F O U N D A T I O N S O F C O M P U T I N G A N D D E C I S I O N S C I E N C E S
Vol. 47 (2022) No. 3
ISSN 0867-6356
DOI: 10.2478/fcds-2022-0014 e-ISSN 2300-3405

On Solving 0/1 Multidimensional Knapsack Problem with a Genetic Algorithm Using a Selection Operator Based on K-Means Clustering Principle

Soukaina Laabadi [1], Mohamed Naimi [2,] *, Hassan El Amri [1], Boujemâa Achchab [2]

Abstract. The growing need for profit maximization and cost minimization has made the optimization field very attractive to both researchers and practitioners. In fact, many authors were interested in this field and they have developed a large number of optimization algorithms to solve either academic or real-life problems. Among such algorithms, we cite a well-known metaheuristic called the genetic algorithm. This optimizer tool, as any algorithm, suffers from some drawbacks; like the problem of premature convergence. In this paper, we propose a new selection strategy hoping to avoid such a problem. The proposed selection operator is based on the principle of the k-means clustering method for the purpose of guiding the genetic algorithm to explore different regions of the search space. We have elaborated a genetic algorithm based on this new selection mechanism. We have then tested our algorithm on various data instances of the 0/1 multidimensional knapsack problem. The obtained results are encouraging when compared with those reached by other versions of genetic algorithms and those reached by an adapted version of the particle swarm optimization algorithm.

Keywords: Genetic Algorithm, K-Means Clustering Method, Optimization, Selection Operator, 0/1 Multidimensional Knapsack Problem,

1. Introduction

The 0/1 multidimensional knapsack problem (0/1 MKP) can be defined as follows: given a set of *N* items and a knapsack having *D* dimensions. Each dimension *j* has a limited capacity $c_j(>0)$, and each item i has a profit $p_i(>0)$ and a size s_{ij} relative to each dimension j, where $(0 \leq s_{ij} \leq c_j)$. The goal is to select a subset of items that maximizes the knapsack total profit and meet all capacity constraints. This optimization problem is modeled as a binary linear programming problem:

[1] Hassan II University, Teacher's Training College (ENS), Casablanca, Morocco, *soukainalaabadi@gmail.com* (S. Laabadi), *hassanelamri@gmail.com* (H. El Amri)
[2] Hassan First University, National School of Applied Sciences (ENSA), Berrechid, Morocco, *mohamed.naimi@uhp.ac.ma* (* Corresponding author: M. Naimi), *achchab@estb.ac.ma* (B. Achchab)

$$\begin{cases} Max \sum_{i=1}^{N} p_i x_i & (1) \\ s.t. & \\ \sum_{i=1}^{N} s_{ij} x_i \leq c_j; \ \forall j \in \{1, \dots, D\} & (2) \\ x_i \in \{0,1\}; \ \forall i \in \{1, \dots, N\} & (3) \end{cases}$$

Where,

$$x_i = \begin{cases} 1 \ if \ the \ i-th \ item \ is \ packed \ into \ the \ knapsack \\ 0 \ otherwise \end{cases} \quad (4)$$

According to Karp [19], the 0/1 MKP belongs to the class of NP-hard problems. This problem can be used by the operational research (OR) community as a test problem to evaluate the performance of their proposed algorithms. It can also be used by OR practitioners to model complex optimization problems in several areas. In fact, the term "knapsack" here is a generic name which could stand for a "container", as in the loading context, "a sprint" as in data warehousing [14] or an "investment", as in capital budgeting problem [21]. For more real-world applications of 0/1 MKP, one can refer to [24].

The Genetic Algorithm (GA) is an optimization technique that mimics the process of natural selection [7] and uses reproduction inspired operators to build a satisfactory solution. Since its apparition, this bio-inspired algorithm has been frequently applied for solving several continuous as well as combinatorial optimization problems [3, 8]. Thanks to its simplicity and its adaptability to different engineering problems, it continues to be one of the most useful optimization tools [31].

The *k*-means method [27] is a clustering technique whose principle consists of partitioning a given number of objects into distinct clusters. Clusters are built in such a way as to have similar or fairly close objects, in terms of distance, within the same cluster. The *k*-means method is the most commonly used clustering technique since it is simple and efficient, mostly when a very large number of objects must be classified. So, this method has been applied for clustering data in many fields as image segmentation [35], data mining [17], and tumor diagnosis [22].

The aim of this paper is to introduce a new selection strategy, in which the basic steps of a clustering technique are incorporated to select the parent chromosomes for outbreeding and hence creating new generations. The clustering technique integrated in this selection strategy is based on the well-known *k*-means method. By fixing the number of clusters to 2 ($k = 2$), we are facing to two groups of parent chromosomes wherein the parents of the same group are genetically closest and the parents of different groups are genetically distant. In this way, the proposed selection strategy can pick out easily a couple of parents with dissimilar features while selecting each one from a different cluster. We then integrate our selection operator into a GA framework in order to validate its performance through computational results for various 0/1 MKP benchmark instances.

The rest of this paper is organized as follows. The second section presents a brief review of the relevant literature. In the third section, we first introduce the fundamental operators of

the GA and we then give the general operating principle of the *k*-means clustering method. The fourth section provides specific details about how the proposed algorithm can be adapted to solve the 0/1 MKP. The fifth section is devoted to examining the performance of the proposed GA in comparison with other similar algorithms in the literature. We finally give some conclusions in the sixth section.

2. Literature review

Because of the drawback of classical GAs that is the premature convergence, many researchers have focused their attention on enhancing the search process of GAs while making an equilibrium between the diversification and intensification strategies. In fact, the diversification and intensification strategies are two major factors of success for any metaheuristic algorithm. The diversification strategy explores the search space hoping to find new promising regions, whilst the intensification strategy strengthens the local search ability for finding good quality solutions.

Here are some works that have been interested in improving the performance of GAs for solving optimization problems. Chu and Beasley [6] have proposed a repair operator using greedy-like heuristic to deal with infeasible chromosomes. Their GA solves small to large 0/1 MKP instances with highly competitive results. Yang et al. [38] have proposed a technique to reduce and integrate rough sets into the crossover operator of the GA. This technique leads to interesting results in the 0/1 MKP context. Varnamkhasti and Lee [34] as well as Laabadi et al. [23, 25] have developed GAs taking into consideration the sex of chromosomes in the mating pool. They proved the good performance of their algorithms versus the traditional GAs using either unidimensional or multidimensional instances of the binary knapsack problem. Juang [18] has hybridized the particle swarm optimization algorithm with GA. The resultant hybrid algorithm has been applied to recurrent networks design problems. The obtained results are quite better than those obtained by a pure GA and those achieved by a pure particle swarm optimization algorithm. Otherwise, Lee et al. [26] have combined GA with the ant colony optimization algorithm to solve the multiple sequence alignment problem. Their approach succeeded in improving the performance of GA for the considered problem.

On the flip side, the *k*-means method has also its own disadvantages. Among these drawbacks, we can cite the following ones. The *k*-means method often falls into local minima. It is also vulnerable to outliers and noisy data. Another drawback of this method is the difficulty of correctly determining the number of clusters in advance. To remedy such weaknesses, some authors had the idea to combine GA with the *k*-means method for tackling the clustering problems. Maulik and Bandyopadhyay [28] have indeed proposed a hybrid algorithm that exploits the searching capability of GA in order to search the most suitable cluster centers by optimizing a given metric of similarity. The computational results have proved the performance of their clustering algorithm when it has been tested on artificial as well as real-life data sets. In the same way, Bandyopadhyay and Maulik [4] have incorporated the GA into *k*-means method for providing a perturbation to escape from local minima. The performance of their algorithm has been tested and validated upon a real-life problem that is classifying the pixels of a satellite image concerning part of the Mumbai city. Furthermore, Santhanam and Padmavathi [30] have developed a well-adapted clustering strategy based on

the *k*-means pattern to remove outliers and noisy data. In their approach, the reduced data is used for selecting the optimal features through a GA. They have applied their algorithm to handle the enormous flow of data available in health industry.

That is, some authors have been interested in solving optimization problems by combining the *k*-means method with a population-based metaheuristic in order to avoid the above-mentioned shortcomings. Indeed, Chehouri et al. [5] have developed a general selection strategy based on two different versions of the *k*-means method; when k is fixed and when k has to be optimized. Their strategy is based on four major steps: clustering, membership phase, fitness scaling and selection. They have tested the performance of both versions of their selection process on seven unconstrained continuous optimization problems. Gupta et al. [16] have elaborated a crossover operator based on k-means clustering technique. Their crossover operator uses the similarity concept between individuals in the genotypic search space of 0/1 MKP. They have integrated this operator into a GA using also a fuzzy roulette wheel selection operator. The resulting algorithm showed its performance in the 0/1 MKP context. García et al. [11] have proposed an operator based on the *k*-means method aiming to binarize two naturally continuous metaheuristics, which are the cuckoo search and the black hole algorithms. The experiments carried out by the authors indicated that their proposal contributes to improve the solutions quality for 0/1 MKP.

3. Preliminaries

In this section, we first present the general principle of the GA. Then, we present the basic functioning of the *k*-means clustering method.

3.1. The genetic algorithm

The GA is an optimization technique inspired from the natural genetics. It iteratively updates a population of individuals (i.e., solutions) through an evolution mechanism in order to produce better individuals for the next generation. The solution retained is the best one obtained at the end of the algorithm running (e.g., after a fixed number of generations). The solutions for a given optimization problem are defined as chromosomes that are composed by a set of genes representing potential values of the design parameters of the considered problem (i.e., decision variables). The quality of chromosomes (i.e., solutions) is measured by a fitness function thanks to the genetic information stored in their genes. In the GA, the chromosomes are iteratively updated by applying bio-inspired operators (i.e., genetic operators) that are selection, crossover and mutation. The chromosomes undergo crossover and mutation operations with a certain probability (the crossover probability p_c and the mutation probability p_m). If it is necessary, a repair operator is applied in order to restore the feasibility of chromosomes that do not satisfy the problem constraints.

The selection and crossover operators play a vital role in altering the diversification and intensification strategy. In fact, these two operators are the key points of interest for researchers seeking to improve the performance of GAs. However, the mutation and repair operators play a secondary role which consists in contributing to increase the genetic diversity within the population while restoring the feasibility of chromosomes if necessary.

Some authors have used local search instead of mutation strategy giving rise to the so-called memetic algorithms [15]. Other authors have preferred to keep infeasible chromosomes in the population by using other techniques like the penalty technique instead of using a repair operator [29].

On one hand, we can find several traditional selection strategies which are essentially based on fitness criterion to select the parent chromosomes. As examples, we cite the tournament selection, the roulette wheel selection, the statistical universal sampling and the truncation selection. More details about these classical selection operators can be found in the survey of Whitley and Sutton [36]. Another type of selection operators is based on the similarity metric between the parents while taking into account the sex of chromosomes. In this sense, we can cite the sexual selection of Varnamkhasti and Lee [34] and that of Goh et al. [12]. In the same way, we can also cite the improved sexual selection of Laabadi et al. [25].

On the other hand, there are two types of crossover operators in the literature. The first type regroups the so-called non-specific-knowledge crossover operators (i.e., general crossover operators that can be independently applied to the problem at hand). Among these operators, we can find the *n*-point crossovers, the uniform crossover and the average crossover [33]. The second type includes the so-called specific-knowledge crossover operators (i.e., they are designed to solve a specific problem taking into account its particular structure). Examples of such operators include the partially-mapped crossover operator for traveling salesman problems [13] and the two-stage recombination operator for knapsack problems [2, 25].

3.2. The k-means clustering method

The *k*-means method [9, 32] is likely the most popular clustering algorithm that has been used to solve a great variety of clustering problems. The principal aim of this method, as for any other clustering algorithm, consists of grouping similar objects together. Each of such groups is called a cluster, that is; a region in which the similarity metric between objects is locally lower than in other regions. The center of each cluster is called a centroid.

The *k*-means method firstly provides *k* initial partitions (i.e., clusters) of the entire set of objects, then it calculates the centroids of these initial clusters (see Figure 1). Thereafter, the objects are moved to the cluster represented by the nearest centroid in order to minimize the similarity metric between objects within the same cluster. It should be noted that the centroids of clusters need to be updated each time some objects move from one cluster to another. This process is then repeated until the centroid of each cluster remains unchanged. In other words, the process stops when no object can change its membership cluster anymore (i.e., the similarity metric can no longer be minimized any further).

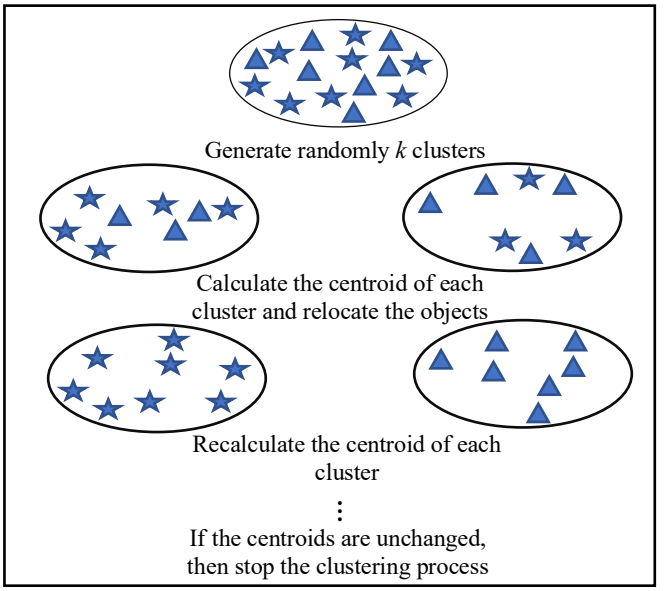


Figure 1. Main steps of the k-means method.

4. The proposed genetic algorithm

The simplicity and especially the efficiency of the k-means method encouraged us to reflect on how to use its principle to develop a new selection strategy of GA in expectation to build dissimilar couples of parent chromosomes for the crossover stage. The goal is to control the genetic diversity from earliest stages of the considered GA. That is, we have elaborated a new selection operator, called the k-means selection strategy (kMS).

Certainly, using k-means as a selection strategy in GA is not a new idea. Indeed, Chehouri et al. [5] were the first authors to propose this combination. However, the principle of their method is quite different from that of ours. That is, our proposed selection strategy is inspired by the principle of clustering to facilitate the selection of dissimilar parents for recombination. It essentially aims to promote diversity within the mating pool, whereas the selection strategy of Chehouri et al. [5] systematically favors the individuals with higher values according to a fitness scaling process. Besides, their GA is proposed for continuous optimization problems, more precisely, for unconstrained ones. In the contrary, our GA is dedicated to combinatorial and constrained optimization problems. In fact, its current version is especially designed for 0/1 MKP. More details about the proposed GA are described below.

The GA using the *k*-means selection strategy works in the phenotypic search space for dealing with 0/1 MKP. It is therefore necessary to encode chromosomes by a real value representation allowing to calculate the clustering metric of the *k*-means method between the candidate chromosomes. In fact, the chromosomes' genes fitness function values (i.e., decision variables' fitness function values) are encoded as follows (for $i \in \{1, \dots, N\}$):

$$f(i) = f(x_i) = \begin{cases} \dfrac{p_i}{\sum_{j=1}^{D} s_{ij}} & if\ the\ item\ i\ is\ packed\ into\ the\ knapsack \\ 0\ otherwise \end{cases} \quad (5)$$

We note that the proposed GA randomly generates the individuals of its initial population by arbitrarily packing the items into knapsack instances (i.e., chromosomes). However, this procedure may generate some infeasible solutions that violating at least one capacity constraint of the knapsack. To deal with this, the proposed GA applies a repair technique considered by Zouache et al. [39]. Their technique is based on the following two steps. First, the repair step that sorts increasingly all items according to the ratio $p_i / \sum_{j=1}^{D} s_{ij}$, and then removes items iteratively until the feasibility of chromosomes is restored. Second, the improvement step that consists of enhancing the quality of obtained feasible chromosomes. In concrete terms, this second step first starts by sorting decreasingly all items according to the aforementioned ratio. Then, it proceeds to gradually insert them into the knapsack until the latter is fulfilled while respecting its capacity.

Inspired by the principle of *k*-means method ($k = 2$), we have divided the population into two dissimilar clusters of chromosomes. To measure the similarity between a chromosome and a centroid (see Definition 1), we have used the Manhattan distance (see Definition 2) since it is more preferable for quantitative measures than other similarity metrics [1].

Definition 1

The initial centroid of each cluster, in the k-means selection strategy, is represented by a chromosome selected from the population. That is, the initial centroid-chromosome of the first cluster is the fittest chromosome in the whole population, while the initial centroid-chromosome of the second cluster is the more unfit chromosome with respect to all other chromosomes. However, to compute the genes' values of the centroid-chromosome of each new cluster k denoted by X_k^*, we proceed as follows:

$$gene_i\,(X_k^*) = \frac{\sum_{l=1}^{\Omega_k} f_l(i)}{\Omega_k};\ for\ i = 1, \ldots, N \quad (6)$$

Where Ω_k is the number of chromosomes in the cluster k and $f_l(i)$ is the fitness function value of the i-th gene in the chromosome l (see equation (5)).

Definition 2

The Manhattan distance (MD) that should be minimized is defined as follows:

$$MD(X, X_k^*) = \sum_{i=1}^{N} |gene_i(X) - gene_i(X_k^*)|\ ;\ k = 1, 2 \quad (7)$$

Where,

$\{X = \{X_l\} / l = 1, \ldots, (P-2)\}$ *is the set of chromosomes contained in the initial population without computing the centroid-chromosomes of the two clusters. Note that P is the size of the initial population.*

$gene_i(X)$ *is the value of the* i*-th gene in the chromosome* X*.*

$gene_i(X_k^*)$ *is the value of the* i*-th gene in the centroid-chromosome* X_k^* *of the cluster* k*.*

During the *k*-means selection strategy, the chromosomes belonging to the cluster with the smallest size are all placed in the mating pool. Moreover, each chromosome of such a cluster

represents the first parent of a couple being constructed (i.e., the current couple). The second parent in the current couple is selected from the next cluster using the binary tournament selection strategy. For the sake of clarity, the binary tournament selection strategy first chooses two chromosomes at random way. Then, it selects the fittest one of them as a winner based on their fitness values. Notice that the parents are considered sequentially without replacement, which means that each parent will be selected once for building a couple.

In the proposed GA, different crossover operators are applied separately and then some of the generated offspring chromosomes are chosen with a certain probability to be mutated with a standard mutation operator. In our case, we have chosen the flip-bit mutation that consists of selecting randomly one item and changing its statute (if it is packed into the knapsack, we will then remove it from the knapsack and vice versa). Such an operator can thus help to increase the genetic diversity in the population. Thereafter, the obtained offspring chromosomes replace their parents, giving rise to the next generation. The proposed GA the algorithm stops as soon as it reaches a certain number of generations fixed beforehand. Of course, we can choose another stopping criterion (e.g., a fixed execution time). The key steps of our GA are shown in Algorithm 1.

Algorithm 1. GA Based on K-Means Clustering Selection.

1: Initialize the population with P chromosomes
2: Define the crossover and the mutation probability
3: Set a maximum number of generations ($MaxGen$)
4: While $t < MaxGen$
5: For each chromosome X_l ($l \in \{1, \dots, P\}$)
6: Check its feasibility
7: Apply the repair operator if it is necessary
8: Evaluate it by the following fitness function: $\sum_{i=1}^{N} p_i\, x_i$
9: End For
10: Divide the population into two clusters by using the proposed variant of the *k*-means method as described earlier in this section
11: Build couples of parent chromosomes (X_{la}, X_{lb}) as follows (if $\Omega_1 < \Omega_2$):

$$\begin{cases} X_{la} \text{ is randomly selected from the cluster } 1 \\ X_{lb} \text{ is obtained by applying the tournament selection to cluster } 2 \end{cases} \quad (8)$$

Where, X_{la} is the first parent chromosome and X_{lb} is the second parent chromosome.
12: Perform the crossover operation on the generated couples with a probability p_c
13: Apply the mutation operator to the obtained children with a probability p_m
14: Apply the repair operator to infeasible children
15: Replace all parent chromosomes by their children and update the population
16: End while
17: Return the fittest chromosome as a solution of 0/1 MKP

5. Simulations and comparisons

In this section, we first introduce the experimental protocol which we have adopted when performing the numerical simulations. Then, we present and discuss the results obtained by the proposed algorithm when compared with other GAs as well as a Binary Particle Swarm Optimization algorithm (BPSO) [20]. We chose two stopping criteria, namely; a fixed number of generations/iterations and a fixed runtime. Such a decision concerning the choice of two different termination criteria aims to have correct comparisons between all considered algorithms.

5.1. Experiments design

A quite large number of simulation experiments have been carried out to investigate the performance of the proposed GA against other algorithms. All simulations were executed on a 64-bits Windows 7 operating system; processor Intel® Core™ i5-2.5 GHz; 8 Go of RAM. It should also be noted that all algorithms are implemented in Java language. Each experiment was repeated 30 times and the average results are retained. On one hand, the simulation results of *k*-means selection strategy (kMS) are compared to those of BPSO (see [20] for more details on the BPSO algorithm used in comparative experiments). On the other hand, the obtained results are also compared to those obtained by other GAs using separately two types of selection strategies:

The traditional selection strategies;

- The binary Tournament Selection (TS).
- The Roulette Wheel Selection (RWS).
- The Truncation Selection (TrS).
- The Stochastic Universal Selection (SUS).

And the sexual selection strategies;

- The Sexual Selection strategy (SS) of Varnamkhasti and Lee [34].
- The Improved Sexual Selection strategy (ISS) of Laabadi et al. [25].

On the flip side, the crossover operation was accomplished via four crossover operators:

- The 2-Point Crossover (2-PC).
- The Uniform Crossover (UC).
- The first version of the two-Stage Recombination operator (2SR-v1) elaborated by Laabadi et al. [25] for the 0/1 MKP.
- The second version of the two-Stage Recombination operator (2SR-v2) developed also in [25].

In the implemented GAs, the population size is set to $N = 50$, with a crossover probability of 0.95 and a mutation probability of $1/N$. For TrS strategy, the truncation threshold is set to 50%. The BPSO is performed by using the following parameters: the same population size as for GAs, a minimum/maximum inertia weight of 0.4/0.9, an acceleration coefficient of 1.5 and a velocity between -2 and 2. It should be noted that all parameters are adjusted either by preliminary tests or by referring to the literature. The used benchmark test instances are those introduced by Chu and Beasley [6] (see http://people.brunel.ac.uk/~mastjjb/jeb/orlib/files/ for data files). We note that these data sets were generated with $N(\in \{100, 250, 500\})$ items, $D(\in \{5, 10, 30\})$ dimensions, and $\alpha(\in \{0.25, 0.50, 0.75\})$ tightness ratios. For each $(N -$

D) combination, there are 30 test problems where each 10 test problems concern a tightness ratio. The test instances having 100, 250 and 500 items are stated respectively, as small, medium, and large-scaled instances denoted by mknapcb*i* where *i* is their index.

The simulated results of the proposed GA using kMS strategy and its rivals are presented in terms of the average fitness value and the average runtime. In order to evaluate the performance of our algorithm, we have used the average percentage gap:

$$\%Gap = \frac{\overline{b} - \overline{z}}{\overline{b}} \times 100 \tag{9}$$

Where $\overline{b} - \overline{z}$ is the difference between the mean of the benchmark solutions and the mean of our obtained solutions for each $(N - D - \alpha)$ combination. We note that a solution is considered better than another solution when the percentage gap of the former is lower compared to the latter. Furthermore, we have applied statistical tests when the differences between the results are quite small.

5.2. Experiments with a fixed number of iterations

In this subsection, the stopping criterion of all compared algorithms is set to 100 iterations.

5.2.1. Results discussion for small-scaled 0/1 MKP instances

Table 1 records the percentage gap of the proposed GA using kMS strategy against BPSO and other GAs using traditional and sexual selection strategies while performing on small 0/1 MKP test instances. For each GA, we have applied separately two classical crossover operators. We note that the first row indicates the name of each test problem. The second row reports the best mean solutions found for the benchmark. The next rows outline the %Gap of all compared algorithms (in total, there are 14 GAs and BPSO). The best outcomes are underlined and formatted in bold. Since the results are quite similar, the Friedman test [10] is carried out for testing whether differences between the comparative algorithms are quite significant or not, at a significance level of 0.05.

In fact, as we can see in Table 2, the p-value obtained by Friedman test indicates that the compared algorithms behave differently since this indicator is less than 0.05. In Tables 1 and 2, the GA based on kMS strategy is ranked first when using 2-PC operator followed by the GAs using ISS and SS strategies. However, its performance degrades from the first rank to the third one when it is combined by the UC operator, but it is still better than the GAs using traditional selection strategies. We can also observe from Table 1 that all GAs' variants, including our proposed GA, are superior than BPSO in terms of solutions quality.

Table 3 represents comparative results (obtained on small 0/1 MKP test instances) of the GA based on kMS strategy against BPSO and GAs using ISS and SS when combined with specific-knowledge crossover operators (namely, 2SR-v1 and 2SR-v2). As in Table 1, Table 3 also indicates the name of each test problem, then it reports the mean best solutions of the corresponding instances. It also shows the %Gap of algorithms while mentioning the ranks of all compared algorithms in the last column. From this table, one can see that the GA using

kMS strategy with 2SR-v1 or 2SR-v2 operators is outperformed by the GAs combining ISS or SS strategy with the same crossover operators as well as by BPSO.

Table 1. Comparison of the average percentage gap of BPSO and GAs using classical crossover operators for small benchmark instances.

Test problem	mknapcb1			mknapcb4			mknapcb7		
Best solution	24197,2	43252,9	60471	22601,9	42659,1	59555,6	21654,2	41431,3	59199,1
GAs with UC operator									
kMS	1,94	**1,19**	**0,83**	3,61	1,96	**1,33**	4,32	2,42	**1,58**
ISS	**1,79**	1,22	1,02	**3,05**	1,89	1,5	**3,73**	**2,33**	1,7
SS	**1,79**	1,23	1,08	3,11	**1,87**	1,49	3,86	2,34	1,74
TS	2,08	1,32	1,17	3,34	2,06	1,62	4,03	2,42	1,81
RWS	2,09	1,42	1,4	3,49	2,13	1,83	4,21	2,58	2,03
TrS	2,26	1,49	1,31	3,6	2,24	1,79	4,26	2,66	1,92
SUS	1,96	1,32	1,05	3,27	2,01	11,06	4,05	2,41	1,73
GAs with 2-PC operator									
kMS	**2,92**	**2,5**	**2,86**	**4,27**	**3,12**	**3,38**	**4,88**	**3,56**	**3,47**
ISS	3,76	3,69	4,69	5,01	4,13	4,94	5,38	4,51	4,91
SS	3,82	3,72	4,78	4,9	4,39	5,03	5,37	4,58	4,94
TS	4,54	4,51	4,77	5,49	4,87	5,39	5,99	5,34	5,28
RWS	4,36	4,12	4,76	5,39	4,75	5,12	5,81	5,01	5,14
TrS	5,02	4,81	5,43	5,93	5,41	5,62	6,13	5,72	5,88
SUS	4,1	3,9	4,29	5,2	4,44	4,72	5,72	4,69	4,94
BPSO									
	8,49	8,19	7,13	9,28	8,56	7,31	9,86	8,93	7,62

Table 2. Friedman test on the average fitness of GAs using UC and 2-PC for small benchmark instances.

GAs with UC	Mean rank	GAs with 2-PC	Mean rank	
kMS_UC	11,89	kMS_2PC	7,11	p-value = 1,76E-16
ISS_UC	12,83	ISS_2PC	5,67	
SS_UC	12,82	SS_2PC	4,89	
TS_UC	10,44	TS_2PC	2,22	
RWS_UC	9,00	RWS_2PC	3,22	
TrS_UC	8,67	TrS_2PC	1,11	
SUS_UC	10,56	SUS_2PC	4,56	

Table 3. Comparison of the average percentage gap of BPSO and GAs using specific-knowledge crossover operators for small benchmark instances.

Test problem	mknapcb1			mknapcb4			mknapcb7			Rank of GAs
Best solution	24197,2	43252,9	60471	22601,9	42659,1	59555,6	21654,2	41431,3	59199,1	
GAs with 2SR_v1 operator										
kMS	9,78	12,1	23,82	10,28	12,17	23,37	10,05	12	22,43	4
ISS	2,41	1,23	0,47	4,29	1,98	1,02	4,63	2,64	1,28	1
SS	2,48	1,25	0,46	4,4	2,02	1,01	4,75	2,67	1,29	2
GAs with 2SR_v2 operator										
kMS	9,86	11,92	24,38	10,23	12,14	23,35	10,14	11,89	22,45	4
ISS	1,03	1,5	1,17	1,61	1,69	1,31	2,05	1,8	1,52	1
SS	1,02	1,52	1,17	1,61	1,68	1,35	2,07	1,84	1,51	2
BPSO										BPSO rank vs all GAs
	8,49	8,19	7,13	9,28	8,56	7,31	9,86	8,93	7,62	3

Figure 2 depicts two radar graphs representing the average time (in seconds) consumed over small instances by the GA based on kMS strategy against that of GAs using traditional as well as sexual selection strategies. The first, respectively the second, graph in this figure concerns GAs using, respectively, UC operator and 2-PC operator.

In Figure 2, the first graph shows the GA based on kMS strategy runs slowly compared to all other comparative GAs when combined with the UC operator. Indeed, it consumes more time as the number of dimensions increases (see mknapcb7 axes). In the same figure, the second graph shows that the GA using kMS selection strategy with 2-PC operator takes less

time than GAs using traditional selection strategies with 2-PC operator. However, it takes a little more time than GAs using sexual selection strategies with 2-PC operator.

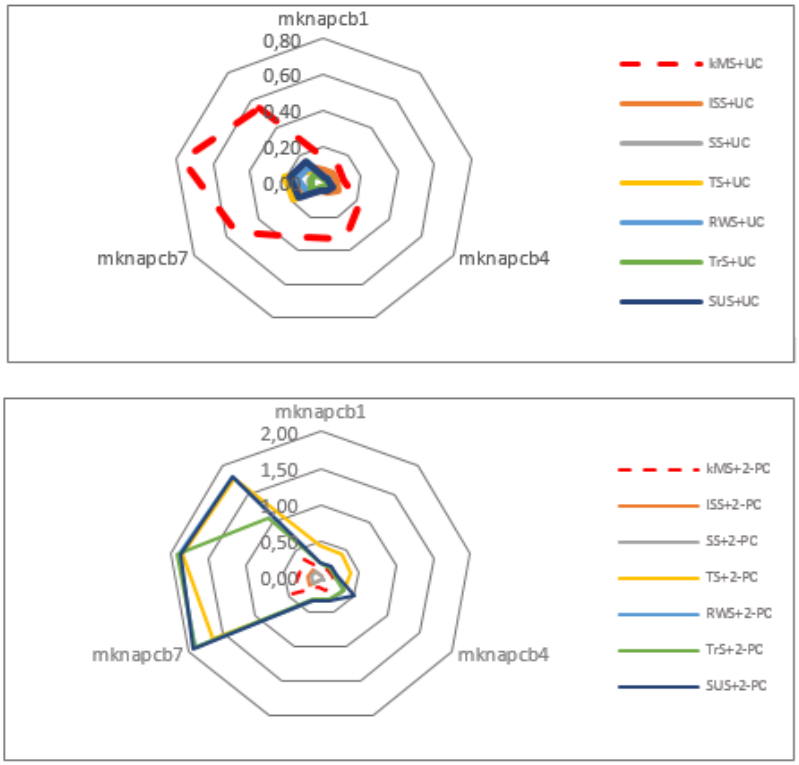


Figure 2. Average time comparison of GAs with traditional crossover operators on small benchmark instances.

In Table 4, a Friedman comparison test is conducted based on the average runtime consumed by the GA using kMS strategy with 2SR-v1 or 2SR-v2 operators. From Table 4, we first see that there is a significant difference between the runtimes of the rival algorithms since p-value is less than the significance level (0.05). Indeed, this Friedman test shows that all GA variants run quickly than BPSO. It also shows that GAs combining kMS with either 2SR-v1 or 2SR-v2 crossover operators need a shorter runtime in comparison with GAs combining ISS or SS strategies with the same crossover operators. Furthermore, we can see that the GA based on kMS takes less time when combined with 2SR-v1 than when combined with 2SR-v2.

Table 4. Friedman test on the average time consumed by BPSO and GAs with specific-knowledge crossover operators for small benchmark instances.

Runtime of algorithms	Mean rank	
kMS with 2SR-v1	2,05	p-value = 1,30E-06
kMS with 2SR-v2	3,22	
ISS with 2SR-v1	2,17	
ISS with 2SR-v2	4,83	
SS with 2SR-v1	2,11	
SS with 2SR-v2	5,17	
BPSO	7,00	

5.2.2. Results discussion for medium to large-scaled 0/1 MKP instances

In this subsection, the simulated results for medium to large-scaled 0/1 MKP instances, including mknapcb2, mknapcb3, mknapcb5, mknapcb6, mknapcb8, and mknapcb9, are discussed. On one hand, Table 5 compares the average percentage gap of the GA based on kMS strategy against BPSO and GAs using ISS, SS, TS, RWS, TrS and SUS strategies when they are working together with UC and 2-PC operators. Since the results are quite similar as in the subsection above, we have performed a Friedman test with the same significance level used before (0.05). Table 6 reports the mean ranks showing how the GAs differ according to this test. Table 7 compares the results accuracy of the GA based on kMS strategy versus the GAs using ISS and SS strategies when applying 2SR-v1 and 2SR-v2 as crossover operators. Note that Table 5 and Table 7 represent the computational results in the same fashion as Table 1 while using the average percentage gap as a performance indicator.

The results depicted in Table 5 and Table 6 demonstrate a better efficiency of the GA based on kMS strategy, when it is combined either with UC or 2-PC, in comparison with all other tested algorithms. Besides, all GAs considered here -including our proposed GA- outperform BPSO. However, as shown in Table 7, the performance of the proposed GA degrades compared to GA based on ISS or SS strategy when combined either with 2SR-v1 or 2SR-v2. Moreover, we observe that this variant of the proposed GA (i.e., kMS combined either with 2SR-v1 or with 2SR-v2) is slightly less efficient than BPSO. Notice that the negative values, appearing in the third column of mknapcb3 test problem in both Table 5 and Table 7, show the superiority of the corresponding GA results against those existing in the literature (see equation (9)).

Table 5. Comparison of the percentage gap of BPSO and GAs using classical crossover operators for medium to large benchmark instances.

Test problem	mknapcb2			mknapcb5			mknapcb8		
Best solution	60409,7	109284,6	151555,9	58993,9	108706,4	151330,4	56875,9	106673,7	150443,5
GAs with UC operator									
kMS	1,63	**1,22**	**0,93**	2,74	**1,83**	**1,27**	3,71	**2,23**	**1,53**
ISS	**1,53**	1,39	1,68	2,36	1,9	2,03	3,15	2,28	2,19
SS	1,58	1,41	1,73	**2,31**	1,88	1,99	**3,13**	2,32	2,18
TS	1,78	1,45	1,74	2,52	2,03	1,98	3,4	2,38	2,2
RWS	1,91	1,8	2,52	2,69	2,3	2,62	3,48	2,64	2,91
TrS	2,04	1,75	1,93	2,84	2,23	2,21	3,59	2,64	2,36
SUS	1,68	1,45	1,71	2,52	1,97	1,94	3,34	2,34	2,13
GAs with 2-PC operator									
kMS	**3,32**	**3,92**	**6,21**	**3,87**	**4,32**	**6,24**	**4,63**	**4,65**	**6,28**
ISS	4,4	5,51	8,69	4,93	5,74	8,66	5,37	6,08	8,84
SS	4,4	5,58	8,68	4,88	5,78	8,89	5,47	6,01	8,66
TS	5,15	6,14	8,63	5,64	6,44	8,84	6,32	6,73	8,66
RWS	4,96	5,78	8,7	5,39	6,23	8,59	6,02	6,54	8,61
TrS	5,72	6,6	9,89	6,04	6,96	9,55	6,51	7,18	9,68
SUS	4,82	5,59	7,84	5,36	5,93	8,01	5,86	6,18	7,97
BPSO									
	12,94	10,54	11,93	13,64	11,16	11,96	14,41	11,51	11,7

Test problem	mknapcb3			mknapcb6			mknapcb9		
Best solution	120615,5	219503,1	295832,5	118565,5	217274,6	302556	115473,5	216156,9	302353,4
GAs with UC operator									
kMS	**1,53**	**1,5**	**-0,05**	**1,95**	**1,81**	**2,23**	2,92	**2,24**	**2,41**
ISS	1,76	2,09	1,16	2,14	2,32	3,37	**2,91**	2,75	3,43
SS	1,78	2,12	1,09	2,11	2,33	3,39	2,89	2,75	3,46
TS	1,91	2,03	0,99	2,18	2,27	3,18	2,95	2,71	3,26
RWS	2,25	2,67	2,32	2,57	2,93	4,57	3,22	3,34	4,58
TrS	2,19	2,35	1,26	2,48	2,63	3,45	3,33	3,05	3,52
SUS	1,94	2,2	1,29	2,27	2,51	3,56	3,04	2,9	3,6
GAs with 2-PC operator									
kMS	**3,88**	**5,7**	**8,91**	4,03	**5,85**	**10,83**	**4,7**	**6,21**	**10,69**
ISS	4,98	7,29	13,98	**3,35**	6,32	15,96	5,6	7,71	15,51
SS	5,03	7,43	13,91	5,03	7,32	15,85	5,59	7,69	15,62
TS	5,83	7,83	13,82	5,8	7,88	15,59	6,34	8,31	15,28
RWS	5,51	7,61	13,73	5,58	7,75	15,4	6,21	8,14	14,9
TrS	6,01	8,23	15	6,11	8,36	16,54	6,62	8,61	16,41
SUS	5,44	7,35	12,2	5,46	7,47	13,82	53,03	52,94	53,37
BPSO									
	15,26	11,97	17,62	15,4	12,3	19,27	15,87	12,81	19,05

Table 6. Friedman test on the average fitness of GAs using UC and 2-PC for medium to large benchmark instances.

GAs with UC	Mean rank	GAs with 2-PC	Mean rank	p-value = 8,77E-39
kMS_UC	13,17	kMS_2PC	6,94	
ISS_UC	12,17	ISS_2PC	4,78	
SS_UC	12,22	SS_2PC	4,50	
TS_UC	11,50	TS_2PC	2,83	
RWS_UC	8,39	RWS_2PC	3,67	
TrS_UC	8,94	TrS_2PC	1,17	
SUS_UC	10,61	SUS_2PC	4,11	

Table 7. Comparison of the percentage gap of BPSO and GAs using specific-knowledge crossover operators for medium to large benchmark instances.

Test problem	mknapcb2			mknapcb5			mknapcb8		
Best solution	60409,7	109284,6	151555,9	58993,9	108706,4	151330,4	56875,9	106673,7	150443,5
GAs with 2SR_v1 operator									
kMS	9,82	12,64	29,64	9,95	12,84	29,26	10,03	12,81	28,99
ISS	2,16	1,13	0,41	**3,41**	**2,11**	**0,73**	**4,24**	**2,5**	**1,09**
SS	**2,12**	**1,12**	**0,4**	3,42	2,12	**0,73**	4,29	2,53	1,1
GAs with 2SR_v2 operator									
kMS	9,8	12,65	29,59	9,86	12,88	29,4	10,1	12,99	28,67
ISS	1,53	3,19	1,89	1,97	3,55	**2,16**	**2,27**	**3,76**	**2,49**
SS	**1,52**	**3,19**	**1,87**	**1,95**	**3,55**	2,19	2,29	**3,76**	2,53
BPSO									
	12,94	10,54	11,93	13,64	11,16	11,96	14,41	11,51	11,7
Test problem	**mknapcb3**			**mknapcb6**			**mknapcb9**		
Best solution	120615,5	219503,1	295832,5	118565,5	217274,6	302556	115473,5	216156,9	302353,4
GAs with 2SR_v1 operator									
kMS	9,4	13,34	30,69	9,32	13,32	31,98	9,57	13,44	31,8
ISS	1,67	**0,9**	**-1,83**	**2,41**	**1,49**	**0,61**	**3,73**	**1,93**	0,94
SS	**1,66**	**0,9**	**-1,84**	2,43	**1,49**	**0,61**	**3,73**	1,96	**0,93**
GAs with 2SR_v2 operator									
kMS	9,36	13,32	30,72	9,33	13,37	32,07	9,52	13,53	31,66
ISS	**2,56**	**5,41**	**0,77**	**3,02**	**5,8**	3,31	3,47	**6,2**	**3,68**
SS	2,58	5,43	0,79	3,03	5,82	**3,29**	**3,41**	6,21	**3,68**
BPSO									
	15,26	11,97	17,62	15,4	12,3	19,27	15,87	12,81	19,05

Figure 3 compares the convergence speed (in seconds) of the GA based on kMS strategy using UC or 2-PC crossover operators versus that of GAs based on traditional or sexual selection strategies using the same crossover operators. As shown in the first radar graph, the GA combining kMS with UC takes a relatively long time to achieve final solutions in comparison with the GAs using other selection techniques combined also with UC. However, one can see from the next radar graph that GA using kMS combined with 2-PC runs quickly than other GAs using classical selection strategies combined also with 2-PC. This graph shows, nevertheless, that GA based on kMS combined with 2-PC runs slowly than GAs based on sexual selection strategies combined also with 2-PC. In Table 8, the comparison test of Friedman proves that the GA incorporating kMS with 2SR-v1 or 2SR-v2 converges rapidly in comparison of BPSO and GAs using sexual selection operators with 2SR-v1 or 2SR-v2.

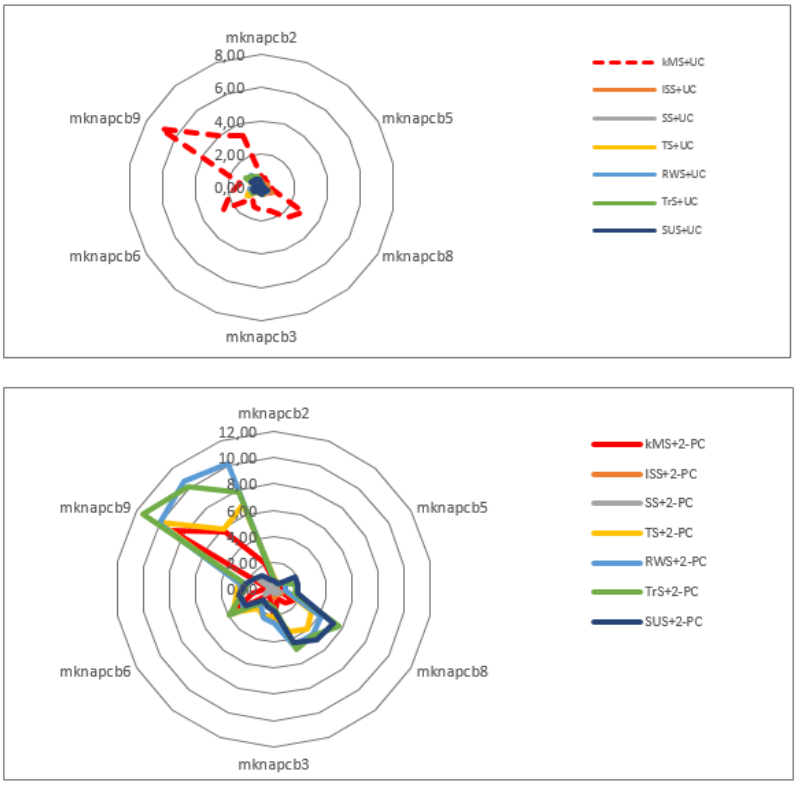


Figure 3. Average time comparison of GAs with traditional crossover operators on medium to large benchmark instances.

Table 8. Friedman test on the average runtime consumed by medium to large benchmark instances.

Algorithms	Mean rank	
kMS with 2SR-v1	2,10	P-value = 2,42E-17
kMS with 2SR-v2	2,72	
ISS with 2SR-v1	3,14	
ISS with 2SR-v2	5,33	
SS with 2SR-v1	2,31	
SS with 2SR-v2	5,61	
BPSO	7,00	

To recap the comparison between the proposed GA and the other considered GAs, let us denote a GA combination by a couple (S, C), where S is the selection strategy and C is the crossover operator used in the corresponding GA. The above-mentioned results illustrate that the combination (kMS, UC) gives good quality solutions mostly in large instances against all other tested combinations, but it requires a considerable computational time. The combination (kMS, 2-PC) is outperformed by (kMS, UC), but it still reaches better solutions than the GAs using classical as well as sexual selection strategies while requiring a reasonable runtime in all considered instances. Concerning the combinations (kMS, 2SR-v1) and (kMS, 2SR-v2), despite of their rapid convergence, they yield solutions of lower quality compared to those reached by other GAs rivals.

To also recap the comparison between our GA and BPSO, we can first say that the latter still requires higher running time than any GA variant in competition. Second, we can confirm that BPSO is clearly outperformed by (kMS, 2-PC) and (kMS, UC) in terms of the solutions quality. However, it slightly outperforms (kMS, 2SR-v1) and (kMS, 2SR-v2).

5.3. Experiments with fixed runtime

We propose in this section to choose the runtime as a termination criterion in order to carry out a legitimate comparison of kMS (combined with 2SR-v1 or 2SR-v2) versus BPSO, SS and ISS (combined with 2SR-v1 or 2SR-v2). To test the performance of these algorithms, we have used 270 Chu and Beasly's instances with different sizes. We have also set the fixed runtime to one second (1 s) at each run. The average results over 30 runs were finally retained for each experiment in order to make comparisons. The Wilcoxon test [37] is applied to compare the considered algorithms. This test is based on a paired comparison, it calculates the difference between a pair of values from two samples. If the difference is negative, this latter is counted as a negative rank; else if the difference is null, they are considered as an ex aequo; else it is counted as a positive rank. These measures are reported in Tables 9 and 10.

From Table 9, we can observe that GAs using kMS with 2SR-v2 or 2PC or UC work better in comparison to sexual GAs using the same crossover operators. However, GA using

kMS with 2SR-v1 has a worst performance than sexual GAs using 2SR-v1. Besides, Table 10 shows that GAs using kMS with either 2PC or UC or 2SR-v1 or 2SR-v1 achieve high-quality solutions when they are compared to BPSO.

Table 9. Wilcoxon test on average fitness concerning GA-based kMS versus GAs-based sexual selection strategies for 270 instances with different sizes.

GAs		Negative ranks	Positive ranks	Ex aequo	p-value
Selection operators	**Crossover operators**				
kMS versus ISS	2SR-v2	87	183	0	2,33E-13
kMS versus SS		58	212	0	6,29E-30
kMS versus ISS	UC	27	243	0	3,46E-27
kMS versus SS		55	215	0	5,17E-13
kMS versus ISS	2PC	6	264	0	1,11E-40
kMS versus SS		8	262	0	1,22E-40
kMS versus ISS	2SR-v1	141	128	1	1,88E-02
kMS versus SS		148	122	0	3,91E-02

Table 10. Wilcoxon test on average fitness concerning the proposed GA versus BPSO for 270 instances with different sizes.

Comparison	Negative ranks	Positive ranks	Ex aequo	p-value
kMS with 2SR-v1 versus BPSO	30	240	0	3,43E-17
kMS with 2SR-v2 versus BPSO	0	270	0	4,93E-46
kMS with UC versus BPSO	0	270	0	4,93E-46
kMS with 2-PC versus BPSO	0	270	0	4,93E-46

6. Conclusions

In this paper, we proposed a new selection strategy based on the *k*-means clustering method for solving 0/1 MKP. The proposed selection strategy depends on the structure of the problem at hand. It is based on two major stages. First, it applies the *k*-means method to build k (= 2) clusters. Then, it selects from each cluster one parent chromosome. The first parent is selected at random way, whilst the second parent is selected according to its fitness value. Selecting parents from different clusters produces non-identical child chromosomes, which can help prevent the premature convergence.

To examine the performance of the proposed selection strategy, we solved small to large-scale benchmark instances of 0/1 MKP. In fact, we chose the test instances so as to have various dimensions and complexities. Knowing that the overall performance of GAs is sensitive to the choice of the crossover operator, we have applied different crossover operators, including the standard and specific-knowledge ones. Then, we have compared our

selection strategy with traditional as well as recent selection techniques. We have also compared our algorithm with BPSO in order to show its performance against another population-based metaheuristic.

In a first step, we have realized numerical simulations while fixing a number of generations/iterations as a termination criterion. The obtained results have proved that kMS runs quickly and reaches slightly best results, when it is combined with general/traditional crossover operators, especially for large-scale instances. However, its performance seems to be slightly worse -in terms of quality- when it is combined with specific crossover operators, especially for small instances.

In a second step, we have fixed the runtime to the same value as a termination criterion for all compared algorithms. In this case, the outcomes showed that GAs based on kMS outperform sexual GAs when using traditional crossover operators. Further, GA based on kMS reaches better results than sexual GAs when combined with 2SR-v2. However, the performance of the proposed GA is slightly worse than sexual GAs when combined with 2SR-v1. Finally, still considering the second termination criterion, we have seen that all considered variants of our GA perform efficiently compared to BPSO.

We can then conclude that the proposed clustering selection method works better with general crossover operators than with specific ones in the 0/1 MKP context. This conclusion motivates us to think about a future research work that consists of adapting the proposed algorithm to some real-world combinatorial optimization problems (e.g., the capital budgeting problem).

References

[1] Aggarwal C. C., Hinneburg A., Keim D. A., On the Surprising Behavior of Distance Metrics in High Dimensional Space, in: *Proceedings of the 8th International Conference on Database Theory*, London, UK, Springer, 2001, 420-434.

[2] Aghezzaf B., Naimi M., The two-stage recombination operator and its application to the multiobjective 0/1 knapsack problem: A comparative study, *Computers and Operations Research*, **36**, 2009, 3247-3262.

[3] Aytug H., Khouja M., Vergara F. E., Use of genetic algorithms to solve production and operations management problems: A review, *International Journal of Production Research*, **41**, 2003, 3955-4009.

[4] Bandyopadhyay S., Maulik U., An evolutionary technique based on K-Means algorithm for optimal clustering in R^N, *Information Sciences*, **146**, 2002, 221-237.

[5] Chehouri A., Younes R., Khoder J., Perron J., Ilinca A., A Selection Process for Genetic Algorithm Using Clustering Analysis, *Algorithms*, **10**, 2017, Article No. 123.

[6] Chu P. C., Beasley J. E., A Genetic Algorithm for The Multidimensional Knapsack Problem, *Journal of Heuristics*, **4**, 1998, 63-86.

[7] Darwin C., *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*, John Murray Publishers, London, 1859.

[8] Dimopoulos C., Zalzala A. M. S., Recent developments in evolutionary computation for manufacturing optimization: problems, solutions, and comparisons, *IEEE Transactions on Evolutionary Computation*, **4**, 2000, 93-113.

[9] Erisoglu M., Calis N., Sakallioglu S., A new algorithm for initial cluster centers in k-means algorithm, *Pattern Recognition Letters*, **32**, 2011, 1701-1705.

[10] Friedman M., Comparison of Alternative Tests of Significance for the Problem of m Rankings, *Annals of Mathematical Statistics,* **11**, 1940, 86-92.

[11] García J., Crawford B., Soto R., Castro C., Paredes F., A k-means binarization framework applied to multidimensional knapsack problem, *Applied Intelligence*, **48**, 2018, 357-380.

[12] Goh K. S., Lim A., Rodrigues B., Sexual Selection for Genetic Algorithms, *Artificial Intelligence Review*, **19**, 2003, 123-152.

[13] Goldberg D. E., Lingle R., Alleles, Loci, and the Traveling Salesman Problem, in: *Proceedings of the First International Conference on Genetic Algorithms and their Applications*, Los Angeles, US, Lawrence Erlbaum Associates Publishers, 1985, 154-159.

[14] Golfarelli M., Rizzi S., Turricchia E., Sprint Planning Optimization in Agile Data Warehouse Design, in: *Proceedings of the 14th International Conference on Data Warehousing and Knowledge Discovery*, Vienna, Austria, Springer, 2012, 30-41.

[15] Gong G., Deng Q., Chiong R., Gong X., Huang H., An effective memetic algorithm for multi-objective job-shop scheduling, *Knowledge-Based Systems*, **182**, 2019, Article No. 104840.

[16] Gupta I. K., Choubey A., Choubey S., Clustered genetic algorithm to solve multidimensional knapsack problem, in: *Proceedings of the 8th International Conference on Computing, Communication and Networking Technologies*, Delhi, India, IEEE, 2017, 1-6.

[17] Hossain M. Z., Akhtar M. N., Ahmad R. B., Rahman M., A dynamic K-means clustering for data mining, *Indonesian Journal of Electrical Engineering and Computer Science*, **13**, 2019, 521-526.

[18] Juang C. F., A hybrid of genetic algorithm and particle swarm optimization for recurrent network design, *IEEE Transactions on Systems, Man, and Cybernetics*, **34**, 2004, 997-1006.

[19] Karp R. M., Reducibility among Combinatorial Problems, in: R.E. Miller, J.W. Thatcher, J.D. Bohlinger (Eds.), *Complexity of Computer Computations*, Springer, Boston, 1972, 85-103.

[20] Kennedy J., Eberhart R. C., A discrete binary version of the particle swarm algorithm, in: *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*. *Computational Cybernetics and Simulation*, **5**, 1997, 4104-4108.

[21] Khalili-Damghani K., Taghavifard M., Solving a bi-objective project capital budgeting problem using a fuzzy multi-dimensional knapsack, *Journal of Industrial Engineering International*, **7**, 2011, 67-73.

[22] Krishnakumar S., Manivannan K., Effective segmentation and classification of brain tumor using rough K means algorithm and multi kernel SVM in MR images, *Journal of Ambient Intelligence and Humanized Computing*, **12**, 2021, 6751–6760.

[23] Laabadi S., Naimi M., El Amri H., Achchab B., A hybrid genetic algorithm for solving 0/1 Knapsack Problem, in: *Proceedings of the First International Conference on Learning and Optimization Algorithms: Theory and Applications*, Rabat, Morocco, ACM, 2018, Article No. 51.

[24] Laabadi S., Naimi M., El Amri H., Achchab B., The 0/1 Multidimensional Knapsack Problem and Its Variants: A Survey of Practical Models and Heuristic Approaches, *American Journal of Operations Research*, **8**, 2018, 395-439.

[25] Laabadi S., Naimi M., El Amri H., Achchab B., An improved sexual genetic algorithm for solving 0/1 multidimensional knapsack problem, *Engineering Computations*, **36**, 2019, 2260-2292.

[26] Lee Z. J., Su S. F., Chuang C. C., Liu K. H., Genetic algorithm with ant colony optimization (GA-ACO) for multiple sequence alignment, *Applied Soft Computing*, **8**, 2008, 55-78.

[27] MacQueen J., Some Methods for Classification and Analysis of Multivariate Observations, in: *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, San Francisco, US, University of California Press, 1967, 281-297.

[28] Maulik U., Bandyopadhyay S., Genetic algorithm-based clustering technique, *Pattern Recognition*, **33**, 2000, 1455-1465.

[29] Rudolph G., Sprave J., A Cellular Genetic Algorithm with Self-adjusting Acceptance Threshold, in: *Proceedings of the First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*, Sheffield, UK, IEEE, 1995, 365–372.

[30] Santhanam T., Padmavathi M. S., Application of K-Means and Genetic Algorithms for Dimension Reduction by Integrating SVM for Diabetes Diagnosis, *Procedia Computer Science*, **47**, 2015, 76-83.

[31] Sun Y., Xue B., Zhang M., Yen G. G., Lv J., Automatically Designing CNN Architectures Using the Genetic Algorithm for Image Classification, *IEEE Transactions on Cybernetics*, **50**, 2020, 3840-3854.

[32] Torrente A., Romo J., Initializing k-means Clustering by Bootstrap and Data Depth, *Journal of Classification*, **38**, 2021, 232–256.

[33] Umbarkar A. J., Sheth P. D., Crossover Operators in Genetic Algorithms: A Review, *ICTACT Journal on Soft Computing*, **6**, 2015, 1083-1092.

[34] Varnamkhasti M. J., Lee L. S., A Genetic Algorithm Based on Sexual Selection for The Multidimensional 0/1 Knapsack Problems, *International Journal of Modern Physics: Conference Series*, **9**, 2012, 422-431.

[35] Wang L., Pan C., Robust level set image segmentation via a local correntropy-based K-means clustering, *Pattern Recognition*, **47**, 2014, 1917-1925.

[36] Whitley D., Sutton A. M., Genetic Algorithms - A Survey of Models and Methods, in: G. Rozenberg, T. Bäck, J.N. Kok (Eds.), *Handbook of Natural Computing*, Springer, Berlin, 2012, 637-671.

[37] Wilcoxon F., Individual Comparisons by Ranking Methods. *Biometrics Bulletin*, **1**, 1945, 80-83.

[38] Yang H. H, Wang M. T, Yang C. H., A Hybrid of Rough Sets and Genetic Algorithms for Solving the 0-1 Multidimensional Knapsack Problem, *International Journal of Innovative Computing, Information and Control*, **9**, 2013, 3537-3548.

[39] Zouache D., Nouioua F., Moussaoui A., Quantum-inspired firefly algorithm with particle swarm optimization for discrete optimization problems, *Soft Computing*, **20**, 2016, 2781-2799.

Received 3.09.2021, Accepted 12.07.2022