KP01 Solved by an n-Dimensional Sampling and Clustering Heuristic

Maria Harita(✉), Alvaro Wong, Dolores Rexachs, and Emilio Luque

Universidad Autonoma de Barcelona, 08193 Barcelona, Spain
MariaDeLosAngeles.Harita@autonoma.cat,
{alvaro.wong,dolores.rexachs,emilio.luque}@uab.es

Abstract. In the field of optimization, NP-Hard problems play an important role concerning its real-world applications, such as resource allocation, scheduling, planning, logistics, etc. In this paper, we propose a heuristic search algorithm based on Montecarlo along with a clustering strategy that analyzes density and performs k-means partitions to solve the classic binary Knapsack Problem (KP01). Our heuristic method, which was designed to solve combinatorial optimization problems, has evolved and can adapt to other optimization problems, such as the KP01 that can be organized in an n-Dimensional search space. Regarding the methodology, we substantially reduced the search space while the areas of interest were located in the clustering stage, which brings us closer to the best solutions. After the experiments, we obtained a high-quality solution, which resulted in an average success rate of above 90%.

Keywords: Heuristic method · Clustering · 01 Knapsack Problem

1 Introduction

In optimization, we set the objectives when looking for the best possible configuration of a set of variables. We can deal with these problems by discretizing the variables or taking the real values. Thus, each optimization problem is specified by defining the possible solutions (or states) and a general objective. The classical optimization paradigm is understood as to how the solution identifies by enumeration or differential calculus, the existence of an assumed (unique) solution, or the convergence of classical optimization methods for the solution of the corresponding first-order conditions.

Probabilistic distribution methods such as Montecarlo offer flexible forms of approximation, with some advantages regarding cost. In this sense, metaheuristics are high-level algorithms that are capable of determining a sufficiently

This research has been supported by the Agencia Estatal de Investigacion (AEI), Spain and the Fondo Europeo de Desarrollo Regional (FEDER) UE, under contract PID2020-112496GB-I00 and partially funded by the Fundacion Escuelas Universitarias Gimbernat (EUG).

D. Groen et al. (Eds.): ICCS 2022, LNCS 13351, pp. 229–236, 2022.
https://doi.org/10.1007/978-3-031-08754-7_31

satisfactory (almost optimal) solution for an approximate optimization problem, while other approaches use similarity.

We propose a heuristic algorithm that works in conjunction with clustering techniques based on [1]. The Montecarlo K-Means (MCKM) method was improved and evaluated with the optimization benchmark functions [4], which are high-Dimensional problems in a large search space. In all the benchmark cases analyzed, we came up with the improved heuristic Montecarlo Density K-Means (MCDKM), obtaining promising results comparable with those in the literature.

For this paper, our proposal goes a step further in solving different types of combinatorial optimization problems where search processes are involved. We have selected the Knapsack Problem (KP), which seeks to maximize the profits provided by a selection of elements to enter a Knapsack. The objective function would be defined as the total profit of the selected objects. KP is a classic combinatorial optimization problem with various applications in different industries, such as resource allocation, tasks, resource management/scheduling, or energy allocation management. KP is also part of a historical list of NP-Complete problems elaborated by Richard Karp [5].

Martello [8], on the background of this classic NP-Hard problem, offers an extensive review of the many effective heuristics to solve the KP. For example, a population-based approach such as incremental learning algorithm based on a greedy strategy in [7]. In addition, other different heuristic techniques, such as [3,6,9] are proposed to provide an acceptable solution to this problem.

Regarding the contribution of our heuristic method MCDKM, it has been validated that it can find feasible solutions. In the methodology, we will see how the partitions in n-dimensions of the search space may reduce the ordering times of the axes. The results section compares the solution quality found when varying the Dimensional arrays that organize the data, ranging from bi-dimensional to more than 50-dimension arrangement.

The paper organizes as follows. Section 2 describes the MCDKM heuristic method. Section 3 introduces the Knapsack problem into the MCDKM method, Sect. 4 details the implementation and the results, and finally, Sect. 5 presents the conclusions and future work.

2 Description of the MCDKM Heuristic Method

MCKM heuristic method [1] was initially developed to solve a particular combinatorial optimization problem. Our proposal goes a step further by adding a previous step. This is, we prepare the problem by parameterizing it and organizing the search in an n-dimensional space. Furthermore, we also perform a density analysis of the generated map. Bringing the problem into our parameters will contribute to the solution's efficiency since, first, the search space can be significantly reduced. Second, the ordering times of the axes are improved, even when sorting them in a more significant number of dimensions. All this finally translates into a general improvement that contributes to increasing the

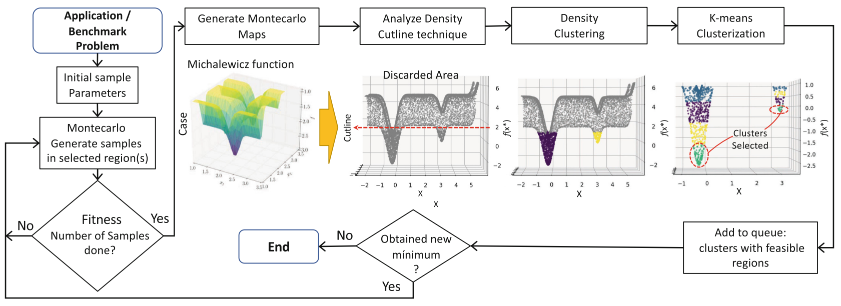

Application / Benchmark Problem
Initial sample Parameters
Montecarlo Generate samples in selected region(s)
Fitness Number of Samples done?
No
Yes
Generate Montecarlo Maps
Michalewicz function
Case
Analyze Density Cutline technique
Discarded Area
Cutline
Density Clustering
K-means Clusterization
Clusters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

Fig. 1. Flowchart showing the redesigned MCKM heuristic algorithm, highlighting its stages and showing benchmarks as a solution example.

quality of the solution and a good ratio between the size of the problem and the analyzed sample.

The procedure is shown in detail in Fig. 1. First, with Montecarlo, we generated a uniform sample that formed a map that is distributed through the whole search space. Then, the sample is reduced through the density analysis (for which we use the DBScan [2] algorithm), sweeping data to discard the sample that is not worth analyzing while keeping the remaining areas to perform another clustering process. Such clustering is made with k-means, which is now able to correctly partition the sample to group the feasible solutions into clusters for its selection.

The now more robust MCDKM heuristic method was able to find the best solution values, which turned out to be very similar and comparable to the optimum solution of the benchmark functions. The success rate was between 97% and 99%, as seen in the results from [4].

3 The Knapsack Problem into the MCDKM Method

We are bringing the Knapsack Problem KP01 into our heuristic method to find a comparable efficient solution. This combinatorial optimization problem is sometimes exemplified as: 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 a Knapsack. The problem can be stated as a problem of vectors $X = (x_1, x_2, x_3, ...x_n)$, which have components of zeros and ones, shown in Eq. 1, and have at most the restriction of the objective function *Z(x)*, as seen in Eq. 2. Mathematically, we have the following:

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

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

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 the i element, meaning that the sum of the weights must not exceed the knapsack capacity, which is W. Now, *Z(x)* is the objective function (maximize or minimize). In addition, a vector x in Eq. 2 that meets the constraint W is feasible if we have a maximum result in *Z(x)*.

Table 1. Dataset EX01: Knapsack capacity (W) = 6,404,180 and profit (P) = 13,549,094 elements n(i)=24

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

Analyzing these statements is one of the essential steps in the methodology to adapt the parameters. First, we find the relationship between the variables to sort the data into a Montecarlo map.

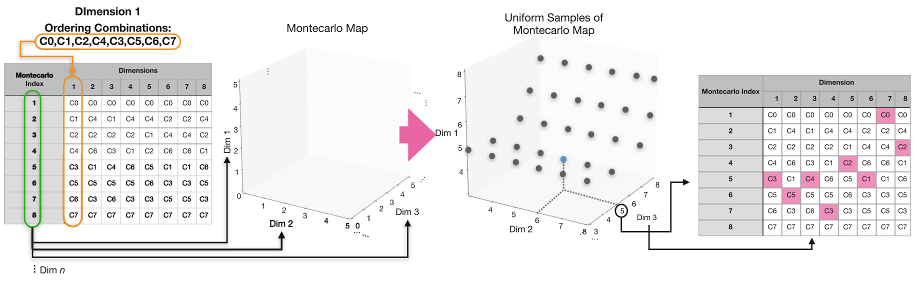
Since this KP modality is a decision problem, our strategy considers the elements as belonging to different dimensions. This is, we divide the search space into multiple dimensions distributing the elements among them, and subsequently ordering according to all the possible combinations between them and identifying them with an ID.

We selected two available open-source datasets, EX01 from Kreher and Stinson which was checked with a branch and bound (B&B) algorithm. EX02 is a dataset available in the ORLibrary. EX03 dataset was based on EX01.

Table 1 shows a Dimensional array we designed for EX01. As seen, for a 2-D search space, 12 elements will be placed in each dimension, and the ordering is made out of all the possible combinations between these 12 elements. In the case of a 24-D search space, each element belongs to a dimension. Ordering the elements should be more efficient, which could be taken as an advantage in terms of resource use.

Table 2. Detail of the possible combinations (C) in the 1st-dim. of an 8-dim array.

Combinations (Cn)			Elements weight				
Id	Binary representation	Element	1	2	3	Weight combination	Profit combination
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



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

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

Fig. 2. Establishing a Montecarlo index for each combination in every dimension. Every sample fulfills the restriction in an 8-Dimensional search space.

Table 3. Solution found by the heuristic method for a 8-Dimensional search space.

	Combination			Elements weight				
Dimension	Id	Binary representation	Elements	1	2	3	Weight combination	Profit combination
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

The Montecarlo map generates by crossing data between the combinations of the elements, which must not exceed the restriction (Eq. 1) or the value is rejected. The uniform map is formed with the samples that meet the weight constraint. The binary code serves as a coordinate representation of such combinations since there are only two possible states. Each digit position represents whether an element will be packed or not, so we know what elements are within each coordinate. Afterward, ordering the axes is carried out in ascending order of best profit. In Table 2 we show how ordering a dimension takes place.

In Fig. 2, we illustrate how we define the map by establishing a Montecarlo index and referring to the position of each combination of elements in each dimension, which returns an integer when generating random samples.

After creating the Montecarlo map, we seek to reduce the search space. Now, the density clustering algorithm can be applied directly to this map, but it has to be adjusted to work efficiently.

We make such adjustment by cutting the map along the objective function axis. Cutting the map helps improve the clustering, and we can avoid analyzing the complete map. Sweeping of the data must occur, and then the stopping criterion is activated when more than 1 cluster is found. We aim for the largest number of clusters since it interprets as many feasible areas. Now, the rest of the sample can be discarded, narrowing the search space.

The results are detailed in Table 3, where the list of combinations found by the heuristic results in a MaxProfit (P) = 13,405,311 fulfilling the restriction since the sum of the weight (W) = 6,383,556.

Table 4. Shows the results for the EX01, EX02 and EX03.

Problem	# of dimensions	Time (sec)	Problem size	Axis sort tme (sec)	# Samples	Max profit result	Weight result
EX01 - Knapsack capacity =	2	0.694	1.676E+07	0.0992	1,727	13,463,657	6,403,149
6,404,163 n = 24	4	0.538	1.676E+07	0.0030	1,644	13,461,329	6,402,653
Fitness = 1E−01	6	0.995	1.676E+07	0.0011	3,147	13,437,475	6,375,863
Optimal profit (P) = 13,549,094	8	2.322	1.676E+07	0.0009	7,004	13,436,707	6,399,256
with a weight (w) = 6,402,560	12	2.048	1.676E+07	0.0006	4,429	13,385,615	6,389,719
Time required = 1.2 s	24	0.432	1.676E+07	0.0010	146	13,300,418	6,392,566
EX02 - Knapsack capacity = 850 n = 50	2	1,277.291	1.125E+15	1276.6851	2,084	6,242	805
Fitness = 1E−01	5	2.043	1.125E+15	0.0489	1,417	6,643	818
Optimal profit (P) = 7,534	10	2.616	1.125E+15	0.0023	2,856	6,962	810
with a weight (w) = 850	25	3.903	1.125E+15	0.0016	5,044	6,233	849
Time required = 0.9 s	50	0.822	1.125E+15	0.0011	524	6,445	772
EX03- Synthetic Knapsack capacity = 64,041,630 n = 240	20	3.302	1.766E+72	1.1231	557	130,279,429	63,791,741
Fitness = 1E−01	30	2.127	1.766E+72	0.0868	1,622	130,589,231	63,981,629
Optimal profit = 135,789,553	40	1.088	1.766E+72	0.0226	899	130,448,449	63,941,709
with a weight (w) = 64,041,800	48	1.873	1.766E+72	0.0162	1,148	130,824,332	63,923,428
Time required = 13,057.79 s	60	5.010	1.766E+72	0.0089	3,170	130,805,991	63,987,106
	80	0.469	1.766E+72	0.0081	172	130,011,304	63,741,250

4 Implementation and Empirical Results

The headers of Table 4 detail the best solutions found for each data-set of the Knapsack Problem, such as the capacity *(W)*, the number of elements *(n)*, the fitness used, and the Optimal Profit *(P)*, obtained with a branch and bound (B&B) algorithm. The rest of the Table represents the best results obtained by the MCDKM heuristic out of 30 executions for each space partition.

As expected, as the number of dimensions increased, the Axis sort time decreased when it required fewer elements, thus decreasing the total time until finding a solution. Therefore, it can be considered an advantage against other types of search algorithms capable of finding the optimal but entailing a high computational cost (like B&B). Also, as seen in Table 4, we emphasize the ratio between the number of samples and the size problem. As a result, the total samples needed to find an efficient solution was less than 1% in all cases.

Regarding EX03, we increased the complexity to compare us with other algorithms. In this case, we used branch and bound. We verified that, as the complexity of the problem increases, the execution time might also increase, influencing the prediction quality. Nevertheless, the solution maintained a prediction quality above 95% in less than 1 s (when partitioned into 80 dimensions). In contrast, the execution time was considerably longer even though the branch and bound algorithm obtained the optimal solution.

Therefore, heuristic search methods, especially ours that perform stochastic optimization, provide a good way to solve large combinatorial problems and offer a high-quality solution.

5 Conclusions and Future Work

The Knapsack Problem was entirely adapted and brought into our parameters. As a result, we found an efficient solution and achieved better results in the execution time when we increased the no. of partitions of the search space. Furthermore, we significantly reduced the number of samples needed to reach the best solutions compared to the full sample (which is the problem size) in all cases. Space ordering and sorting is an essential step in the methodology.

As we present in this paper, the MCDKM heuristic search algorithm proved some of its advantages, such as its applicability in combinatorial optimization problems. The MCDKM method has the potential to solve any KP01 using an n-Dimensional search space approach obtaining an efficient, high-quality solution with a success rate above 90% average. Solving the KP01, grants MCDKM the capability to deal with a range of other combinatorial optimization problems, such as resource allocation, production scheduling, distribution processes, etc.

Our heuristic model is constantly improving and considering more factors, for example, the relationship among multiple items. In addition, our model is expected to solve multi-objective and multi-Dimensional optimization problems. We believe that parallelization of the method would provide a fast convergence with the improvement in the execution times.

References

1. Cabrera, E., Taboada, M., Iglesias, M.L., Epelde, F., Luque, E.: Optimization of healthcare emergency departments by agent-based simulation. Procedia Comput. Sci. **4**, 1880–1889 (2011)
2. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, KDD 1996, pp. 226–231. AAAI Press (1996)
3. Gao, Y., Zhang, F., Zhao, Y., Li, C.: Quantum-inspired wolf pack algorithm to solve the 0–1 knapsack problem. Math. Probl. Eng. **2018**, 1–10 (2018)
4. Harita, M., Wong, A., Rexachs, D., Luque, E.: Evaluation of the quality of the "Montecarlo plus k-means" heuristics using benchmark functions. In: Short Papers of the 8th Conference on Cloud Computing Conference, Big Data & Emerging Topics (JCC-BD&ET), pp. 36–39 (2020)

5. Karp, R.M.: Reducibility among combinatorial problems. In: Complexity of Computer Computations, pp. 85–103. Springer, Cham (1972). https://doi.org/10.1007/978-1-4684-2001-2_9
6. Li, Z., Li, N.: A novel multi-mutation binary particle swarm optimization for 0/1 knapsack problem. In: 2009 Chinese Control and Decision Conference, pp. 3042–3047 (2009)
7. Liu, L.: Solving 0–1 knapsack problems by greedy method and dynamic programming method. Adv. Mater. Res. **282–283**, 570–573 (2011)
8. Martello, S., Pisinger, D., Toth, P.: New trends in exact algorithms for the 0–1 knapsack problem. Eur. J. Oper. Res. **123**, 325–332 (2000)
9. Zhao, J., Huang, T., Pang, F., Liu, Y.: Genetic algorithm based on greedy strategy in the 0–1 knapsack problem. In: 2009 Third International Conference on Genetic and Evolutionary Computing, pp. 105–107 (2009)