4	C2	C2	C4
3	C2	C2	C2	C2	C1	C4	C4	C2
4	C4	C6	C3	C1	C2	C6	C6	C1
5	C3	C1	C4	C6	C5	C1	C1	C6
6	C5	C5	C5	C5	C6	C3	C3	C5
7	C6	C3	C6	C3	C3	C5	C5	C3
8	C7	C7	C7	C7	C7	C7	C7	C7

FIGURE 12. A single sample is made of the combination of elements in 8Dimensions.

size, we also aimed to establish optimal solutions for these instances, facilitating meaningful comparisons. By defining optimal solutions, we enabled a robust assessment of the quality of our heuristic approach. As our results indicate, our heuristic approach has indeed demonstrated a commendable level of quality.

However, despite the success demonstrated in addressing synthetic instances, the MCSA methodology encounters typical limitations when tackling high-dimensional problems. Experimentation revealed that, with an increase in the number of dimensions, the complexity of the search space grows exponentially, giving rise to the well-known "curse of dimensionality." In n-dimensional problems, where the range is conserved, the sheer quantity of potential solutions escalates substantially, posing challenges to the efficacy of any heuristic algorithm. The exploration of the search space becomes more scattered and less focused, potentially resulting in a loss of precision and efficiency.

VI. CONCLUSION AND FUTURE WORK

This paper presents the heuristic MCSA, which is used to solve combinatorial optimization problems. It comprises two Phases: an Exploration Phase where the sampling processes are carried out in a $n - dimensional$ search space; and Exploitation Phase, where we seek to locate feasible regions, where the best values are located. The solution can be improved through iterations on bounded regions, re-exploring the space until a better solution is achieved.

The adapting work needed to bring the parameters into those of MCSA heuristic method is a preliminary task that will enable MCSA to solve optimization problems, making the distribution of the elements in a $n - dimensional$ search space. Furthermore, ordering and sorting of the elements is another essential step in the methodology, since it highlights the continuity in the sampling process.

Furthermore, we conducted an evaluation of MCSA using benchmark optimization functions. Encouraged by the promising outcomes, we applied our approach to tackle a practical optimization challenge, selecting the Knapsack Problem due to its significance in various domains, particularly in engineering and computing. Our results demonstrated high quality, with an average solution performance of 90% across all instances, as compared to exhaustive methods.

We extended our approach to address the multi-objective variant of the Knapsack Problem, aiming to optimize multiple objectives simultaneously. This evolution in the MCSA heuristic methodology highlights its adaptability to diverse problem types that share similarities with the Knapsack, such as resource allocation, production scheduling, and distribution processes. By acknowledging the limitations of existing approaches and identifying areas for improvement, we can assess the overall quality and enable meaningful comparisons. The scalability of the MCSA algorithm in high-dimensional scenarios is now a critical consideration for enhancing performance and maintaining relevance in the challenge of finding high-quality solutions while conserving a low ratio with the number of samples used for this purpose.

A significant step forward involves the parallelization of MCSA, allowing us to address another class of combinatorial optimization problems. This advancement gains particular significance in complex domains, such as healthcare, and decision-making in emergency healthcare settings. The potential of parallelization holds promise in enhancing the efficiency and applicability of MCSA, offering more effective solutions for urgent decision-making scenarios in the healthcare domain. As part of our ongoing efforts to improve the algorithm's performance, we are actively focusing on parallelization during the Exploration phase. This strategic emphasis aims to ensure the generation of a homogeneous sample distributed effectively across the search space, thereby enhancing efficiency and optimizing resource utilization in terms of compute.

ACKNOWLEDGMENT

Declaration of Generative AI and AI-assisted technologies in the writing process. Statement: During the preparation of

this work the author(s) used Grammarly in order to improve readability and language. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the content of the publication.

REFERENCES

[1] S. Balan, "Metaheuristics in optimization: Algorithmic perspective," OR-MS-Tomorrow INFORMS Mag., 2021, p. 29. Accessed: Sep. 25, 2022. [Online]. Available: https://www.informs.org/Publications/OR-MSTomorrow/Metaheuristics-in-Optimization-Algorithmic-Perspective

[2] A. Gaspar-Cunha, J. A. Covas, and J. Sikora, "Optimization of polymer processing: A review (Part I-extrusion)," *Materials*, vol. 15, no. 1, p. 384, Jan. 2022.

[3] F. S. Gharehchopogh and H. Gholizadeh, "A comprehensive survey: Whale optimization algorithm and its applications," *Swarm Evol. Comput.*, vol. 48, pp. 1–24, Aug. 2019.

[4] A. S. Varde, S. Ma, M. Maniruzzaman, D. C. Brown, E. A. Rundensteiner, and R. D. Sisson, "Comparing mathematical and heuristic approaches for scientific data analysis," *Artif. Intell. Eng. Design, Anal. Manuf.*, vol. 22, no. 1, pp. 53–69, Feb. 2008.

[5] B. Abdollahzadeh and F. S. Gharehchopogh, "A multi-objective optimization algorithm for feature selection problems," *Eng. Comput.*, vol. 38, no. 3, pp. 1845–1863, Aug. 2022.

[6] F. S. Gharehchopogh, I. Maleki, and Z. A. Dizaji, "Chaotic vortex search algorithm: Metaheuristic algorithm for feature selection," *Evol. Intell.*, vol. 15, no. 3, pp. 1777–1808, Sep. 2022.

[7] H. Mohmmadzadeh and F. S. Gharehchopogh, "An efficient binary chaotic symbiotic organisms search algorithm approaches for feature selection problems," *J. Supercomput.*, vol. 77, no. 8, pp. 9102–9144, Aug. 2021.

[8] N. Rahnema and F. S. Gharehchopogh, "An improved artificial bee colony algorithm based on whale optimization algorithm for data clustering," *Multimedia Tools Appl.*, vol. 79, nos. 43–44, pp. 32169–32194, Nov. 2020.

[9] D. Pršić, N. Nedić, and V. Stojanović, "A nature inspired optimal control of pneumatic-driven parallel robot platform," *Proc. Inst. Mech. Eng., C, J. Mech. Eng. Sci.*, vol. 231, no. 1, pp. 59–71, Jan. 2017.

[10] L. Abualigah, D. Yousri, M. A. Elaziz, A. A. Ewees, M. A. A. Al-qaness, and A. H. Gandomi, "Aquila optimizer: A novel meta-heuristic optimization algorithm," *Comput. Ind. Eng.*, vol. 157, Jul. 2021, Art. no. 107250.

[11] V. Faber, "Clustering and the continuous k-means algorithm," *Los Alamos Sci.*, vol. 22, p. 67, Jan. 1994.

[12] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations*. Boston, MA, USA: Springer, 1972, pp. 85–103.

[13] T. Pradhan, A. Israni, and M. Sharma, "Solving the 0–1 knapsack problem using genetic algorithm and rough set theory," in *Proc. IEEE Int. Conf. Adv. Commun., Control Comput. Technol.*, May 2014, pp. 1120–1125.

[14] L. M. Antonio and C. A. C. Coello, "Coevolutionary multiobjective evolutionary algorithms: Survey of the state-of-the-art," *IEEE Trans. Evol. Comput.*, vol. 22, no. 6, pp. 851–865, Dec. 2018.

[15] L. Bayas-Jiménez, F. J. Martínez-Solano, P. L. Iglesias-Rey, and D. Mora-Meliá, "Search space reduction for genetic algorithms applied to drainage network optimization problems," *Water*, vol. 13, no. 15, p. 2008, Jul. 2021.

[16] K. Hussain, M. N. M. Salleh, S. Cheng, and R. Naseem, "Common benchmark functions for metaheuristic evaluation: A review," *Int. J. Inform. Vis.*, vol. 1, nos. 4–2, pp. 218–223, Nov. 2017.

[17] G. Wu, R. Mallipeddi, and P. N. Suganthan, "Problem definitions and evaluation criteria for the CEC 2017 competition on constrained real-parameter optimization," Nat. Univ. Defense Technol., Changsha, China, Kyungpook Nat. Univ., Daegu, South Korea and Nanyang Technol. Univ., Singapore, Tech. Rep., Oct. 2017. [Online]. Available: https://github.com/P-NSuganthan/CEC2017-BoundContrained/blob/master/Definitions%20of%20%20CEC2017%20benchmark%20suite%20final%20version%20updated.pdf

[18] M. Harita, A. Wong, D. R. del Rosario, and E. L. Fadón, "Evaluation of a heuristic search algorithm based on sampling and clustering," in *Proc. 9th Conf. Cloud Comput., Big Data Emerg. Topics*, Jun. 2021, pp. 55–59.

[19] L. Bianchi, M. Dorigo, L. M. Gambardella, and W. J. Gutjahr, "A survey on metaheuristics for stochastic combinatorial optimization," *Natural Comput.*, vol. 8, no. 2, pp. 239–287, Jun. 2009.

[20] D. Yan and H. Mukai, "Stochastic discrete optimization," *SIAM J. Control Optim.*, vol. 30, no. 3, pp. 594–612, May 1992.

[21] S. Andradóttir, "A method for discrete stochastic optimization," *Manage. Sci.*, vol. 41, no. 12, pp. 1946–1961, Dec. 1995.

[22] W-B. Gong, Y-C. Ho, and W. Zhai, "Stochastic comparison algorithm for discrete optimization with estimation," in *Proc. 31st IEEE Conf. Decis. Control*, Dec. 1992, pp. 795–800.

[23] L. Shi and S. O'lafsson, "Nested partitions method for stochastic optimization," *Methodol. Comput. Appl. Probab.*, vol. 2, no. 3, pp. 271–291, Sep. 2000.

[24] T. Eftimov and P. Korošec, "A novel statistical approach for comparing meta-heuristic stochastic optimization algorithms according to the distribution of solutions in the search space," *Inf. Sci.*, vol. 489, pp. 255–273, Jul. 2019.

[25] P. A. Vikhar, "Evolutionary algorithms: A critical review and its future prospects," in *Proc. Int. Conf. Global Trends Signal Process., Inf. Comput. Commun. (ICGTSPICC)*, Dec. 2016, pp. 261–265.

[26] H. Zhang, J. Sun, T. Liu, K. Zhang, and Q. Zhang, "Balancing exploration and exploitation in multiobjective evolutionary optimization," *Inf. Sci.*, vol. 497, pp. 129–148, Sep. 2019.

[27] M. Crepinšek, S. H. Liu, and M. Mernik, "Exploration and exploitation in evolutionary algorithms: A survey," *ACM Comput. Surv.*, vol. 45, no. 3, pp. 1–33, Jul. 2013.

[28] K. C. Tan, S. C. Chiam, A. A. Mamun, and C. K. Goh, "Balancing exploration and exploitation with adaptive variation for evolutionary multi-objective optimization," *Eur. J. Oper. Res.*, vol. 197, no. 2, pp. 701–713, Sep. 2009.

[29] B. N. Örnek, S. B. Aydemir, T. Düzenli, and B. Özak, "A novel version of slime mould algorithm for global optimization and real world engineering problems," *Math. Comput. Simul.*, vol. 198, pp. 253–288, Aug. 2022.

[30] J. Cai and G. Thierauf, "Evolution strategies for solving discrete optimization problems," *Adv. Eng. Softw.*, vol. 25, nos. 2–3, pp. 177–183, Mar. 1996.

[31] J.-H. Kim and H. Myung, "Evolutionary programming techniques for constrained optimization problems," *IEEE Trans. Evol. Comput.*, vol. 1, no. 2, pp. 129–140, Jul. 1997.

[32] B. Abdollahzadeh, F. S. Gharehchopogh, and S. Mirjalili, "African vultures optimization algorithm: A new nature-inspired metaheuristic algorithm for global optimization problems," *Comput. Ind. Eng.*, vol. 158, Aug. 2021, Art. no. 107408.

[33] B. N. Örnek, S. B. Aydemir, T. Düzenli, and B. Özak, "A novel version of slime mould algorithm for global optimization and real world engineering problems: Enhanced slime mould algorithm," *Math. Comput. Simul.*, vol. 198, Aug. 2022, pp. 253–258.

[34] E. Cabrera, M. Taboada, M. L. Iglesias, F. Epelde, and E. Luque, "Optimization of healthcare emergency departments by agent-based simulation," *Proc. Comput. Sci.*, vol. 4, pp. 1880–1889, Jan. 2011.

[35] W. Gong, L. Pang, J. Wang, M. Xia, and Y. Zhang, "A social-aware K means clustering algorithm for D2D multicast communication under SDN architecture," *Int. J. Electron. Commun.*, vol. 132, Apr. 2021, Art. no. 153610.

[36] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. 2nd Int. Conf. Knowl. Discovery Data Mining*, vol. 96, Aug. 1996, pp. 226–231.

[37] S. Raj and D. Ghosh, "Improved and optimal DBSCAN for embedded applications using high-resolution automotive radar," in *Proc. 21st Int. Radar Symp. (IRS)*, Oct. 2020, pp. 343–346.

[38] K. Mardani and K. Maghooli, "Enhancing retinal blood vessel segmentation in medical images using combined segmentation modes extracted by DBSCAN and morphological reconstruction," *Biomed. Signal Process. Control*, vol. 69, Aug. 2021, Art. no. 102837.

[39] F. Fouedjio, "Clustering of multivariate geostatistical data," *Wiley Interdiscipl. Rev., Comput. Statist.*, vol. 12, no. 5, Sep. 2020, Art. no. e1510.

[40] A. Gaudiani, E. Luque, P. García, M. Re, M. Naiouf, and A. Di Giusti, "Computing, a powerful tool for improving the parameters simulation quality in flood prediction," *Proc. Comput. Sci.*, vol. 29, pp. 299–309, Jan. 2014.

[41] A. Tangherloni, L. Rundo, and M. S. Nobile, "Proactive particles in swarm optimization: A settings-free algorithm for real-parameter single objective optimization problems," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2017, pp. 1940–1947.

[42] M. S. Nobile, A. Tangherloni, L. Rundo, S. Spolaor, D. Besozzi, G. Mauri, and P. Cazzaniga, "Computational intelligence for parameter estimation of biochemical systems," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2018, pp. 1–8.

[43] A. Tangherloni, S. Spolaor, P. Cazzaniga, D. Besozzi, L. Rundo, G. Mauri, and M. S. Nobile, ''Biochemical parameter estimation vs. benchmark functions: A comparative study of optimization performance and representation design,'' *Appl. Soft Comput.*, vol. 81, Aug. 2019, Art. no. 105494.

[44] M. Harita, A. Wong, D. Rexachs, and E. Luque, ''KP01 solved by an n-dimensional sampling and clustering heuristic,'' in *Proc. 22nd Int. Conf. Comput. Sci.*, vol. 13351. London, U.K.: Springe, Jun. 2022, pp. 229–236.

[45] E. Luque, A. Ripoll, and J. J. Ruz, ''Dynamic microprogramming in computer architecture redefinition,'' *Euromicro Newslett.*, vol. 6, no. 2, pp. 98–103, Mar. 1980.

MARIA HARITA is currently pursuing the Ph.D. degree with Universitat Autònoma de Barcelona, Spain. She was a collaborating teacher in subjects, such as distributed systems. Her research interests include algorithm design, heuristic methodologies, and optimization-related techniques. Her research is supported by the Spanish government.

ALVARO WONG is currently an Associate Researcher with the Department of Computer Architecture and Operating Systems, Universitat Autònoma de Barcelona (UAB), Spain. He has worked in performance prediction of HPC applications in the ITEA 2 European Project, research centers, and industries. He has coauthored a total of 20 full-reviewed technical papers in journals and conference proceedings.

REMO SUPPI is currently an Associate Professor with the School of Engineering, Universitat Autònoma de Barcelona (UAB), and a Researcher with the High-Performance Computing for Efficient Applications and Simulation Research Group (HPC4EAS). His research interests include HPC, high-performance simulation on ABM models, and big data processing technologies. He is the coauthor of more than 60 scientific papers on the above topics in international journals and conferences and has been a supervisor of seven Ph.D. theses.

DOLORES REXACHS is currently an Associate Professor with the Department of Computer Architecture and Operating Systems, Universitat Autònoma de Barcelona (UAB), Spain. She has been a supervisor of ten Ph.D. theses and an invited lecturer with Universities in Argentina, Brazil, Chile, and Paraguay. She has coauthored more than 70 fully reviewed technical papers in journals and conference proceedings. Her research interests include parallel computer architecture, parallel I/O sub-systems, fault tolerance in parallel computers, and tools to evaluate, predict, and improve performance in parallel computers.

EMILIO LUQUE is currently an Emeritus Professor with the Department of Computer Architecture and Operating Systems, Universitat Autònoma de Barcelona (UAB), Spain. He is an invited lecturer at universities in the USA, South America, Europe, and Asia; a keynote speaker at several conferences; and a leader in several research projects funded by the European Union (EU), the Spanish government, and different industries. He has supervised 20 Ph.D. theses and coauthored more than 230 technical papers in journals and conference proceedings. His major research interests include parallel and distributed simulation, performance prediction and efficient management of multicluster-multicore systems, and fault tolerance in parallel computers.