

Solving the 0/1 Knapsack Problem Using a Galactic Swarm Optimization with Data-Driven Binarization Approaches

Camilo Vásquez[1], José Lemus-Romani[1(✉)], Broderick Crawford[1], Ricardo Soto[1], Gino Astorga[2], Wenceslao Palma[1], Sanjay Misra[3], and Fernando Paredes[4]

[1] Pontificia Universidad Católica de Valparaíso, Valparaíso, Chile
{camilo.vasquez.e,jose.lemus.r}@mail.pucv.cl,
{broderick.crawford,ricardo.soto,wenceslao.palma}@pucv.cl
[2] Universidad de Valparaíso, Valparaíso, Chile
gino.astorga@uv.cl
[3] Covenant University, Ota, Nigeria
sanjay.misra@covenantuniversity.edu.ng
[4] Universidad Diego Portales, Santiago, Chile
fernando.paredes@udp.cl

Abstract. Metaheuristics are used to solve high complexity problems, where resolution by exact methods is not a viable option since the resolution time when using these exact methods is not acceptable. Most metaheuristics are defined to solve problems of continuous optimization, which forces these algorithms to adapt its work in the discrete domain using discretization techniques to solve complex problems. This paper proposes data-driven binarization approaches based on clustering techniques. We solve different instances of Knapsack Problems with Galactic Swarm Optimization algorithm using this machine learning techniques.

Keywords: Galactic Swarm Optimization · Metaheuristic · Knapsack problem · K-means · DBSCAN · Machine learning

1 Introduction

Solving Knapsack Problem (KP) using metaheuristics, we provide acceptable solutions in a reasonable time to solve large-scale science and engineering problems, this through an iterative generation process that guides a subordinate heuristic through the combination of different intelligent concepts to explore and exploit the search space, metaheuristics represent a part of the optimization techniques that can be grouped into complete or incomplete techniques. Exact or complete techniques are those in which we find an optimal result independently of the process time instead. On the other hand, incomplete strategies are

© Springer Nature Switzerland AG 2020
O. Gervasi et al. (Eds.): ICCSA 2020, LNCS 12254, pp. 511–526, 2020.
https://doi.org/10.1007/978-3-030-58817-5_38

those where, in a limited processing period, a successful answer (not necessarily the best) is sought. This approach is suited best to the actual circumstances of the problems as solutions to the problems are required in daily life at a certain point. We consider metaheuristics inside provisional or incomplete technologies. We will solve KP using Galactic Swarm Optimization (GSO) which must be adapted to work in discrete domain, using different combinations between transfer techniques and binarization methods, in addition to the K-means and DBScan binarization algorithm to solve complex problems as KP.

The Knapsack Problem, it can be applied modelling, for example, the use of containers in customs, the selection of projects, where each project can be as a container of different items such as: project selection [5], the distribution of resources [12], the problem of network interdiction [13], investment decision making [8], among others.

To solve the KP, the algorithm will be tested with different binarization schemes, the novelty of this work is that we use K-means and DBSCAN, as general mechanisms to binarize continuous metaheuristics. And finally, the results obtained in the tests will be commented.

The remainder of the essay includes the following, Sect. 2 gives a brief explanation on metaheuristics, Sect. 3 defines the problem to be solved, Sect. 4 details the operation of the GSO algorithm, Sect. 5 explains the binarization techniques to be implemented, while Sect. 6 presents the experimental results, ending Sect.7 with the conclusions.

2 Metaheuristics

Metaheuristics is formally described as a method that guides a subordinate heuristic by intelligently combining different concepts to explore and use the space of quest, unlike exact methods, metaheuristics allow to address cases of important problems by offering satisfactory solutions in a reasonable time. There is no guarantee of finding global optimal solutions or even limited solutions. Its use in many applications shows its efficiency and effectiveness to solve large and complex problems. The application of metaheuristics is found in a large number of areas, some are:

- Engineering design, topology optimization and structural optimization in electronics and Very Large Scale Integration (VLSI), aerodynamics, fluid dynamics, telecommunications, automotive and robotics.
- Machine learning and data mining in bioinformatics and computational biology and finance.
- Modelling, simulation and identification of systems in chemistry, physics and biology; control, signal and image processing.
- Routing problem planning, robot planning, programming and production problems, logistics and transportation, supply chain management, environment.

Metaheuristics Classification

- Nature inspired vs. non-nature inspired: In general, metaheuristics may be graded because they are based on the algorithm origin. It considers whether their models were naturally inspired. There are bio-inspired algorithms, such as Crow Search Algorithm (CSA) [11] and Intelligent Water Drops (IWD) [2], and those not inspired by nature, for example Tabu Search (TS) and Local Iterate Search (ILS). This classification is not very significant after hybrid algorithms have been developed.
- Population-based vs. single-point search (trajectory): In such situations, the number of solutions used at the same time is the feature used for the classification. First of all, single-point search algorithms work on a single solution that describes a search space trajectory during the search process. They cover local search-based metaheuristics, like variable neighborhood searchh (VNS), tabu search (TS) and iterated local search (ILS). In the other hand, population-based approaches function on the so-called population-based approach.
- Static vs. dynamic objective function: The metaheuristics with static objective functions, preserve the objective function in the problem during the entire process. However, other algorithms, such as Guided Local Search (GLS) have complex objective features, which change fitness during quest and add data gathered during the quest to avoid local optimums.

GSO and many other metaheuristics are motivated in a vector space $\mathbb{R}^n$, they cannot solve discrete or binary optimization problems. There are a variety of methods proposed which allow the use in discreet or binary problems of metaheuristic real optimization. The following methods are known as discretization if the method allows the real technique to resolve integer problems and is known as binarization if it resolves binaries [1].

3 The 0/1 Knapsack

The Knapsack problem is one of the classic NP-hard optimization problems and the decision problem belongs to the NP-complete class. The decision problem 0/1 knapsack can be defined as follows. Given a set of N objects. Each object O has a specific weight w_j and a specified value u_j. Given a capacity, which is the maximum total weight of the knapsack, and a fee, which is the minimum total value that one wishes to obtain. The decision problem 0–1 knapsack is to find a subset of the objects whose total weight is at most equal to the capacity. The problem is to optimize the objective function:

$$\max f(\chi) = \sum_{j=1}^{n} u_j \chi_j \tag{1}$$

Subject to:

$$\sum_{j=1}^{n} \omega_j \chi_j \leq C \tag{2}$$

$$\chi_j = \begin{cases} 1 & \text{if the article is included} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Binary decision variables x_j are used to indicate whether item j is included in the knapsack or not. It can be assumed that all benefits and weights are positive, A weight w_i and a value χ_j for each item and an integer C (total knapsack capacity), find a subset knapsack with the highest possible value, but that the total weight does not exceed C.

Problem Example. Given a knapsack with a maximum capacity of w and $Values$ items, each with its own value and weight, throw the items into the knapsack so that the final content has the maximum value (Fig. 1).

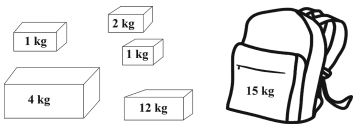


Fig. 1. Example Knapsack problem.

This is the general way in which the problem is explained: consider that a thief enters a house to steal and carries a knapsack. There is a fixed number of items in the home, each with its own weight and value, jewelry, with less weight and the highest value compared to the tables, with less value but much heavier. The thief has a limited capacity knapsack. Obviously, the selection of each item is given by 1/0 indicating whether to take it or leave it, that is:

$$w = (4, 1, 2, 1, 12) \tag{4}$$

Each object has a specific value, which is:

$$Value = (1, 3, 5, 1, 2) \tag{5}$$

The knapsack has a capacity C of 15 kg.

$$C = 15 \tag{6}$$

According to this notation and the formulation, the optimization problem is expressed as:

$$max = \chi_1 + 3\chi_2 + 5\chi_3 + \chi_4 + 2\chi_5 \tag{7}$$

Subject to:

$$4\chi_1 + \chi_2 + 2\chi_3 + \chi_4 + 12\chi_5 \leq 15 \tag{8}$$

where the valid solutions are those that exceed the capacity of the knapsack.

4 Galactic Swarm Optimization Algorithm

Galactic Swarm Optimization is proposed by Venkataraman Muthiah-Nakarajan in 2016 [7]. GSO simulates the movements of stars, galaxies and superclusters in the cosmos. Stars are not distributed evenly in the cosmos but grouped into galaxies that in turn are not evenly distributed. GSO simulates the motion of stars. The GSO algorithm emulates stars in a galaxy, massive masses, and galaxies to certain large masses as follows: individuals in any sub-population that are drawn to better PSO algorithm solutions in the sub-population, any sub-population is a better solution that sub-populations consider and view as superswarm. Each sub-population moving to the PSO algorithm includes superswarm the best solutions.

The swarm and superswarm movement can be achieved since it is population-based, providing multiple exploration and exploitation cycles by dividing the search in terms of offers, providing the algorithm with more opportunities to accurately locate a local minimum, in the first level it is considered the exploratory phase where potential local minimums are identified, the second level of the GSO algorithm is the exploratory phase which uses the best solutions already calculated, by the sub swarms considering the information already calculated in the first level.

The swarm is a group of X D-Tuples that contains ($\mathrm{x}_j^{(i)} \in \mathbb{R}^D$) that consists of M partitions, called subswarms X_i, each of size N, X is randomly initialized within the search space $[x_{min}, x_{max}]^D$, the full frame of the swarm is defined by:

$$X_i \subset X : i = 1, 2, ..., M \tag{9}$$

$$x_j^{(i)} \in X_i : j = 1, 2, ..., N \tag{10}$$

$$X_i \cap X_j : if\ \emptyset\ i \neq j \tag{11}$$

$$\cup_{i=1}^{M} X_i = X \tag{12}$$

X_i is a swarm of size N. The velocity and the best personal relationship with each particle $x_j^{(i)}$ are represented by $V_j^{(i)}$ and $P_j^{(i)}$, respectively. Each subswarm independently scans the search space, the declaration for updating the velocity and the position are:

$$v_j^{(i)} \leftarrow \omega_1 + c_1 r_1 (p_j^{(i)} - x_j^{(i)}) + c_2 r_2 (g^{(i)} - x_j^{(i)}) \tag{13}$$

$$x_j^{(i)} \leftarrow x_j^{(i)} + v_j^{(i)} \tag{14}$$

where the inertial weight ω_1, and the random numbers r_1 and r_2 are given by

$$\omega_1 = 1 - \frac{k}{L_1 + 1} \tag{15}$$

$$r_1 \sim \cup(-1, 1) \tag{16}$$

$$r_2 \sim \cup(-1, 1) \tag{17}$$

k is the current integer iteration number that varies from 0 to L1. The best solutions participate in the next stage of grouping creating a new superswarm X_i.

$$y^{(i)} \in Y : i = 1, 2, ..., M \tag{18}$$

$$y^{(i)} = g^{(i)} \tag{19}$$

In this second grouping stage the velocity $v^{(i)}$ and the position $y^{(i)}$ are updated according to the following expression:

$$v^{(i)} \leftarrow \omega_2 v^{(i)} + c_3 r_3 (p^{(i)} + y^{(i)}) + c_4 r_4 (g - y^{(i)}) \tag{20}$$

$$y^{(i)} \leftarrow y^{(i)} + v^{(i)} \tag{21}$$

$p^{(i)}$ is the best staff in relation to the vector $y^{(i)}$, is defined in relation to ω_2, r_3 and r_4 are similar way as it is done in the first level, g indicates us as the best global and is not updated unless the search finds us a better one and this is indicated as the best overall of the subswarm.

In two steps, the binarization performed according to the corresponding combination is performed in the highlighted segments:

Algorithm 1: GSO + Two steps

```
Level 1 Initialization: x_j^(i), v_j^(i), p_j^(i), g^(i) within [x_min, x_max]^D randomly.
Level 2 Initialization: v^(i), p^(i), g within [x_min, x_max]^D randomly.
for EP ← 1 to EP_max do
    Begin PSO: Level 1
    for i ← 1 to M do
        for k ← 0 to L_1 do
            for j ← 1 to N do
                for d ← 1 to D do
                    v_ij^(d) ← ω_1 v_ij^(d) + c_1 r_1(p_ij^(d) - x_ij^(d)) + c_2 r_2(g^(i) - x_ij^(d));
                    x_ij^(d) ← x_ij^(d) + v_ij^(d);
                    Apply binarization to, x_ij^(d);
                end
                if f(x_j^(i)) < f(p_j^(i)) then
                    p_j^(i) ← x_j^(i)
                    if f(p_j^(i)) < f(g^(i)) then
                        g^(i) ← p_j^(i);
                        if f(g^(i)) < f(g) then
                            g ← g^(i);
                        end
                    end
                end
            end
        end
    end
    Begin PSO: Level 2
    Initialize Swarm y^(i) = g^(i) : 1,2,...,M;
    for k ← 0 to L_2 do
        for i ← 1 to M do
            for d ← 1 to D do
                v_i^(d) ← ω_2 v_i^(d) + c_3 r_3(p_i^(d) - y_i^(d)) + c_4 r_4(g - y_i^(d));
                y_i^(d) ← y_i^(d) + v_i^(d);
                Apply binarization to, y_i^(d);
            end
            if f(y^(i)) < f(p^(i)) then
                p^(i) ← y^(i);
                if f(p^(i)) < f(g) then
                    g ← p^(i);
                end
            end
        end
    end
end
```

When designing and implementing the algorithms corresponding to the clustering technique, it is performed in the following outstanding segments:

Algorithm 2: GSO + Clustering techniques

```
- Level 1 Initialization: x_j^(i), v_j^(i), p_j^(i), g^(i) within [x_min, x_max]^D randomly.
Level 2 Initialization: v^(i), p^(i), g within [x_min, x_max]^D randomly.
for EP ← 1 to EP_max do
    Begin PSO: Level 1
    for i ← 1 to M do
        for k ← 0 to L_1 do
            for j ← 1 to N do
                for d ← 1 to D do
                    v_ij^(d) ← ω_1 v_ij^(d) + c_1 r_1(p_ij^(d) - χ_ij^(d)) + c_2 r_2(g^(i) - χ_ij^(d));
                    χ_ij^(d) ← χ_ij^(d) + v_ij^(d);
                end
            end
            Apply binarization to, χ;
            if f(χ^(i)) < f(p^(i)) then
                p^(i) ← χ^(i)
                if f(p^(i)) < f(g^(i)) then
                    g^(i) ← p^(i);
                    if f(g^(i)) < f(g) then
                        g ← g^(i);
                    end
                end
            end
        end
    end
    Begin PSO: Level 2
    Initialize Swarm y^(i) = g^(i) : 1,2,...,M;
    for k ← 0 to L_2 do
        for i ← 1 to M do
            for d ← 1 to D do
                v_i^(d) ← ω_2 v_i^(d) + c_3 r_3(p_i^(d) - y_i^(d)) + c_4 r_4(g - y_i^(d));
                y_i^(d) ← y_i^(d) + v_i^(d);
            end
        end
        Apply binarization to, y;
        if f(y) < f(p) then
            p ← y;
            if f(p) < f(g) then
                g ← p;
            end
        end
    end
end
```

5 Binarizating Continuous Metaheuristics

When modifying GSO to adapt its work in the discrete domain, different binarization schemes are used, which are described in this section.

5.1 Classic Binarization Techniques

This method operates for continuous operators. In the process of binarisation, the first step refers to the normalization procedure and the second step relates to the binarization scheme, in which a binarization rule is used to turn the object into a binary.

Transfer Functions. Transfer functions define the probability of changing the elements of a position vector from 0 to 1 and vice versa. Transfer functions force particles to move in a binary space. When choosing a transfer function, it should be taken into account to give velocity values as follows:

- The range of a transfer function should be limited in the interval [0,1], since they represent the probability that a particle changes its position.
- For a large absolute velocity evaluation a transfer function must provide a high probability to change the position. Components with strong absolute values of their speeds are definitely not the right way to swap their places in the next iteration.
- A transfer function should also have a small chance of changing the position for a small absolute velocity value.
- The return value of a transfer function should increase as the velocity increases. Particles that are moving away from the best solution should have their chance of changing their position vectors to return to their previous positions.
- The return value of a transfer function should decrease the velocity is reduced.

The 8 transfer functions corresponding to S-Shaped and V-Shaped are shown below (Table 1):

Table 1. $\mathcal{S}$-Shape and $\mathcal{V}$-Shape transfer functions.

$\mathcal{S}$-Shape		$\mathcal{V}$-Shape	
$\mathcal{S}1$:		$\mathcal{V}1$:	
$T(x_i^j)$	$= \frac{1}{1+e^{-2x_i^j}}$	$T(x_i^j)$	$= \left\lvert erf\left(\frac{\sqrt{\pi}}{2}x_i^j\right)\right\rvert$
$\mathcal{S}2$:		$\mathcal{V}2$:	
$T(x_i^j)$	$= \frac{1}{1+e^{2x_i^j}}$	$T(x_i^j)$	$= \left\lvert tanh\left(x_i^j\right)\right\rvert$
$\mathcal{S}3$:		$\mathcal{V}3$:	
$T(x_i^j)$	$= \frac{1}{1+e^{\frac{-x_i^j}{2}}}$	$T(x_i^j)$	$= \left\lvert \frac{x_i^j}{\sqrt{1+[x_i^j]^2}}\right\rvert$
$\mathcal{S}4$:		$\mathcal{V}4$:	
$T(x_i^j)$	$= \frac{1}{1+e^{\frac{-x_i^j}{3}}}$	$T(x_i^j)$	$= \left\lvert \frac{2}{\pi} arctan\left(\frac{\pi}{2}x_i^j\right)\right\rvert$

Binarization Schemes. Together with the transfer techniques the experiments are carried out with 4 methods of binarization, which are shown below where:

- **Standard:** It is the most common method to assign a binary element, because a random variable is responsible for deciding whether its value will be 0 or 1.

$$x_i^j(t+1) = \begin{cases} 1 \text{ if } rand < T\left(x_i^j(t+1)\right) \\ 0 \text{ otherwise} \end{cases} \tag{22}$$

– **Elitist:** In this equation, the value of the particle is assigned if the random value is contained in the probability of said variable, if not, it is assigned 0.

$$x_i^j(t+1) = \begin{cases} x_{best}^j \text{ if rand } < T\left(x_i^j(t+1)\right) \\ 0 \qquad\qquad\qquad\qquad\qquad\quad \text{otherwise} \end{cases} \tag{23}$$

– **Complement:** If the random value is in the range of the probability it is assigned the complement of the value of the particle, in another case, it is assigned zero.

$$x_i^j(t+1) = \begin{cases} \text{complement } \left(x_i^j(t)\right) & \text{if } rand < T\left(x_i^j(t+1)\right) \\ 0 & \text{otherwise} \end{cases} \tag{24}$$

– **Static Probability:** In this equation, the assignment of the particle variable is decided if the probability is less than, it is assigned the best particle until the moment, if the probability is in the range between and $1 = 2\ (1 +)$ it is assigns the best particle of the generation, in any other case, it is maintained.

$$x_i^j(t+1) = \begin{cases} x_i^j(t) & \text{if } 0 < T\left(x_i^j(t+1)\right) \leq \alpha \\ x_i^j(t+1) & \text{if } \alpha < T\left(x_i^j(t+1)\right) \leq \frac{1}{2}(1+\alpha) \\ x_{bcst}^j & \text{if } \frac{1}{2}(1+\alpha) < T\left(x_i^j(t+1)\right) \leq 1 \end{cases} \tag{25}$$

5.2 Binarization Techniques Inspired by Machine Learning

When comes to integrate machine learning and metaheuristics [9,10], two large groups can be mainly indicated. The first group, corresponds metaheuristic techniques improve the performance of machine learning algorithms. The second one covers machine learning algorithms enhance the metaheuristic algorithms quality. For the first group, we find four main areas of application: improving clustering algorithms, feature selection applications, improving classification algorithms and strengthening regression algorithms.

In the case of clustering, a variety of methods to solve it have been reported. One of the main problems that presents a greater algorithmic complexity, the centroid search is the same as the group of objects better studied. Since NP-hard is this problem, provisional methods for solving it have been suggested. There is a long list of studies in this area, however, in recent years the focus is to solve applied problems.

K-Means. A K-means Transition Ranking (KMTR) [3] as a general mechanism to binarize continuous Swarm Intelligence metaheuristics. This algorithm is composed of three operators. The main operator corresponds to the K-means transition operator. This operator translates the velocities obtained the continuous space to transition probabilities in binary spaces (Fig. 2).

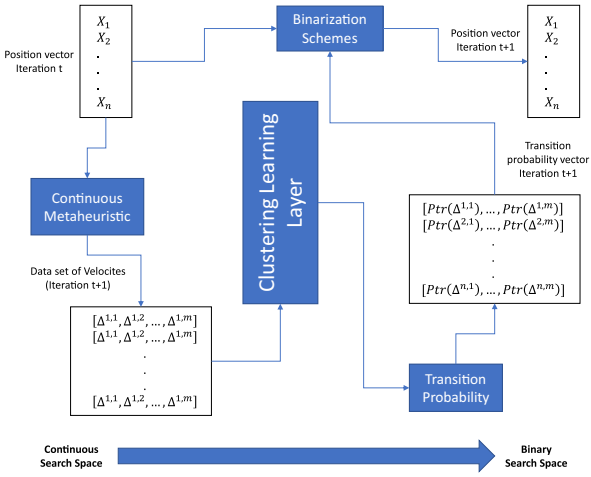


Fig. 2. Mapping the continuous search space to a discrete search space.

This translation is done by performing the clustering of solutions considering the solution velocity module in each dimension as metric. Because the clustering is executed at each iteration, the value of the transition probabilities of each solution is adapted depending on the clusters generated by the K-means Transition operator. In addition to the K-means transition operator, perturbation and repair operators are used.

DBSCAN. DBScan is an unsupervised learning technique that explores with the objective of using it in the binarization process of metaheuristic algorithms of continuous swarm intelligence. The objective of the binary db-scan operator is to group together the various solutions obtained by continuous metaheuristics. We must recognize the role of the particulate as a solution place in the quest space while considering solutions as particles. The velocity represents the vector for the transition from iteration to iteration of the particle $t + 1$.

DBScan utilizes the concept of density in the clustering: DBScan groups points with many close neighbors, given a set of S points in a metric space that marks outskirts those which are alone in low-density regions [4].

6 Experimental Results

To run the solution, instances are generated in Google Cloud Plaftform, which are configured with an Intel (R) Xeon (R) CPU @ 2.30GHz 3.5Gb RAM, 4gb of RAM, with Centos 7 distribution, for the development of the Solution uses Python language in its version 3 and Visual Studio Code and Spyder 3 tools (Fig. 3).

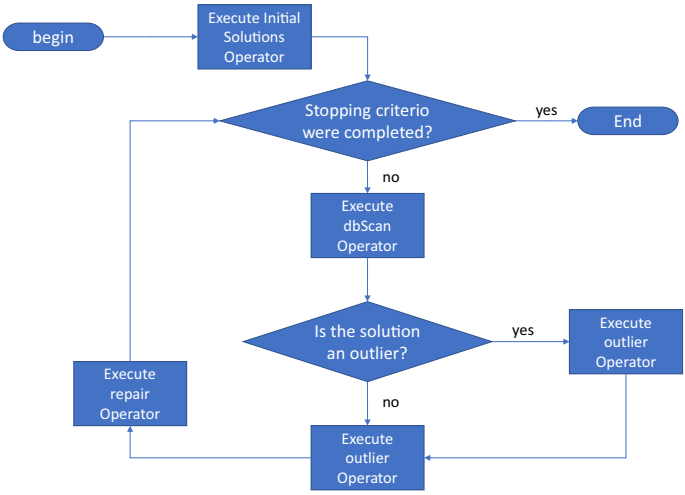


Fig. 3. General flow chart of the binary DBSCAN algorithm.

During the execution of the experiments, GSO has been executed with different two-step methods, K-means and DBSCAN, the instances performed in each problem are shown below.

The input parameters specified in the Table 3 were considered in the experiments conducted. The Table 2 shows the instances executed by the KP.

While the results obtained experimentally are divided in the following tables, Table 4 gives the results of the two-step binarization techniques, Table 5 the results obtained by K-means and Table 6 those corresponding to DBSCAN. The three tables have the same structure, in the first column the instance to be solved is presented, in the following two columns the transfer function and the discretization method that obtained the best performance are shown, in the following columns the optimal value of the instance, the minimum and maximum value found by the algorithm and the binarization technique, the average value and the Relative Percentage Deviation (RPD) defined in the EQ are shown. The average value and the relative percentage deviation (RPD) defined in the Eq. 26, as well as the computation time in seconds, are shown.

In addition, the violin graphs in Fig. 4 are presented, which help to identify the distribution of the data obtained.

$$\text{RPD} = \frac{100\,(Z_{min} - Z_{opt})}{Z_{opt}} \tag{26}$$

Table 2. Knapsack instances.

Instance	Vector	Parameters
f1	10	w = 95, 4, 60, 32, 23, 72, 80, 62, 65, 46; p = 55, 10, 47,5, 4, 50, 8, 61, 85, 87; b = 269
f2	20	w = 92, 4, 43, 83, 84, 68, 92, 82, 6, 44, 32, 18, 56, 83,25, 96, 70, 48, 14, 58; p = 44, 46, 90, 72, 91, 40, 75, 35,8, 54, 78, 40, 77, 15, 61, 17, 75, 29, 75, 63; b = 878
f3	4	w = 6, 5, 9, 7; p = 9, 11, 13, 15; b = 20
f4	4	w = 2, 4, 6, 7; p = 6, 10, 12, 13; b = 11
f5	15	w = 56.358531, 80.87405, 47.987304, 89.59624, 74.660482, 85.894345, 51.353496, 1.498459, 36.445204, 16.589862, 44.569231, 0.466933, 37.788018, 57.118442, 60.716575; p = 0.125126,19.330424, 58.500931, 35.029145, 82.284005, 17.41081, 71.050142, 30.399487, 9.140294, 14.731285, 98.852504, 11.908322, 0.89114, 53.166295, 60.176397; b = 75
f6	6	w = 30, 25, 20, 18, 17, 11, 5, 2, 1, 1; p = 20, 18, 17, 15,15, 10, 5, 3, 1, 1; b = 60
f7	7	w = 31, 10, 20, 19, 4, 3, 6; p = 70, 20, 39, 37, 7, 5,10; b = 50
f8	23	w = 983, 982, 981, 980, 979, 978, 488, 976, 972, 486, 486, 972, 972, 485, 485, 969, 966, 483, 964, 963, 961,958, 959; p = 81,980, 979, 978, 977, 976, 487, 974,970, 85, 485, 970, 970, 484, 484, 976, 974, 482, 962,961, 959, 958, 857; b = 10000
f9	5	w = 15, 20, 17, 8, 31; p = 33, 24, 36, 37, 12; b = 80
f10	20	w = 84, 83, 43, 4, 44, 6, 82, 92, 25, 83, 56, 18, 58, 14,48, 70, 96, 32, 68, 92; p = 91, 72, 90, 46, 55, 8, 35, 75,61, 15, 77, 40, 63, 75, 29, 75, 17, 78, 40, 44; b = 879

Table 3. Parameters used in GSO.

M	N	L1	L2	EPmax	c1 = c2 = c3 = c4
10	5	50	250	3	2,05

Table 4. KP - Two steps results

Instance	Step 1	Step 2	Optimo	Min	Max	Avg	RPD	Time
f1	sShape1	Elitist	295	284	295	293.06	0	0.69
f2	sShape1	Elitist	1024	894	1024	986.32	0.0	1.25
f3	sShape1	Complement	35	24	35	32.35	0	0.38
f4	sShape1	Complement	23	22	23	22.9	0	0.37
f5	sShape1	Elitist	481.069443	3.814.508	481.069.368	468.13	1,00E-07	1.0
f6	sShape1	Complement	52	39	52	47.32	0	0.7
f7	sShape1	Complement	107	90	107	105.9	0	0.57
f8	sShape1	Elitist	9767	9731	9756	9747.19	11	1.49
f9	sShape1	Complement	130	73	130	103.87	0	0.42
f10	sShape1	Elitist	1025	876	1025	989.74	0	1.28

Table 5. KP - K-means results

Instance	Optimo	Min	Max	Avg	RPD	Time
f1	295	295	**295**	295	0	422,51
f2	1024	1009	**1024**	1021,77	0	420,92
f3	35	35	**35**	35	0	286,95
f4	23	23	**23**	23	0	300,93
f5	481,0694	44,507,889	481,069,368	479,73	1,00E-07	420,86
f6	52	52	**52**	52	0	424,6
f7	107	107	**107**	107	0	413,62
f8	9767	9741	9756	9748,29	11	437
f9	130	130	**130**	130	0	349,82
f10	1025	1010	**1025**	1024,32	0	434,25

Table 6. KP - DBSCAN results

Instance	Optimo	Min	Max	Avg	RPD	Time
f1	295	295	**295**	295	0	108,52
f2	1024	933	**1024**	984,03	0	283
f3	35	35	**35**	35	0	35,68
f4	23	23	**23**	23	0	39,44
f5	481,0694	469,161,046	481,069,368	479,02	1,00E-07	183,49
f6	52	52	**52**	52	0	115,05
f7	107	107	**107**	107	0	70,84
f8	9767	9737	9755	9747,16	12	353,36
f9	130	130	**130**	130	0	46,86
f10	1025	935	1017	975,87	78	275,36



Fig. 4. Violin charts

6.1 KP Instance Distribution

It can be seen that in the figures corresponding to the instance distribution corresponding to KP, the great part of the experiments performed for the different instances shows promising results, K-means and DBScan have an identical behavior, obtaining in the majority good results.

Table 7. p-value Mann-Whitney-Wilcoxon test, KP - Two steps vs K-means.

Instance	Two steps vs K-means	K-means vs Two steps
f1	0.5	0.5
f2	1,0868E-03	1,0000E+00
f3	0.5	0.5
f4	0.5	0.5
f5	0.0231855071	0.976814493
f6	0.5	0.5
f7	0.5	0.5
f8	0.177609742	0.822390258
f9	0.5	0.5
f10	5,4801E-05	1,0000E+00

Table 8. p-value Mann-Whitney-Wilcoxon test, KP - Two steps vs DBSCAN.

Instance	Two steps vs DBSCAN	DBSCAN vs Two steps
f1	0.99967722	0.000322779755
f2	8,9637E+03	0.999910363
f3	0.999985704	1,43E+03
f4	0.841344746	0.158655254
f5	0.986436334	0.0135636663
f6	1,0000E+00	7,82E-05
f7	0.960840275	0.039159725
f8	0.000702254007	0.999297746
f9	0.999999997	2,95E-01
f10	0.999999892	1,08E+01

KP - Statical Test The general average of the 13,020 experiments performed to solve KP in Table 10.

A general comparison is made of the averages in Tables 7, 8 and 9 obtained in all the experiments performed, in this comparison all the experiments of the binarization schemes are included. KP corresponds to a maximization problem, in this case the average of the results obtained are similar and the best of them is DBScan, for both solved problems the best averages correspond to DBScan.

Table 9. p-value Mann-Whitney-Wilcoxon test, KP - K-means vs DBSCAN.

Instance	K-means vs DBSCAN	DBSCAN vs K-means
f1	0.99967722	0.000322779755
f2	0.999832952	0.000167047508
f3	0.999985704	1,4296E+03
f4	0.841344746	0.158655254
f5	0.999886089	0.000113910952
f6	1,0000E+00	7,8204E-05
f7	0.960840275	0.039159725
f8	0.00895109511	0.991048905
f9	0.999999997	2,9483E-01
f10	0.999999892	1,0812E+01

Table 10. Average results KP

Two steps	K-means	DbScan
1212,40	1291,60	1282,80

7 Conclusion

By modifying the GSO to solve the classical KP optimization problem using different binarization schemes considering two steps as well as new clustering techniques, different analyses and comparisons based on the results and statistical tests are obtained.

It can be seen that in the figures corresponding to the distribution of instances corresponding to KP, most of the experiments performed for the different instances show promising results, K-means and DBSCAN have an identical behavior, obtaining mostly good results. In addition, it should be considered that the use of K-means and DBSCAN allows to find viable solutions without the need to perform a large number of experiments as required by the two steps.

In addition to carrying out hypothesis testing in our report for the comparison in problem and solved instances, a general comparison of the averages obtained in all the experiments carried out in the different binarization schemes was made. Considering that KP corresponds to a maximization problem, in this case the average of the results obtained is similar and the best of them is the DBSCAN, for both solved problems the best averages based on proficiency correspond to the DBSCAN.

In a future work we want to investigate the behavior of other metaheuristics with the use of clustering techniques already used to solve NP-Hard problems and also include autonomous search that explore new variations between operators exploration and exploitation properties. Along with identifying metrics and

quality indicators of the exploration and exploitation of the search space, which are fundamental for the correct functioning of the techniques [6].

Acknowledgements. Broderick Crawford is supported by Grant CONICYT/FONDECYT/REGULAR/1171243, Ricardo Soto is supported by Grant CONICYT/FONDECYT/REGULAR/1190129. José Lemus-Romani is supported by National Agency for Research and Development (ANID)/Scholarship Program/DOCTORADO NACIONAL / 2019 - 21191692.

References

1. Crawford, B., Soto, R., Astorga, G., García, J., Castro, C., Paredes, F.: Putting continuous metaheuristics to work in binary search spaces. Complexity **2017**(2), 1–19 (2017)
2. Crawford, B., Soto, R., Astorga, G., Lemus-Romani, J., Misra, S., Rubio, J.-M.: An adaptive intelligent water drops algorithm for set covering problem. In: 2019 19th International Conference on Computational Science and Its Applications (ICCSA), pp. 39–45. IEEE (2019)
3. García, J., Crawford, B., Soto, R., Astorga, G.: A clustering algorithm applied to the binarization of swarm intelligence continuous metaheuristics. Swarm Evol. Comput. **44**, 646–664 (2019)
4. García, J., et al.: A Db-scan binarization algorithm applied to matrix covering problems. Comput. Intell. Neurosci. **2019**, 3238516–3238572 (2019)
5. Mavrotas, G., Diakoulaki, D., Kourentzis, A.: Selection among ranked projects under segmentation, policy and logical constraints. Eur. J. Oper. Res. **187**(1), 177–192 (2008)
6. Morales-Castañeda, B., Zaldívar, D., Cuevas, E., Fausto, F., Rodríguez, A.: A better balance in metaheuristic algorithms: does it exist? Swarm Evol. Comput. **54**, 100671 (2020)
7. Muthiah-Nakarajan, V., Noel, M.M.: Galactic swarm optimization: a new global optimization metaheuristic inspired by galactic motion. Appl. Soft Comput. **38**, 771–787 (2016)
8. Peeta, S., Salman, F.S., Gunnec, D., Viswanath, K.: Pre-disaster investment decisions for strengthening a highway network. Comput. Oper. Res. **37**(10), 1708–1719 (2010)
9. Song, H., Triguero, I., Özcan, E.: A review on the self and dual interactions between machine learning and optimisation. Prog. Artif. Intell. **8**(2), 143–165 (2019). https://doi.org/10.1007/s13748-019-00185-z
10. Talbi, E.-G.: Machine learning into metaheuristics: a survey and taxonomy of data-driven metaheuristics, Working paper or preprint, June 2020
11. Valdivia, S., et al.: Bridges reinforcement through conversion of tied-arch using crow search algorithm. In: Misra, S., et al. (eds.) ICCSA 2019. LNCS, vol. 11623, pp. 525–535. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-24308-1_42
12. Vanderster, D.C., Dimopoulos, N.J., Parra-Hernandez, R., Sobie, R.J.: Resource allocation on computational grids using a utility model and the knapsack problem. Future Gener. Comput. Syst. **25**(1), 35–50 (2009)
13. Yates, J., Lakshmanan, K.: A constrained binary knapsack approximation for shortest path network interdiction. Comput. Indu. Eng. **61**(4), 981–992 (2011)