Expert Systems With Applications 70 (2017) 67–82

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Analyzing the effects of binarization techniques when solving the set covering problem through swarm optimization

Jose M. Lanza-Gutierrez[a,*], Broderick Crawford[b], Ricardo Soto[b], Natalia Berrios[b], Juan A. Gomez-Pulido[a], Fernando Paredes[c]

[a] University of Extremadura, School of Technology, Campus Universitario s/n, 10003 Cáceres, Spain
[b] Pontificia Universidad Católica de Valparaíso, 2362807 Valparaíso, Chile
[c] Universidad Diego Portales, Escuela de Ingeniería Industrial, 8370109 Santiago, Chile

ARTICLE INFO

Article history:
Received 19 April 2016
Revised 25 October 2016
Accepted 25 October 2016
Available online 5 November 2016

Keywords:
Binarization technique
Cat swarm optimization
Continuous optimization
Discrete optimization
Set covering problem
Swarm intelligence

ABSTRACT

The Set Covering Problem (SCP) is one of the classical Karp's 21 NP-complete problems. Although this is a traditional optimization problem, we find many papers assuming metaheuristics for solving the SCP in the current literature. However, while the SCP is a discrete problem, most metaheuristics are defined for solving continuous optimization problems, specially Swarm Intelligence Algorithms (SIAs). Hence, such algorithms should be adapted for working on the discrete scope, but most authors did not perform any study to select a concrete binarization approach. This situation might lead to the conclusion that selecting a concrete binarization technique does not influence the behavior of the algorithm, but rather the general approach of the metaheuristic. This circumstance led us to write this paper focusing on the inherent difficulty in binarization of metaheuristics designed for continuous optimization, when solving a discrete optimization problem, concretely the SCP. To this end, we consider a recent SIA inspired by the behavior of cats and adapted to the discrete scope, which is called Binary Cat Swarm Optimization (BCSO). We replace the binarization technique assumed in the original BCSO by forty different approaches from the current literature. The results obtained while solving a standard SCP benchmark are analyzed through a widely accepted statistical method, concluding that it is crucial to select an adequate binarization approach to ensure that the solving algorithm reaches its full potential. Thus, the task of adapting a metaheuristic to the discrete scope is not a simple matter and should be carefully studied. To this end and as a result of this study, we give some recommendations to perform this task.

1. Introduction

The Set Covering Problem (SCP) is one of the classical 21 problems shown to be NP-complete by Karp (1972) and whose optimization version is NP-hard (Garey & Johnson, 1979). Although the SCP is a traditional optimization problem, it is widely considered in the current literature for designing expert systems, which emulate the decision-making ability of human experts in a given field (Reggia, Nau, & Wang, 1983). For example, we find works considering the SCP for facility location (Farahani, Asgari, Heidari, Hosseininia, & Goh, 2012), ship scheduling (de Mare, Spliet, & Huisman, 2014), production planning (Adulyasak, Cordeau, & Jans, 2015), crew scheduling (Chen & Shen, 2013; Juette & Thonemann, 2012), vehicle routing (Bai, Xue, Chen, & Roberts, 2015; Cacchiani, Hemmelmayr, & Tricoire, 2014; Vidal, Crainic, Gendreau, & Prins, 2013), musical composition (Simeone, Nouno, Mezzadri, & Lari, 2014), information retrieval (Zhang, Wei, & Chen, 2014), and territory design (Elizondo-Amaya, Rios-Mercado, & Diaz, 2014), among many others.

Chvatal (1979) defined the SCP as follows. Given a set M of m objects and a collection S of n sets of these objects, each set with a non-negative cost associated. The goal is to find a minimum cost family of subsets $C \subseteq S$, such that each element $i \in M$ belongs to at least one subset of the family C.

Some authors solved the SCP by applying exact techniques, such as branch-and-bound and branch-and-cut algorithms. However, such methods are not recommended for solving this type of complex problems, because computational times rise exponentially with the problem dimension.

Instead, approximate techniques should be considered, such as metaheuristics. This type of techniques is successfully considered in the literature for solving NP-hard problems from different fields,

* Corresponding author.

E-mail addresses: jmlanza@unex.es (J.M. Lanza-Gutierrez), broderick.crawford@pucv.cl (B. Crawford), ricardo.soto@pucv.cl (R. Soto), natalia.berrios.p@mail.pucv.cl (N. Berrios), jangomez@unex.es (J.A. Gomez-Pulido), fernando.paredes@udp.cl (F. Paredes).

http://dx.doi.org/10.1016/j.eswa.2016.10.054
0957-4174/© 2016 Elsevier Ltd. All rights reserved.

including the SCP (Dasgupta & Michalewicz, 2013). However, while the SCP is defined as a discrete optimization problem, many metaheuristics are designed for solving continuous optimization problems, specially Swarm Intelligence Algorithms (SIAs). Thus, such metaheuristics should be adapted for working on the discrete search space.

There are many SIAs adapted for solving general discrete optimization problems, such as Binary Gravitational Search Algorithm (BGSA) (Rashedi, Nezamabadi-Pour, & Saryazdi, 2010), Binary Magnetic Optimization Algorithm (BMOA) (Mirjalili & Hashim, 2012), and Binary Cat Swarm Optimization (BCSO) (Sharafi, Khanesar, & Teshnehlab, 2013). Usually, the algorithms are adapted by following the two-step binarization method introduced by Kennedy and Eberhart (1997) in their approach of Binary Particle Swarm Optimization (BPSO) for transforming real numbers into binary ones. In this case, the authors explained how to get a new binary solution according to the particle velocity, which is a real number. The method followed by the authors is as follows. Firstly, we map the real value to a number in the interval [0, 1] through a transfer function. Secondly, we transform the number in the interval [0, 1] into a binary value through a discretization function. In this line, there are eight major transfer functions and five major discretization functions in the current literature, denoted as $S_1, S_2, \ldots, S_4, V_1, V_2, \ldots, V_4$ and $D_1, D_2, \ldots, D_8$, respectively (Crawford et al., 2015c; Mirjalili & Lewis, 2013).

Most authors did not do any study to select a concrete binarization approach when adapting a metaheuristic. This situation might lead to the conclusion that selecting a concrete binarization technique does not influence the behavior of the algorithm, but rather the general approach of the metaheuristic. To the best of our knowledge, this is the first work performing this study in the literature. We do the following three main tasks throughout this study:

- We select a recent SIA from the current literature, which was initially designed for continuous optimization and later adapted to the discrete scope. Specifically, the BCSO algorithm inspired by the behavior of cats, whose original continuous approach was proposed by Chu, Tsai, and Pan (2006).
- The authors of BCSO considered a transfer and discretization function, without performing any formal study to this task. We change the original formulation of BCSO by combining the eight transfer functions and the five discretization functions introduced before, *i.e.*, we get forty BCSO approaches.
- We apply the forty BCSO approaches for solving two freely available SCP sets. We study the results obtained through an accepted statistical methodology to analyze if selecting a binarization technique influences the behavior of the metaheuristic.

The rest of this paper is structured as follows. We list the acronyms considered in Table 1. In Section 2, we discuss the related work, including the major motivations for performing this work. In Section 3, we give a formal SCP definition, including a problem example. In Section 4, we explain the BCSO metaheuristic. In Section 5, we describe the transfer and discretization functions in this study. In Section 6, we discuss the experimental method followed and the results obtained. In Section 7, we give some implementation details. Finally, conclusions are left for Section 8.

2. Related work

We find many works solving the SCP. Starting with exact algorithms, (Beasley & Jornsten, 1992; Fisher & Kedia, 1990), and (Balas & Carrera, 1996) considered branch-and-bound and branch-and-cut techniques. Bar-Yejuda and Even (1981); Beasley (1987), and El-Darzi and Mitra (1990) considered linear programming.

Table 1
Acronyms.

ACO	Ant Colony Optimization
ABC	Artificial Bee Colony
BCSO	Binary Cat Swarm Optimization
BGSA	Binary Gravitational Search Algorithm
BMOA	Binary Magnetic Optimization Algorithm
BPSO	Binary Particle Swarm Optimization
EA	Evolutionary Algorithm
EM-like	ElectroMagnetism-like
FA	Firefly Algorithm
FFOA	Fruit Fly Optimization Algorithm
RPD	Relative Percentage Deviation
SCP	Set Covering Problem
SFLA	Shuffled Frog Leaping Algorithm
SIA	Swarm Intelligence Algorithm
TA	Trajectory Algorithm
TLBO	Teaching-Learning-Based Optimization
RVP	Relative Variation Percentage

Caprara, Fischetti, and Toth (2000) compared several exact algorithms for optimizing the SCP, reaching that the best exact algorithm is CPLEX. This type of exact techniques is not suitable for solving NP-hard problems, because no polynomial time algorithm exists. Instead, computation times raise exponentially. In this way, exact algorithms can only solve SCP instances of a limited size and are time-consuming.

On the contrary, approximate algorithms as heuristics and metaheuristics sacrifice getting optimal solutions for the sake of obtaining approximate ones in an appreciably reduced computational time. In this line, there are works comparing exact techniques to metaheuristics, concluding that exact methods were able to optimally solve small instances in a reduced time, even lower than metaheuristics. However and for large instances, metaheuristics outperformed exact techniques in many cases, keeping low computational times. In many other cases, exact techniques were unable to be applied because of the computational effort needed (Grohmann, Urosevic, Carrizosa, & Mladenovic, 2016; Niroomand & Vizvari, 2015).

We may cite some authors considering heuristics for solving the SCP. Chvatal (1979) assumed a classical greedy algorithm and (Vasko, Lu, & Zyma, 2016) analyzed the performance of construction heuristics expanding on column and row knowledge classical functions. Some authors improved greedy algorithms by adding some randomness, *e.g.*, Feo and Resende (1989); Vasko (1984), and Haouari and Chaouachi (2002). Other authors considered heuristics based on Lagrangian relaxation, such as Beasley (1990); Ceria, Nobili, and Sassano (1998), and Caprara, Fischetti, and Toth (1999).

In the last decades, metaheuristics had a great impact, combining basic heuristic methods and effectiveness exploring the search space. Usually, metaheuristics are split into three categories: Evolutionary Algorithms (EAs), SIAs, and Trajectory Algorithms (TAs).

Starting with EAs, we may cite the works of Beasley and Chu (1996); Chu and Beasley (1997), and Aickelin (2004). TAs were assumed by Lust and Tuyttens (2014) and Colombo, Cordone, and Lulli (2015).

In the last years and as discussed below, SIAs were successfully considered in the literature for solving the SCP. As we know, most of these algorithms are designed for solving continuous optimization problems and the SCP is a discrete problem. Consequently, such techniques should be adapted for solving the problem. Next, we review authors assuming SIAs, detailing the adaptation procedure considered if necessary.

Crawford and Castro (2006); Lessing, Dumitrescu, and Stutzle (2004); Ren, Feng, Ke, and Zhang (2010), and Crawford, Soto, Monfroy, Paredes, and Palma (2011) considered the Ant Colony Optimization (ACO) algorithm proposed by Dorigo and Gambardella (1997). This metaheuristic is one of the few SIAs, which solve both discrete and continuous problems without modifying the original proposal.

Sundar and Singh (2012a) and Crawford, Soto, Cuesta, and Paredes (2014a) applied the Artificial Bee Colony (ABC) algorithm proposed by Karaboga and Basturk (2007). In both works, the authors considered the approach proposed by Singh (2009) for generating neighboring solutions instead of assuming the original continuous formulation of ABC. To this end, a part is randomly dropped from the solution and in its place, another different part is randomly selected from other solution in the population. In the case that a different part could not be found, the algorithm generates a new solution through a low-cost local search.

Crawford et al. (2014b) and Crawford et al. (2015d) assumed the Firefly Algorithm (FA) designed by Yang (2010). In both works, they changed the original proposal by incorporating transfer and discretization functions. Thus, Crawford et al. (2015d) assumed all the transfer and discretization functions in this paper; and Crawford et al. (2014b) applied the eight transfer functions and the discretization functions D_2, D_3, and D_4.

Naji-Azimi, Toth, and Galli (2010) and Soto, Crawford, Muñoz, Johnson, and Paredes (2015) applied the ElectroMagnetism-like (EM-like) algorithm proposed by Birbil and Fang (2003). Naji-Azimi et al. (2010) considered a new binary formulation and Soto et al. (2015) assumed the transfer function S_1 and the discretization function D_4.

Crawford et al. (2015c) considered the Shuffled Frog Leaping Algorithm (SFLA) designed by Eusuff, Lansey, and Pasha (2006), assuming all the transfer and discretization functions in this paper.

Crawford et al. (2015a) and Lu and Vasko (2015) assumed the Teaching-Learning-Based Optimization (TLBO) algorithm proposed by Rao, Savsani, and Vakharia (2011). Crawford et al. (2015a) applied all the transfer and discretization functions in this paper. Lu and Vasko (2015) combined a randomized greedy algorithm for generating the initial population and a TLBO approach for improving the population solutions. To this end, Lu and Vasko (2015) adapted the TLBO metaheuristic to the binary scope through a straightforward methodology. Zyma, Lu, and Vasko (2015) successfully considered the same binarization technique to solve multiple-choice multidimensional knapsack problems.

Crawford et al. (2015e) applied the Fruit Fly Optimization Algorithm (FFOA) designed by Pan (2012). The authors assumed all the transfer and discretization functions in this paper.

Analyzing this review from the point of view of the results obtained, while solving the SCP, we check that both exact algorithms and simple heuristics typically do not give as good results as the more advanced metaheuristics, thereby the best historical results were found by SIAs, obtaining optimal or near optimal solutions for the classical SCP benchmark. Thus, we have an idea of the benefits of solving the SCP by an approximate approach. In this line, Lu and Vasko (2015) achieved a really good approach because of the merit of the simple binarization technique considered for adapting the TLBO algorithm, which is totally different from the 40 techniques discussed in this paper. This is obvious because when Crawford et al. (2015a) applied the same SIA with the 40 transfer and discretization functions in this paper, their results were much poorer (3.02% deviation from optimum based on best over 30 trials) than the high-quality results of Lu and Vasko (2015) (0.06% deviation from optimum based on best over 20 trials) on the same 65 SCPs.

The purpose of this work is not to outperform the results obtained in earlier works nor providing a new metaheuristic. Instead, our main goal is to analyze the impact of binarization methods when adapting an SIA to the discrete scope. To this end, we select an SIA previously applied for solving the SCP, getting results not really close to the optimal ones, such as the BCSO algorithm in Crawford et al. (2015b). If we select a really good approach as the proposal in Lu and Vasko (2015), the results obtained could not be improved, on the contrary, they could only get worse. However, if we select a proposal, which could be improved, such as the BCSO algorithm, we could analyze if the results obtained get better or worse according to a binarization technique or another.

Next, we list the main expected benefits of performing this work:

- We expect to check if selecting a binarization approach could influence the behavior of a metaheuristic.
- If we confirm the first statement, we expect to check if there is any binarization approach which we could recommend for solving the SCP, or instead, if we could give some general recommendations to adapt a metaheuristic.
- If we confirm the first statement, we expect to check if the original BCSO could be outperformed by considering a new binarization approach, and in such a case, if we could recommend the new algorithm for solving the SCP in future works.

To the best of our knowledge, this is the first work performing this study. From the previous literature review, we know that Crawford et al. (2015a, 2015c, 2015d), and Crawford et al. (2015e) assumed several binarization techniques for solving the SCP. However, they did not perform any study to analyze if the differences observed between the b methods were significant, which is the main purpose of this work.

3. Set covering problem

In this section, we provide a formal statement of the SCP and a small location problem as an example of this formulation.

3.1. Problem definition

The SCP is formally defined by assuming a binary matrix A of m-rows and n-columns, where $a_{i,j} \in \{0, 1\}$ denotes the value of the cell (i, j) of A, with $i \in 1, 2, \ldots, m$ and $j \in 1, 2, \ldots, n$. A is formally defined as

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \ldots & a_{1,n} \\ a_{2,1} & a_{2,2} & \ldots & a_{2,n} \\ \ldots & \ldots & \ldots & \ldots \\ a_{m,1} & a_{m,2} & \ldots & a_{m,n} \end{pmatrix}. \tag{1}$$

We say that a column j covers a row i if $a_{i,j}$ equals 1 and 0 otherwise. Each column j is associated with a non-negative real cost $c_j \in C$, where $C = \{c_1, c_2, \ldots, c_n\}$. Let $I = \{1, 2, \ldots, m\}$ and $J = \{1, 2, \ldots, n\}$ be the row and column sets, respectively. The SCP calls for a minimum cost subset $S \subseteq J$, such that each row $i \in I$ is covered by at least one column $j \in S$. Thus, the optimization problem is expressed as

$$min \sum_{j \in J} c_j x_j, \tag{2}$$

subject to

$$\sum_{j \in J} a_{i,j} x_j \geq 1, \quad \forall i \in I, \tag{3}$$

$$x_j \in \{0, 1\}, \quad \forall j \in J, \tag{4}$$

where x_j equals 1 if column j is in the solution S and 0 otherwise. From this formulation, we reach that the goal is to minimize the sum of the costs of the selected columns. Note that the constraint in Eq. (3) ensures that each row i is covered by at least one column.

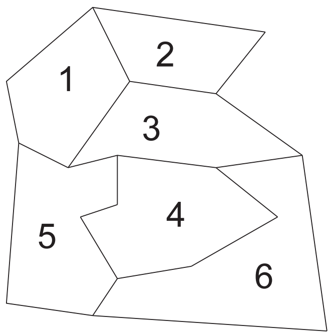


Fig. 1. Zones for the location problem example.

3.2. Problem example

We propose a small location problem as an example of the SCP formulated before. Suppose that we need to offer fire services at the lowest possible cost in the city composed of six zones as shown in Fig. 1. In this case, we have the following constraints:

- A fire station can only attend the zone in which it is located and immediately adjacent zones, *e.g.*, a fire station in zone 1 can attend zones 1, 2, 3, and 5.
- The fire services should attend all the zones.
- The greatest number of fire stations per zone is 1.

Let A be the binary matrix denoting which zones are covered by a hypothetical fire station according to the zone in which it is placed, that is

$$A = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}, \qquad (5)$$

where $a_{i,\,j}$ is the value of the cell $(i,\,j)$ of A, equaling 1 if the fire station at the jth zone covers the ith zone, with i and $j \in 1, 2, \ldots, 6$. Let x_j be the indicator function equaling 1 if a fire station is built in the jth zone and 0 otherwise. Let C be the set denoting the cost of building a fire station according to the zone, that is

$$C = (\ 3,\ 5,\ 6,\ 4,\ 2,\ 1\ ) \quad .$$

According to this notation and the formulation in Eqs. (2) to (4), the optimization problem is expressed as

$$min \quad 3\ x_1 + 5\ x_2 + 6\ x_3 + 4\ x_4 + 2\ x_5 + 1\ x_6,$$

subject to

$$\begin{array}{l} a_{1,1}\ x_1 + a_{1,2}\ x_2 + a_{1,3}\ x_3 + a_{1,4}\ x_4 + a_{1,5}\ x_5 + a_{1,6}\ x_6 \geq 1 \\ a_{2,1}\ x_1 + a_{2,2}\ x_2 + a_{2,3}\ x_3 + a_{2,4}\ x_4 + a_{2,5}\ x_5 + a_{2,6}\ x_6 \geq 1 \\ a_{3,1}\ x_1 + a_{3,2}\ x_2 + a_{3,3}\ x_3 + a_{3,4}\ x_4 + a_{3,5}\ x_5 + a_{3,6}\ x_6 \geq 1 \\ a_{4,1}\ x_1 + a_{4,2}\ x_2 + a_{4,3}\ x_3 + a_{4,4}\ x_4 + a_{4,5}\ x_5 + a_{4,6}\ x_6 \geq 1 \\ a_{5,1}\ x_1 + a_{5,2}\ x_2 + a_{5,3}\ x_3 + a_{5,4}\ x_4 + a_{5,5}\ x_5 + a_{5,6}\ x_6 \geq 1 \\ a_{6,1}\ x_1 + a_{6,2}\ x_2 + a_{6,3}\ x_3 + a_{6,4}\ x_4 + a_{6,5}\ x_5 + a_{6,6}\ x_6 \geq 1 \end{array}, \qquad (6)$$

$$x_j \in \{0, 1\}, \quad \forall j \in 1, \ldots, 6 \quad .$$

Eq. (6) is simplifiable by replacing the values of $a_{i,\,j}$ in Eq. (5) based on the notation in Eq. (1), that is

$$\begin{array}{l} 1\ x_1 + 1\ x_2 + 1\ x_3 + 0\ x_4 + 1\ x_5 + 0\ x_6 \geq 1 \\ 1\ x_1 + 1\ x_2 + 1\ x_3 + 0\ x_4 + 0\ x_5 + 0\ x_6 \geq 1 \\ 1\ x_1 + 1\ x_2 + 1\ x_3 + 1\ x_4 + 1\ x_5 + 1\ x_6 \geq 1 \\ 0\ x_1 + 0\ x_2 + 1\ x_3 + 1\ x_4 + 1\ x_5 + 1\ x_6 \geq 1 \\ 1\ x_1 + 0\ x_2 + 1\ x_3 + 1\ x_4 + 1\ x_5 + 1\ x_6 \geq 1 \\ 0\ x_1 + 0\ x_2 + 1\ x_3 + 1\ x_4 + 1\ x_5 + 1\ x_6 \geq 1 \end{array} \quad . \qquad (7)$$

Excluding the null terms in Eq. (7), the final expression is given by

$$\begin{array}{l} x_1 + x_2 + x_3 + x_5 \geq 1 \\ x_1 + x_2 + x_3 \geq 1 \\ x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \geq 1 \\ x_3 + x_4 + x_5 + x_6 \geq 1 \\ x_1 + x_3 + x_4 + x_5 + x_6 \geq 1 \\ x_3 + x_4 + x_5 + x_6 \geq 1 \end{array} \quad .$$

The optimal solution to this problem is to build a fire station in zones 1 and 6, getting a total cost of 4.

4. Binary cat swarm optimization

Domestic cats show great abilities for hunting and being alert to possible dangers (Adamec, 1976; Adler, 1995). Based on this idea, Chu et al. (2006) proposed an SIA inspired by the behavior of real cats. This metaheuristic was later adapted to the discrete scope by Sharafi et al. (2013). The authors identified two main modes of behavior for simulating cats:

- Seeking mode: it simulates the situation in which a cat is looking around, calculating, and evaluating the next movement.
- Tracing mode: it simulates the situation in which a cat is hunting through tracking targets.

The metaheuristic considers a population Q_g of k cats, in which at generation $g \geq 1$, a cat is a solution to the problem. Based on the SCP definition in Section 3, each cat $\eta \in Q_g$ is composed of a binary vector of n elements, where η_j denotes the value of the jth cell of η, with $j \in 1, 2 \ldots, n$. The meaning of η_j is the same as described before for x_j, that is

$$\eta_j = \begin{cases} 1 & \text{if the } j\text{th column is in the solution of } \eta \\ 0 & \text{otherwise} \end{cases} \quad . \qquad (8)$$

As stated in Algorithm 1, Q_g is randomly generated in a first step (line 1). Then and so long as a stop condition is not reached (line 2), Q_g is randomly divided into two parts S_g and T_g according to the value of mr, which denotes the percentage of cats in tracing mode. Accordingly, the other cats are in seeking mode (line 3). Next, the cats in S_g act in seeking mode and the cats in T_g act in tracing mode, generating the two new groups S_{g+1} and T_{g+1}, respectively (lines 4–5). Then, the two new groups are joined to generate a new Q_{g+1} by assuming the k-best solutions in the union (line 6).

Algorithm 1 Binary cat swarm optimization algorithm.

```
1: $Q_g \leftarrow$ initializePopulation($k$)
2: while not stop condition do
3:     $S_g, T_g \leftarrow$ selectGroupCats($Q_g, mr$)
4:     $S_{g+1} \leftarrow$ applySeekingMode($S_g$)
5:     $T_{g+1} \leftarrow$ applyTracingMode($T_g$)
6:     $Q_{g+1} \leftarrow$ selectBestSolutions($S_{g+1}, T_{g+1}, k$)
7: end while
```

In the next subsections, we discuss the general behavior of both modes and an improving operator for repairing and simplifying solutions.

4.1. Seeking mode

Some parameters affect this mode:

- *cdc*: counts of dimensions to change. It indicates the percentage of columns selected for a possible mutation.

- *pmo*: the probability of mutation operation. It is the chance that a given column is mutated.
- *smp*: the size of the seeking memory pool. It defines the size of the pool considered for each cat, *i.e.*, the number of identical copies generated from a given solution.

Algorithm 2 shows the general scheme of this mode. Below, we describe the algorithm step by step:

Algorithm 2 Seeking mode.

```
1: for each $\eta \in S_g$ do
2:     $PIC \leftarrow$ generatePool($\eta, smp$)                      ▷ step 1
3:     $PIC \leftarrow$ mutatePool($PIC, cdc, pmo$)                 ▷ step 2
4:     $PIC \leftarrow$ evaluateRepairSolutions($PIC$)              ▷ step 3
5:     $\eta \leftarrow$ rouletteWheelSelection($PIC$)              ▷ step 4
6: end for
```

- **Step 1** (line 2): a pool of *smp* identical copies to a given solution η is generated. This pool is denoted as *PIC*.
- **Step 2** (line 3): according to *cdc*, several columns of the solutions in *PIC* are selected for a possible mutation. Next, such columns are mutated according to *pmo* by changing zero by one and vice versa.
- **Step 3** (line 4): the mutated solutions in the previous step are evaluated. If a solution is infeasible, it is repaired through the procedure described in Section 4.3.
- **Step 4** (line 5): η is replaced by a solution selected from *PIC* through a roulette wheel selection. Thus, each $\zeta \in PIC$ is assigned a selection probability sp_ζ given by

$$sp_\zeta = \left| \frac{f(\zeta) - f(worst_{PIC})}{f(worst_{PIC}) - f(best_{PIC})} \right|,$$

where $f(\cdot)$ is the fitness value of a solution calculated as given by Eq. (2), $best_{PIC}$ and $worst_{PIC}$ are the best and the worst solution in *PIC*, respectively, and $|\cdot|$ provides the absolute value of a number.

4.2. Tracing mode

Some parameters affect this mode:

- $iw \in [0, 5]$: inertia weight.
- $ac \in [0, 5]$: acceleration constant.
- $vb \in [0, 1]$: velocity bound.

Algorithm 3 shows the general scheme of this mode. Below, we describe the algorithm step by step:

Algorithm 3 Tracing mode.

```
1: for each $\eta \in T_g$ do
2:     for $j = 1, 2, \ldots, n$ do
3:         $v_\eta^j \leftarrow$ calculateVelocity($\eta, j, iw, ac$)             ▷ step 1
4:         $v_\eta^j \leftarrow$ analyzeVelocityBound($v_\eta^j, vb$)            ▷ step 2
5:         $pmut_\eta^j \leftarrow$ calculateMutationProbability($v_\eta^j$)   ▷ step 3
6:         $\eta^j \leftarrow$mutateColumn($pmut_\eta^j, \eta, j$)               ▷ step 4
7:     end for
8:     $\eta \leftarrow$ evaluateRepairSolution($\eta$)                      ▷ step 5
9: end for
```

- **Step 1** (line 3): we calculate the velocity of the cat $\eta \in T_g$ as a probability of change. This value is calculated for each column of η. Thus, the velocity v_η^j of the *j*th column of η is given by

$$v_\eta^j = \begin{cases} w_1^j & \text{if } \eta_j = 0 \\ w_0^j & \text{otherwise} \end{cases}, \tag{9}$$

where w_1^j and w_0^j are the probabilities that η_j changes to one and zero, respectively. The values of w_1^j and w_0^j are given by

$$w_1^j = w_1^{j\prime} iw + d_1^j,$$

$$w_0^j = w_0^{j\prime} iw + d_0^j,$$

where $w_1^{j\prime}$ and $w_0^{j\prime}$ are the probabilities that η_j changes to one and zero in the previous iteration of the algorithm, respectively. The values of d_1^j and d_0^j are expressed as

$$d_1^j = \begin{cases} \alpha\ ac & \text{if } best_j = 1 \\ -\alpha\ ac & \text{otherwise} \end{cases},$$

$$d_0^j = \begin{cases} -\alpha\ ac & \text{if } best_j = 1 \\ \alpha\ ac & \text{otherwise} \end{cases},$$

where α is a uniform random number in the interval [0, 1] and $best_j$ is the value of the *j*-column of the best solution in Q_g. Note that $w_1^{j\prime}$ and $w_0^{j\prime}$ are initialized to zero at the beginning of the algorithm.
- **Step 2** (line 4): we check if v_η^j is within the bound determined by *vb*. If the velocity is greater than this limit, it is replaced by the value of *vb*.
- **Step 3** (line 5): we dynamically calculate the mutation probability $pmut_\eta^j$ of each column of η based on the velocity obtained in the previous step. To this end, we consider the transfer functions introduced in the next section.
- **Step 4** (line 6): we mutate the column η^j based on the mutation probability obtained in the previous step. To this end, we consider the discretization methods introduced in the next section.
- **Step 5** (line 8): the new solution η is evaluated and improved based on the procedure described in Section 4.3.

4.3. Improving operator

Based on the SCP definition discussed in Section 3, it is possible that a solution does not satisfy the constraints, resulting in an infeasible solution. In this section, we describe an improving operator for transforming infeasible solutions into feasible ones and removing redundant columns to reduce the solution cost (Beasley & Chu, 1996). Note that a column is redundant if after removing it, the solution remains feasible.

Algorithm 4 shows the general scheme of this operator. Below, we describe the algorithm step by step, considering a notation inspired by the formulation in Section 3. In steps 2 and 3, we describe the repairing function. In steps 4 and 5, we describe the removing redundancy function.

- **Step 1** (line 1): we identify the set of non-covered rows $U \subseteq I$ in the current solution $S \subseteq J$. To this end, we first calculate the number of columns w_i in *S* covering a given row $i \in I$, which is given by

$$w_i = ||S \cap J_i||,$$

where $||\cdot||$ is the cardinal of a set and $J_i \subseteq J$ is the column set covering the row *i*. Then, *U* is calculated as

$$U = \{i \in I : w_i = 0\} \quad .$$

Algorithm 4 Improving operator.

```
1:  $w_i, U \leftarrow$ initializeVariables($S$)                    ▷ Step 1
2:  for each $i \in U$ do                                     ▷ Repairing function
3:      $j \leftarrow$ selectColumnCoveringRow($i$)             ▷ Step 2
4:      $S, w_i, U \leftarrow$ updateVariables(j)               ▷ Step 3
5:  end for
6:  for each $j \in S$ do                                     ▷ Removing redundancy function
7:      if $w_i > 1, \forall i \in I_j$ then                  ▷ Step 4
8:          $S, w_i \leftarrow$ removeColumn(j)                ▷ Step 5
9:      end if
10: end for
```

- **Step 2** (line 3): we select a column $j \in J_i$ covering a given row $i \in U$, which minimizes the function given by
$$\frac{c_j}{||U \cap I_j||},$$
where $I_j \subseteq I$ is the row set covered by the column j.
- **Step 3** (line 4): once the column j is selected, we include this column in the solution S, that is
$$S \leftarrow S \cup \{j\} \quad .$$
As such column covers the row $i \in U$, we update the number of columns in S covering the row i, that is
$$w_i \leftarrow w_i + 1 \quad .$$
Next, U is updated by removing the row set covered by the column j, that is
$$U \leftarrow U - I_j \quad .$$
- **Step 4** (line 7): we study if a given column j is redundant. To this end, we check if the number of columns in S covering the row i is greater than one, $\forall i \in I_j$. If the condition holds, then we remove the column j by going to step 5. Otherwise, the column is not removable.
- **Step 5** (line 8): the column j is removed from the solution by following the expression given by
$$S \leftarrow S - j \quad .$$
Next, we update the number of columns covering the rows in I_j, that is
$$w_i \leftarrow w_i - 1, \quad \forall i \in I_j \quad .$$

5. Transfer and discretization functions

In this section, we discuss the transfer and discretization functions considered for addressing the steps 3 and 4 of the tracing mode described in Section 4.2.

5.1. Transfer functions

Transfer functions give the probability $pmut_\eta^j$ of mutating the jth column of the solution $\eta \in T_g$ according to the current velocity v_η^j of η_j as given by Eq. (9), with $j \in 1, 2, \ldots, n$.

We consider the eight transfer functions formulated in Table 2 and illustrated graphically in Fig. 2, Kennedy and Eberhart (1997) and Mirjalili, Mohd, Taherzadeh, Mirjalili, and Salehi (2011) proposed S_2 and V_2, respectively; and Mirjalili and Hashim (2012) proposed the others. The functions are divided into two groups, called s-shape (S_1, S_2, S_3, and S_4) and v-shape (V_1, V_2, V_3, and V_4), based on the function shapes. Thus, s-shape functions are s-shaped and v-shape functions are v-shaped as shown in Fig. 2(a) and (b), respectively. Moreover, s-shape functions are odd and v-shape functions are even. Mirjalili and Hashim (2012) compared both groups, concluding that v-shape functions outperformed s-shape ones for a standard general framework. However, they did not perform any formal study of the differences observed.

Table 2
Formulation of the transfer functions.

s-shape	v-shape
$\mathbf{S_1}: pmut_\eta^j = \frac{1}{1+e^{-2v_\eta^j}}.$	$\mathbf{V_1}: pmut_\eta^j = \left\lvert \mathrm{erf}\left(\frac{\sqrt{\pi}}{2} v_\eta^j\right)\right\rvert.$
$\mathbf{S_2}: pmut_\eta^j = \frac{1}{1+e^{-v_\eta^j}}.$	$\mathbf{V_2}: pmut_\eta^j = \lvert \tanh(v_\eta^j)\rvert.$
$\mathbf{S_3}: pmut_\eta^j = \frac{1}{1+e^{\frac{-v_\eta^j}{2}}}.$	$\mathbf{V_3}: pmut_\eta^j = \left\lvert \frac{v_\eta^j}{\sqrt{1+(v_\eta^j)^2}}\right\rvert.$
$\mathbf{S_4}: pmut_\eta^j = \frac{1}{1+e^{\frac{-v_\eta^j}{3}}}.$	$\mathbf{V_4}: pmut_\eta^j = \left\lvert \frac{2}{\pi}\arctan\left(\frac{\pi}{2} v_\eta^j\right)\right\rvert.$

Analyzing Fig. 2, we note that all the functions in a group are similar but having a different degree of smoothness. Thus, functions with less smoothness have a smaller range of input values v_η^j providing non-extreme values of the output interval [0,1] than smoother functions.

For example, S_1 is less smooth than S_2. Based on S_1 formulation, if we consider v_η^j equaling 2.3 and −2.3, we get a $pmut_\eta^j$ value of 0.99 and 0.01, respectively. On the other hand and based on S_2 formulation, the same $pmut_\eta^j$ values are obtained by assuming v_η^j equaling 4.6 and −4.6, respectively. Thus, the range of input values of S_2 providing non-extreme values is greater than the range of S_1.

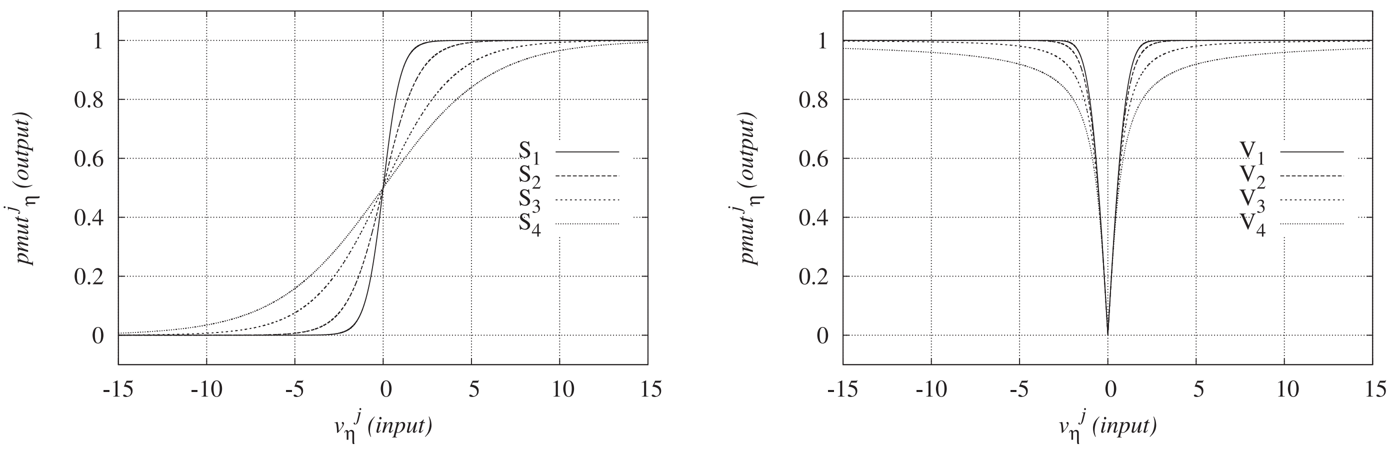
The motivation of including both groups of transfer functions is as follows. A high velocity implies that the cat is away from the optimal solution and a low velocity implies that the cat is close to the optimal solution. Based on this velocity, the strategy of both groups of transfer functions is different. S-shape functions cause that cats with a low velocity have associated a low mutation probability, while cats with a high velocity have associated a high mutation probability (see Fig. 2a). On the contrary, v-shape functions cause that the mutation probability is high for cats with low and high velocities (see Fig. 2b). Thus, we could define the following indicative order from more conservative to more aggressive exploration strategy: S_4, S_3, S_2, S_1, V_4, V_3, V_2, and V_1.

5.2. Discretization functions

Discretization functions generate a new binary value according to the mutation probability given by transfer functions. We consider the five discretization functions reviewed by Crawford et al. (2015c). Such functions are in Table 3, where $\bar{\cdot}$ is the logical complement of a proposition, $pmut_\eta^j$ is the output of the transfer function, α is a uniform random number in the interval [0, 1], η_j is given by Eq. (8), $\eta_j{}'$ is the expected output of the discretization function, $f(\cdot)$ is the fitness value of a solution calculated as given by Eq. (2), and $best_j$ is the value of the jth column of the best solution in Q_g.

Next, we briefly describe each function:

- Elitist roulette (D_1): it considers a probability-based selection process, where $P\left[\eta_j{}' = \delta_j\right]$ denotes the probability that $\eta_j{}'$ assumes the value of the jth column of $\delta \in Q_g$. Thus, if the condition holds, a value is randomly selected based on the rule: the better the solution, it is more likely to be selected. Otherwise, a value of 0 is returned.
- Complement (D_2): if the condition holds, it returns the complement of the input value η_j. Otherwise, it returns η_j.
- Static probability (D_3): if the first condition holds, it returns a value of 0. If the second condition holds, it returns the value of



(a) S-shape functions. (b) V-shape functions.

Fig. 2. S-shape and v-shape transfer functions.

Table 3
Formulation of the discretization functions.

D_1: (elitist roulette) $\begin{cases} P\left[\eta_j{}' = \zeta_j\right] = \dfrac{f(\zeta)}{\sum_{\delta \in Q_g} f(\delta)} & \text{if } \alpha \le pmut^j_\eta, \\ P\left[\eta^{j\prime} = 0\right] = 1 & \text{otherwise.} \end{cases}$
D_2: (complement) $\eta_j{}' = \begin{cases} \overline{\eta_j} & \text{if } \alpha \le pmut^j_\eta, \\ \eta_j & \text{otherwise.} \end{cases}$
D_3: (static probability) $\eta_j{}' = \begin{cases} 0 & \text{if } pmut^j_\eta \le \alpha, \\ best_j & \text{if } \alpha \le pmut^j_\eta \le \frac{1}{2}(1+\alpha), \\ 1 & \text{otherwise.} \end{cases}$
D_4: (set the best) $\eta_j{}' = \begin{cases} best_j & \text{if } \alpha \le pmut^j_\eta, \\ \eta_j & \text{otherwise.} \end{cases}$
D_5: (standard) $\eta_j{}' = \begin{cases} 1 & \text{if } \alpha \le pmut^j_\eta, \\ 0 & \text{otherwise.} \end{cases}$

the jth column of the best solution in Q_g. Otherwise, it returns a value of 1.

- Set the best (D_4): if the condition holds, it returns the value of the jth column of the best solution in Q_g. Otherwise, it returns the input value η_j.
- Standard (D_5): if the condition holds, it returns a value of 1. Otherwise, it returns a value of 0.

The motivation of including these five functions is because they all offer different capabilities to the search strategy. As a way of identifying such capabilities, we could define the following indicative order from more exploratory to more exploitative strategies: D_2, D_5, D_1, D_3, and D_4. Crawford et al. (2015c) checked that the behavior of such discretization functions was different. However, they did not perform any formal study of the differences observed.

6. Experimentation

In this section, we discuss both the experimental method followed and the experimental results obtained.

6.1. Experimental methodology

We apply the BCSO algorithm for solving two different problem sets:

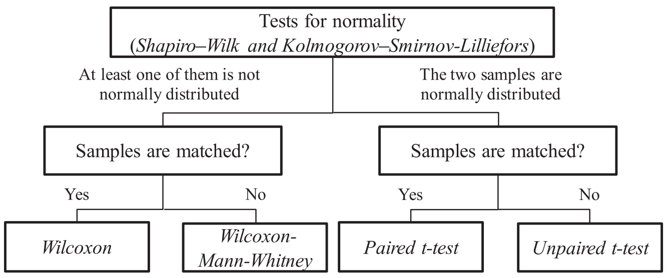

Tests for normality
(Shapiro–Wilk and Kolmogorov–Smirnov-Lilliefors)
At least one of them is not normally distributed
The two samples are normally distributed
Samples are matched?
Samples are matched?
Yes
No
Yes
No
Wilcoxon
Wilcoxon-Mann-Whitney
Paired t-test
Unpaired t-test

Fig. 3. Statistical methodology.

- The standard OR-library benchmark described in Table 4 includes 65 randomly generated non-unicost instances, which are available in Beasley (2016). This dataset is widely considered in the literature to test complex algorithms as the SIA assumed in this work, *e.g.*, Ren et al. (2010), Lu and Vasko (2015); Naji-Azimi et al. (2010); Sundar and Singh (2012b), and Vasko et al. (2016).
- Table 5 shows various combinatorial optimization problems modeled as unicost SCPs. In general, unicost problems are more challenging compared to their non-unicost approaches. The dataset is available in Beasley (2016).

In Tables 4 and 5, *Density* field means the percentage of ones in the *A* matrix given by Eq. (1) and *Optimal solution* field shows two possible values, known and unknown, according if the instances have associated a solution checked to be optimal, or instead it could not be tested due to the instance complexity. Thus, instance sets 4, 5, 6, A, B, C, D, NRE, and NRF have associated a solution checked to be optimal. On the contrary, we only know the best historical solution for sets NRG, NRH, and all the unicost instances.

The authors of BCSO adapted the SIA to the discrete scope assuming the transfer function S_2 and the discretization function D_3. Now, we solve the two problem sets by considering the forty binarization techniques introduced before to study if there are significant differences between the binarization approaches. To this end, we perform 30 independent runs for each instance and binarization technique, being 30 a widely accepted value for getting statistical conclusions (Hays & Winkler, 1970). Next, we analyze the distributions of 30 samples for each instance and binarization technique through the methodology shown in Fig. 3 (Hays & Winkler, 1970).

Table 4
Standard OR-library benchmark description.

Instance set	Number of instances	m	n	Cost range	Density (%)	Optimal solution
4	10	200	1000	[1,100]	2.00	known
5	10	200	2000	[1,100]	2.00	known
6	5	200	1000	[1,100]	5.00	known
A	5	300	3000	[1,100]	2.00	known
B	5	300	3000	[1,100]	5.00	known
C	5	400	4000	[1,100]	2.00	known
D	5	400	4000	[1,100]	5.00	known
NRE	5	500	5000	[1,100]	10.00	known
NRF	5	500	5000	[1,100]	20.00	known
NRG	5	1000	10,000	[1,100]	2.00	unknown
NRH	5	1000	10,000	[1,100]	5.00	unknown

Table 5
Unicost benchmark description.

Instance name	m	n	Density (%)	Optimal solution
CYC.6	240	192	2.10	unknown
CYC.7	672	448	0.90	unknown
CYC.8	1792	1024	0.40	unknown
CLR.10-4	511	210	12.30	unknown
CLR.11-4	1023	330	12.40	unknown
CLR.12-4	2047	495	12.50	unknown
CLR.13-4	4095	715	12.50	unknown

The type of stop condition is the same for all the instances, being based on the number of evaluations. This type of criterion is fairer than others, such as elapsed time, which depends on the machine performance. We consider two different values for the stop condition according to the problem size: 40000 evaluations for the sets 4, 5, 6, A, B, C, D and 5000 evaluations for the sets NRE, NRF, NRG, NRH, and all the unicost instances. We assume two different values because the computation effort required for both groups is notably different. As proof of this and assuming the machine described in Section 7, the average execution time of one run for the first group is 29.46 seconds, while the average execution time of one run for the second group is 221.23 seconds. We could consider the same value of 40,000 evaluations for both groups. However, the computational time for the second group would be excessively high. On the contrary, if we consider the same value of 5000 evaluations for both groups, we would get a reduced computational time for the first group. However, the reliability of the results obtained could be also reduced, while considering 40,000 evaluations involves a low computational effort.

As any other metaheuristic, we should configure the BCSO algorithm before running the experiments. To this end and for each parameter of the algorithm, we define a range of values to study and a default configuration. Then, we perform 30 independent runs for each configuration of a parameter, instance, and binarization technique, resulting in 1200 runs for each value of a parameter and instance. Then, we select the configuration, providing the best performance on average. Next, we select another parameter until all of them are set. Table 6 shows for each parameter the range of values and the configurations selected. Note that *iw, ac*, and *vb* parameters were experimentally set to 1.00, 1.00, and 0.80 to decrease the parametric swap complexity, *i.e.*, the number of parameters to configure.

Table 6 includes the configurations obtained for each benchmark. Note that with the purpose of optimizing the algorithm behavior solving the SCP, we give a specific configuration based on the problem size, instead of getting a general configuration for the two problems sets. As expected, this parametric swap needs a large amount of computation effort. Thus, for the OR-library, we group the instances by a similar number of rows x columns as follows: The first group includes instance sets 4, 5, and 6 with size 200 $\times$ 1000 and 200 $\times$ 2000. The second group includes instance sets A and B with size 300 $\times$ 3000. The third group includes instance sets C and D with size 400 $\times$ 4000. The fourth group includes instance sets NRE and NRF with size 500 $\times$ 5000. The fifth group includes instance sets NRG and NRH with size 1000 $\times$ 10000. As a result, we get five configurations for solving the OR-library (65 instances). On the contrary, for the unicost benchmark, we give a configuration for each instance because the number of rows x columns is significantly different and then we opted not to group the instances.

6.2. Experimental results

As a quality metric, we consider the average Relative Percentage Deviation (RPD), quantifying how close is a solution to the optimal one. The average RPD metric $\overline{rpd}^{i}_{d,t}$ for a discretization function $d \in D_1, D_2, \ldots, D_5$, a transfer function $t \in S_1, S_2, \ldots, S_4, V_1, V_2, \ldots, V_4$, and an instance $i \in$ {OR-library, unicost-library} is calculated as

$$\overline{rpd}^{i}_{d,t} = \left(\frac{\overline{z}^{i}_{d,t} - z^{i}_{opt}}{z^{i}_{opt}} \right) 100,$$

where z^{i}_{opt} is the optimal solution or the best historical solution, depending on the case, for the instance i and $\overline{z}^{i}_{d,t}$ is the average fitness value obtained for 30 runs while solving the instance i through d and t functions.

Tables 7 and 8 show the average RPD for each instance and binarization technique, where lower (better) RPD values are shaded. Note that Table 7 includes s-shape transfer functions and Table 8 includes v-shape ones. In these tables, some binarization techniques seem to offer better performance than others for a given instance. However, we do not know if the differences observed are significant.

To this end, we first study if the data follow a normal distribution through Kolmogorov–Smirnov–Lilliefor's (Lilliefors, 1967) and Shapiro–Wilk's (Shapiro & Wilk (1965)) tests, assuming the hypothesis H_0: data follow a normal distribution and H_1: otherwise. We obtained p-values lower than 0.05 for all the cases. Hence, we cannot assume that the data follow a normal distribution. Thus, we should consider the median as the average value. Note that Tables 7 and 8 were generated after performing this study and then the median was assumed.

Next, we study if there are significant differences among the binarization techniques. As samples are independent and data do not follow a normal distribution, we assume the Wilcoxon–Mann–Whitney's Mann and Whitney (1947) test with the hypothesis H_0: $\overline{rpd}^{i}_{a,b} \geq \overline{rpd}^{i}_{c,d}$, $\forall a, c \in D_1, D_2, \ldots, D_5$ and $\forall b, d \in S_1, S_2, \ldots, S_4, V_1, V_2, \ldots, V_4$.

Table 6
Parametric swap.

Parameter	Range	OR-library benchmark		Unicost benchmark			
		Instance set	Selected	Instance name	Selected	Instance name	Selected
		4, 5, and 6	100	CYC.6	50	CLR.10-4	75
		A and B	50	CYC.7	50	CLR.10-5	50
k	[10,15,... ,200]	C and D	30	CYC.8	25	CLR.10-6	75
		NRE and NRF	25			CLR.10-7	25
		NRG and NRH	20				
		4, 5, and 6	0.70	CYC.6	0.75	CLR.10-4	0.75
		A and B	0.65	CYC.7	0.75	CLR.10-5	0.75
mr	[0.10,0.15,..,0.90]	C and D	0.50	CYC.8	0.25	CLR.10-6	0.25
		NRE and NRF	0.50			CLR.10-7	0.75
		NRG and NRH	0.50				
		4, 5, and 6	5	CYC.6	10	CLR.10-4	25
		A and B	5	CYC.7	10	CLR.10-5	10
smp	[5,10,... ,50]	C and D	10	CYC.8	10	CLR.10-6	25
		NRE and NRF	15			CLR.10-7	10
		NRG and NRH	20				
		4, 5, and 6	0.97	CYC.6	0.80	CLR.10-4	0.80
		A and B	0.93	CYC.7	0.80	CLR.10-5	0.80
pmo	[0.01,0.02,... ,1.00]	C and D	0.90	CYC.8	0.80	CLR.10-6	0.90
		NRE and NRF	1.00			CLR.10-7	0.90
		NRG and NRH	1.00				
		4, 5, and 6	0.10	CYC.6	2.00	CLR.10-4	2.00
		A and B	0.10	CYC.7	0.60	CLR.10-5	1.00
cdc(%)	[0.10,0.20,..,90.00]	C and D	0.20	CYC.8	0.10	CLR.10-6	2.00
		NRE and NRF	0.20			CLR.10-7	0.60
		NRG and NRH	1.00				

We analyze the p-values obtained by considering a significance level of 0.05. Based on this analysis and for each instance set, Table 9 shows the percentage of cases in which a binarization technique offers the best significant performance compared to all others. In this table, better values are shaded from a darker to a lighter tone, *i.e.*, from better to worse behavior.

Analyzing this table, we reach that for the instance set 4 the best combination is (D_1, V_3). For the set 5 are (D_3, S_4), (D_3, V_3), and (D_2, V_4). For the set 6 is (D_3, V_4). For the set A is (D_2, V_3). For the set B is (D_3, S_2). For the set C is (D_4, V_4). For the set D is (D_1, S_2). For the set NRE is (D_4, V_4). For the set NRF is also (D_4, V_4). For the set NRG is (D_1, S_3). For the set NRH is also (D_1, S_3). For the set CYC is (D_1, V_1) and for the set CLR is (D_3, S_4).

From this first study, we do not reach any specific trend for selecting a concrete binarization approach. To this end, Tables 10 and 11 show the Relative Variation Percentage (RVP) for each family of instance sets and binarization approach, where we group the instances by a similar complexity level forming families. Thus, we define the following family order from lower to higher complexity: $F_1 = \{4, 5, 6\}$, $F_2 = \{A, B\}$, $F_3 = \{C, D\}$, $F_4 = \{NRE, NRF\}$, $F_5 = \{CYC, CLR\}$, and $F_6 = \{NRG, NRH\}$.

The RVP metric $rvp^f_{d,t}$ for a discretization function d, a transfer function t, and a family of instance sets $f \in F_1, F_2, \ldots, F_6$ is expressed as

$$rvp^f_{d,t} = \frac{|apc^f_{best} - apc^f_{d,t}|}{apc^i_{best}},$$

where $apc^f_{d,t}$ is the average percentage of cases, compared to all others, in which the combination of d and t functions offers the best significant performance solving the family f, and apc^f_{best} is the average percentage of cases, compared to all others, in which the best binarization approach solves the family f. Formally, $apc^f_{d,t}$ is given by

$$apc^f_{d,t} = \frac{\sum\limits_{j \in F_f} pc^j_{d,t}}{||F_f||},$$

where $pc^j_{d,t}$ is the percentage of cases, compared to all others, in which the combination of d and t functions offers the best significant performance solving the instance set j. Note that $pc^j_{d,t}$ values are provided in Table 9 as a result of the statistical study performed before. Consequently, apc^f_{best} is given by

$$apc^f_{best} = \arg \max_{\substack{d \in D_1, D_2, \ldots, D_5 \\ t \in S_1, S_2, \ldots, S_4, V_1, V_2, \ldots, V_4}} \{apc^f_{d,t}\},$$

where argmax{·} provides the point/points where a function gets its maximum value/values.

Tables 10 and 11 have the same RVP values, but showing the information differently to analyze the behavior of transfer and discretization functions in a better way. Thus, Table 10 presents the information grouped by transfer functions and Table 11 presents the information grouped by discretization functions. Note that the lower the RVP value, the binarization approach is better. In this line, better RVP values are shaded from a darker to a lighter tone in both tables.

Table 10 is sorted based on the indicative order defined for transfer functions in Section 5.1, *i.e.*, from more conservative to more aggressive exploration strategy. Analyzing this table, we find the following trend: v-shape transfer functions outperforms s-shape ones for solving small and medium problems (I_1, I_2, I_3, and I_4 families), while s-shape functions are better for solving large problems (I_5 and I_6 families). This fact agrees with the behavior of the transfer functions described before in Section 5.1. V-shape functions follow an aggressive exploration strategy, assigning high mutation probabilities for both near and far optimal solutions. On the contrary, s-shape functions follow a conservative exploration strategy, assigning high mutation probabilities only for far optimal solutions. Thus, v-shape functions are fit for solving limited search space problems and s-shape functions are fit for solving large search space problems. The same behavior is shown in Table 12, where RVP values are presented for each family regardless of the discretization function considered. To this end, the

Table 7
Average RPD for each instance and binarization technique: s-shape transfer functions.

Instance	Discretization and s-shape transfer functions																			
	D_1	D_2	D_3	D_4	D_5	D_1	D_2	D_3	D_4	D_5	D_1	D_2	D_3	D_4	D_5	D_1	D_2	D_3	D_4	D_5
	S_1					S_2					S_3					S_4				
4.1	3.58	2.80	2.84	3.80	3.24	2.58	6.11	6.44	2.74	3.16	2.83	6.34	6.85	3.01	3.11	3.08	5.93	3.92	3.70	3.13
4.2	5.11	4.11	4.77	3.89	4.71	4.40	8.44	6.22	4.62	3.81	4.40	8.83	6.00	5.16	5.05	4.72	8.14	5.03	3.71	4.05
4.3	7.95	7.31	7.18	7.54	8.09	7.98	11.92	7.93	7.71	7.96	7.90	11.52	7.95	7.41	8.28	7.49	10.72	7.86	8.51	7.89
4.4	4.15	4.45	5.26	5.16	3.62	3.29	5.92	3.90	4.28	3.85	4.44	7.32	3.73	3.61	4.22	4.68	6.71	3.77	3.91	3.54
4.5	2.77	2.78	2.44	2.46	2.55	2.51	4.28	2.75	2.55	2.19	2.81	4.36	2.53	2.60	2.40	2.29	4.19	2.57	2.42	2.34
4.6	1.73	2.33	1.24	1.55	1.61	1.74	4.37	1.24	1.32	1.42	1.50	3.92	1.66	1.42	1.68	1.57	4.92	1.43	1.30	1.13
4.7	2.09	2.05	2.36	2.29	2.19	2.57	3.97	2.18	2.24	2.26	2.09	4.00	2.13	2.15	2.52	2.33	4.29	2.07	2.22	1.98
4.8	5.65	4.07	5.09	4.95	4.63	4.75	10.00	4.98	5.22	4.30	4.78	8.64	3.92	4.97	5.09	4.80	9.48	5.63	5.73	5.37
4.9	5.82	5.82	5.54	5.40	5.83	5.90	8.32	6.04	5.60	5.92	5.28	8.13	5.96	5.68	5.46	5.63	6.97	5.37	5.43	5.55
4.10	2.52	2.72	2.79	2.15	2.61	2.85	4.99	2.63	2.57	3.17	2.51	4.51	2.27	2.24	2.83	3.05	5.40	2.72	2.53	2.47
5.1	3.62	4.41	3.97	3.79	4.16	4.44	6.09	3.77	3.60	3.78	4.22	6.23	4.02	3.54	4.49	4.08	5.72	3.53	3.65	3.75
5.2	4.48	4.25	4.62	4.58	4.12	4.45	6.92	5.11	4.23	4.02	4.18	6.56	4.07	4.30	3.77	4.42	6.58	4.03	4.24	4.67
5.3	3.72	4.31	3.88	3.83	3.39	3.94	6.28	3.58	2.98	3.01	3.42	6.14	3.33	3.08	3.39	3.67	6.08	3.61	3.94	3.86
5.4	1.58	1.60	1.46	1.50	1.47	1.45	1.96	1.49	1.54	1.46	1.47	1.97	1.46	1.49	1.56	1.52	1.87	1.38	1.51	1.53
5.5	4.49	4.09	4.00	4.19	4.33	4.25	5.61	4.31	4.27	4.44	4.04	5.37	4.27	4.23	4.28	4.66	5.61	4.28	4.31	4.27
5.6	6.24	5.99	5.70	6.65	5.99	5.46	10.53	6.12	6.12	5.20	6.29	9.61	6.32	6.04	6.08	5.54	9.56	6.40	5.07	5.18
5.7	4.43	4.82	4.45	4.23	4.37	4.55	9.31	4.60	4.45	4.46	3.61	9.62	3.79	4.53	4.23	4.82	9.40	4.78	3.98	5.01
5.8	6.72	6.30	6.69	6.62	6.82	7.06	10.31	6.42	6.82	6.85	7.03	11.04	6.63	6.55	6.69	6.22	10.57	6.40	7.30	6.82
5.9	0.61	1.18	1.46	0.56	0.62	0.58	1.58	1.49	0.62	0.51	0.72	1.95	0.72	1.18	1.18	0.58	1.82	0.61	0.62	1.15
5.10	4.05	4.11	3.92	4.00	4.36	3.90	5.60	3.92	4.23	4.30	4.05	5.58	4.29	4.14	3.95	3.87	5.26	3.70	4.23	4.09
6.1	7.10	7.08	7.00	7.15	7.00	6.67	12.44	6.57	6.84	7.29	6.45	11.74	7.78	7.51	6.62	7.08	12.63	7.46	7.32	6.57
6.2	3.07	2.95	2.94	2.93	2.97	2.74	5.21	2.74	2.74	2.74	2.74	5.27	2.74	2.99	3.02	2.74	5.32	2.74	2.74	2.95
6.3	4.94	5.31	5.21	5.10	5.06	5.22	6.28	5.15	5.10	4.90	5.26	6.90	5.13	5.54	5.13	4.74	6.25	5.82	4.87	5.86
6.4	2.65	2.93	2.80	3.00	2.93	2.90	3.64	2.65	3.03	2.93	2.88	3.92	2.95	2.98	3.23	3.08	3.94	3.00	2.93	3.18
6.5	4.84	5.24	5.01	4.64	5.07	5.09	6.21	4.87	5.16	5.65	5.13	6.31	5.57	5.18	4.93	4.80	6.25	5.20	4.89	4.70
A.1	8.87	8.85	8.85	9.00	9.14	9.16	10.84	9.16	9.18	9.45	8.74	10.67	8.75	9.33	9.16	9.05	10.75	9.10	8.88	9.28
A.2	5.03	5.16	5.20	5.20	5.09	5.38	6.20	5.16	5.17	5.21	4.87	6.19	5.34	5.20	5.05	5.05	6.18	5.13	5.12	5.24
A.3	5.13	5.47	4.86	4.68	5.47	5.17	8.72	5.19	5.01	5.01	5.46	7.82	4.86	4.84	5.26	5.53	7.57	4.70	4.77	5.29
A.4	4.91	4.83	5.26	4.96	5.37	4.99	5.90	5.07	4.66	4.91	5.16	6.04	5.24	5.24	5.09	4.93	6.41	5.41	4.91	5.07
A.5	1.27	1.26	1.12	1.13	1.17	1.26	1.57	1.27	1.27	1.12	1.21	1.36	1.30	1.27	1.23	1.36	1.41	1.16	1.19	1.26
B.1	8.41	8.84	7.58	8.21	6.81	7.78	10.34	8.79	7.83	8.70	8.45	11.64	7.83	7.90	8.45	7.63	11.26	8.36	8.02	8.16
B.2	11.49	11.49	12.02	11.71	11.93	12.46	14.74	10.26	12.11	11.58	11.67	15.04	12.37	11.71	11.40	11.40	14.21	12.50	11.45	11.45
B.3	3.54	3.29	3.46	4.00	3.79	3.75	5.29	3.50	3.54	3.83	3.25	6.46	3.54	3.42	2.83	4.04	6.00	3.87	3.67	2.92
B.4	6.33	6.33	6.33	6.33	6.33	6.33	7.17	6.33	6.29	6.33	6.33	7.26	6.33	6.33	6.33	6.33	7.05	6.33	6.33	6.54
B.5	1.39	1.39	1.39	1.39	1.39	1.39	1.39	1.39	1.39	1.39	1.39	1.39	1.39	1.39	1.39	1.39	1.39	1.39	1.39	1.39
C.1	3.38	3.25	3.32	3.33	3.30	3.38	4.41	3.39	3.26	3.36	3.32	4.82	3.33	3.36	3.42	3.27	4.76	3.32	3.45	3.38
C.2	5.13	5.10	5.05	5.04	5.33	5.18	6.93	5.43	5.31	4.84	5.08	6.83	5.07	5.16	5.25	4.63	7.05	5.08	5.07	4.87
C.3	9.16	8.67	9.20	8.90	9.11	9.27	12.47	9.42	9.40	9.81	9.01	12.17	9.47	9.00	8.94	8.79	12.72	9.57	9.49	9.55
C.4	8.87	8.81	8.54	9.01	9.19	9.24	11.17	8.86	9.39	8.86	11.71	11.05	9.33	9.27	8.71	9.21	11.29	9.06	9.68	8.93
C.5	6.96	7.18	7.18	7.12	7.12	6.79	9.22	6.47	6.88	7.97	9.57	9.49	7.49	7.16	6.88	7.09	8.56	6.81	7.07	7.32
D.1	8.33	8.64	7.94	7.67	8.64	9.01	7.72	7.33	8.97	7.67	9.14	7.78	8.33	8.77	8.33	7.56	7.67	8.33	6.72	8.83
D.2	6.06	6.06	6.06	6.06	6.06	5.71	6.06	6.06	6.06	6.06	6.06	5.71	6.06	5.71	6.06	6.06	6.06	6.06	6.06	5.61
D.3	9.26	9.03	9.31	9.21	9.07	9.12	9.31	9.44	9.12	9.72	10.09	9.72	9.72	9.72	9.72	9.40	9.21	9.35	9.26	9.12
D.4	6.13	6.24	6.34	6.51	6.13	5.81	6.29	5.91	5.97	5.97	7.37	6.40	5.43	6.13	5.86	6.08	6.18	5.59	6.02	6.13
D.5	6.99	6.67	6.99	6.61	6.83	6.50	6.56	6.56	6.28	6.50	7.70	7.05	7.16	6.45	6.72	6.72	7.16	6.89	6.39	6.61
NRE.1	3.45	3.45	3.45	3.45	3.45	3.45	3.45	3.45	3.45	3.45	3.45	3.45	3.45	3.45	3.45	3.45	3.45	3.45	3.45	3.45
NRE.2	14.56	15.00	14.67	15.11	15.00	14.89	15.00	15.56	15.33	15.00	15.89	15.00	14.56	15.22	14.78	15.11	14.78	15.00	15.11	15.56
NRE.3	23.21	23.58	21.36	22.72	22.96	23.95	24.07	23.21	24.20	23.58	23.95	21.85	22.84	22.10	23.70	23.46	23.09	23.21	22.72	22.72
NRE.4	17.86	17.86	17.86	17.86	17.86	17.86	17.86	17.86	17.86	17.86	17.86	17.86	17.86	17.86	17.86	17.86	17.86	17.86	17.86	17.86
NRE.5	7.14	7.14	7.14	7.14	7.14	7.14	7.14	7.14	7.14	7.14	7.14	7.14	7.14	7.14	7.14	7.14	7.14	7.14	7.14	7.14
NRF.1	21.43	21.43	21.43	21.43	21.43	21.43	21.43	21.43	21.43	21.43	21.43	21.43	21.43	21.43	21.43	21.43	21.43	21.43	21.43	21.43
NRF.2	20.00	20.00	18.00	20.00	20.00	20.00	20.00	20.00	18.44	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00
NRF.3	21.43	21.43	21.43	21.43	21.43	21.43	21.43	21.43	21.43	21.43	21.43	21.43	21.43	21.43	21.43	21.43	21.43	21.43	21.43	21.43
NRF.4	23.81	23.33	21.43	24.76	24.76	25.00	24.52	25.00	21.43	24.29	23.81	22.86	22.86	24.05	24.52	24.52	24.05	22.62	24.05	23.10
NRF.5	23.08	23.08	23.08	23.08	23.08	23.08	23.08	23.08	23.08	23.08	23.08	23.08	23.08	23.08	23.08	23.08	23.08	23.08	23.08	23.08
NRG.1	9.72	10.36	10.28	10.27	10.42	9.85	10.02	10.30	10.32	10.52	9.79	10.00	10.30	10.45	10.27	10.32	10.51	10.35	10.34	10.31
NRG.2	8.81	8.61	8.66	8.81	8.66	8.77	8.14	8.79	8.61	8.74	8.07	8.16	8.81	8.79	8.61	8.79	8.48	8.64	8.87	8.64
NRG.3	9.94	10.04	9.98	9.84	9.86	9.96	9.64	9.92	9.90	9.90	9.64	9.64	9.94	10.06	10.00	10.00	9.94	9.92	9.90	9.94
NRG.4	9.15	9.25	9.13	9.13	9.09	9.01	8.97	9.15	9.27	9.11	8.89	9.15	9.52	9.01	9.07	9.13	9.21	9.17	9.52	9.11
NRG.5	10.08	9.88	9.92	10.02	9.84	9.94	9.92	9.80	9.76	10.06	8.93	10.00	9.70	9.88	9.96	9.92	10.00	10.06	9.66	9.76
NRH.1	15.19	14.87	14.92	15.19	15.19	14.97	14.50	15.19	14.97	15.87	12.70	14.81	15.08	14.81	14.71	15.34	15.19	14.71	14.81	15.34
NRH.2	6.35	6.35	6.35	6.35	6.35	6.35	6.35	6.35	6.35	6.35	6.35	6.35	6.35	6.35	6.35	6.35	6.35	6.35	6.35	6.35
NRH.3	16.95	16.95	16.95	16.95	16.95	16.95	16.95	16.95	16.95	16.95	16.95	16.95	16.95	16.95	16.95	16.95	16.95	16.95	16.95	16.95
NRH.4	15.29	15.75	15.80	15.40	15.69	15.29	15.52	15.52	15.52	15.06	15.52	15.75	15.98	15.52	15.46	15.81	15.52	15.63	15.69	15.11
NRH.5	10.91	10.91	10.91	10.91	10.91	10.91	10.91	10.91	10.91	10.91	10.91	10.91	10.91	10.91	10.91	10.91	10.91	10.91	10.91	10.91
CYC.6	20.00	20.00	18.33	20.00	20.00	20.00	21.67	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00	21.67	20.00	20.00
CYC.7	24.31	24.31	24.31	25.00	25.00	25.00	24.31	25.00	24.31	25.00	25.00	25.00	25.00	24.31	23.61	23.61	24.31	25.69	25.69	24.31
CYC.8	24.42	24.13	24.42	24.13	24.71	25.58	24.13	24.71	24.71	24.42	24.42	25.00	24.71	25.29	24.42	24.13	24.42	24.42	25.00	24.71
CLR.10-4	16.00	16.00	16.00	16.00	20.00	16.00	20.00	16.00	20.00	20.00	16.00	16.00	16.00	16.00	20.00	16.00	20.00	20.00	20.00	20.00
CLR.11-4	30.43	30.43	30.43	30.43	30.43	30.43	30.43	30.43	30.43	30.43	30.43	30.43	30.43	30.43	30.43	30.43	30.43	30.43	30.43	30.43
CLR.12-4	34.78	34.78	34.78	34.78	34.78	34.78	34.78	34.78	34.78	34.78	34.78	34.78	34.78	34.78	34.78	34.78	34.78	34.78	34.78	34.78
CLR.13-4	52.17	52.17	52.17	52.17	47.83	47.83	47.83	52.17	52.17	52.17	52.17	47.83	52.17	47.83	52.17	52.17	52.17	47.83	52.17	52.17

Table 8
Average RPD for each instance and binarization technique: v-shape transfer functions.

Instance	Discretization and v-shape transfer functions																			
	D_1	D_2	D_3	D_4	D_5	D_1	D_2	D_3	D_4	D_5	D_1	D_2	D_3	D_4	D_5	D_1	D_2	D_3	D_4	D_5
	V_1					V_2					V_3					V_4				
4.1	2.76	3.50	3.14	3.30	2.94	3.01	3.08	3.28	3.64	3.05	2.69	3.64	3.19	3.48	3.61	3.23	6.90	2.95	3.57	3.38
4.2	4.45	4.20	4.55	3.49	4.18	5.18	4.52	4.88	3.72	4.14	3.95	4.83	4.46	4.15	4.30	3.74	5.62	5.02	4.11	4.34
4.3	8.31	7.77	7.73	7.69	7.67	7.40	7.51	7.20	8.54	7.96	7.55	8.64	7.84	8.12	7.59	7.50	7.70	7.64	7.81	7.13
4.4	4.95	4.20	3.73	4.89	3.79	3.87	4.22	4.14	3.60	4.15	4.14	3.92	3.91	4.64	4.57	3.92	4.33	4.10	3.70	4.23
4.5	2.29	2.86	2.89	2.66	2.32	2.49	3.07	2.42	2.49	2.74	2.94	2.58	2.43	2.81	2.37	2.36	2.79	3.25	2.73	2.54
4.6	1.33	1.47	1.83	1.53	1.23	1.57	1.50	1.09	1.42	1.26	1.55	1.40	1.55	1.39	1.15	1.68	1.57	1.70	1.32	1.51
4.7	2.49	2.18	2.24	2.22	2.44	2.42	2.35	2.27	2.41	1.91	1.75	2.18	2.60	2.47	2.54	2.17	2.51	2.50	2.11	2.75
4.8	5.06	5.43	5.14	5.29	5.15	5.62	5.12	4.56	4.80	5.49	5.28	4.19	4.74	5.35	5.38	4.83	5.87	5.64	4.71	3.88
4.9	5.66	5.72	5.73	5.45	5.61	5.62	5.46	5.69	5.94	6.00	5.84	5.65	5.46	5.93	5.21	5.91	5.56	5.40	5.52	5.54
4.10	2.69	2.74	2.50	2.13	2.82	3.05	2.52	2.85	2.44	2.96	2.44	2.13	2.68	2.92	2.36	2.26	2.85	2.84	2.38	3.15
5.1	3.37	3.76	3.63	3.80	4.57	3.99	4.68	3.72	3.97	4.01	3.63	3.87	3.71	3.47	3.82	4.58	3.38	3.86	3.83	3.76
5.2	4.03	4.46	4.69	4.15	4.19	4.27	4.32	4.48	3.86	3.82	4.38	4.14	3.74	4.07	4.37	4.18	4.17	4.37	3.93	3.98
5.3	3.13	3.27	4.07	3.35	3.45	3.73	3.69	3.29	3.44	3.38	4.12	3.32	3.39	3.55	3.92	3.50	4.00	3.53	3.22	3.61
5.4	1.54	1.58	1.45	1.52	1.57	1.45	1.63	1.46	1.54	1.34	1.54	1.53	1.29	1.57	1.53	1.36	1.58	1.47	1.52	1.49
5.5	4.09	4.15	4.33	4.60	4.30	4.27	4.47	4.25	4.41	4.22	4.50	4.45	4.15	4.27	4.17	4.41	4.27	4.47	4.15	4.38
5.6	6.49	5.83	4.89	6.43	5.93	5.60	6.06	6.79	5.57	6.12	5.23	5.92	5.76	6.10	5.76	6.10	5.76	5.82	5.70	5.20
5.7	4.11	4.66	4.82	4.32	4.82	3.55	3.92	4.33	4.49	4.97	4.77	4.62	4.23	4.62	4.08	4.23	4.02	3.88	4.27	4.97
5.8	7.25	7.12	7.09	6.28	6.52	6.57	6.33	6.98	6.68	6.89	6.61	6.47	6.38	6.70	6.62	6.71	7.23	6.70	6.15	6.68
5.9	0.72	1.46	1.34	0.58	0.55	0.58	0.72	1.23	0.63	1.22	0.72	0.66	0.61	1.31	0.55	1.28	0.57	0.58	1.08	0.58
5.10	4.24	4.25	4.06	4.20	4.42	4.26	3.94	4.20	4.04	4.13	4.00	3.94	3.89	4.16	3.92	4.10	4.25	4.04	4.10	3.96
6.1	7.10	7.05	6.84	6.52	6.76	7.03	5.94	6.79	7.25	7.44	6.47	6.52	7.37	7.39	6.11	6.64	7.87	7.05	6.98	7.49
6.2	2.74	2.74	2.99	2.95	2.95	2.74	3.03	3.10	3.01	2.74	2.15	2.74	2.97	2.94	2.74	3.03	2.92	2.74	3.06	2.74
6.3	5.47	4.99	5.66	4.83	5.26	5.22	4.76	4.99	4.85	5.06	5.17	4.87	4.67	5.20	5.36	5.06	4.97	4.74	5.08	5.43
6.4	2.98	3.00	2.82	3.13	3.05	3.05	3.21	3.10	2.95	2.65	3.05	2.75	2.70	3.05	2.95	2.93	3.00	2.60	2.72	2.90
6.5	4.97	4.76	4.91	5.05	4.39	4.70	5.07	5.11	5.22	5.09	4.93	5.09	5.49	4.99	4.97	4.82	4.72	4.45	5.01	5.01
A.1	8.56	8.59	9.14	9.09	9.01	9.20	9.20	9.03	9.03	8.91	8.84	9.06	9.01	8.92	8.87	8.99	8.79	8.83	8.88	8.88
A.2	5.25	5.13	5.37	5.13	5.21	5.11	5.20	5.17	5.46	5.30	5.28	5.16	5.34	5.08	4.99	5.25	5.20	5.20	5.13	5.26
A.3	5.79	4.83	5.01	5.57	5.24	5.13	5.10	5.16	5.20	4.77	4.94	4.54	5.07	5.70	5.07	5.73	5.50	5.99	5.16	5.65
A.4	4.90	5.40	4.94	5.11	4.96	4.87	5.28	5.20	4.87	5.11	5.28	4.99	5.23	4.81	5.09	4.97	5.16	5.10	4.86	5.11
A.5	1.10	1.19	1.17	1.21	1.33	1.19	1.17	1.05	1.10	1.14	1.34	1.05	1.29	1.24	1.34	1.20	1.27	1.09	1.27	1.27
B.1	8.21	8.12	8.12	8.12	7.87	7.87	8.26	8.16	8.89	7.58	8.55	8.07	8.36	8.55	7.97	8.36	7.39	8.41	8.21	8.12
B.2	11.75	11.97	11.23	11.18	12.15	12.11	10.83	11.40	11.75	10.75	12.50	11.71	10.83	11.93	11.18	10.75	11.54	11.40	11.62	10.79
B.3	3.21	4.04	3.58	3.92	4.33	3.50	2.96	3.71	3.46	4.08	3.96	3.38	3.50	3.75	4.08	3.08	3.75	3.08	2.83	3.46
B.4	6.33	6.33	6.33	6.33	6.33	6.33	6.62	6.33	6.33	6.33	6.84	6.33	5.86	6.33	6.33	6.33	6.33	6.58	6.33	6.33
B.5	1.39	1.39	1.39	1.39	1.39	1.39	1.39	1.39	1.39	1.39	1.39	1.39	1.39	1.39	1.39	1.39	1.39	1.39	1.39	1.39
C.1	3.35	3.52	3.35	3.29	3.22	3.25	3.39	3.30	3.35	3.25	3.45	3.52	3.38	3.30	3.52	3.39	3.37	3.31	3.30	3.41
C.2	4.84	5.08	5.39	4.75	5.14	4.60	5.48	4.73	5.56	5.33	5.14	4.81	5.24	5.21	5.62	5.81	4.89	5.19	5.13	5.07
C.3	9.57	9.44	9.26	9.40	9.37	9.68	9.95	9.38	9.33	8.82	9.31	9.36	9.56	9.73	9.44	9.55	8.86	9.57	8.81	9.48
C.4	8.89	9.27	9.41	9.33	9.42	8.80	9.24	9.63	9.30	10.12	9.36	9.24	9.12	9.50	9.09	8.75	8.92	9.06	9.32	8.78
C.5	6.42	6.84	7.01	6.54	7.01	6.81	7.15	7.30	6.93	7.69	6.43	6.82	6.99	6.64	6.93	7.09	7.09	7.13	6.33	6.74
D.1	8.64	7.67	7.39	6.94	7.89	7.44	6.78	7.28	7.44	7.67	7.72	7.59	7.83	7.78	8.87	7.83	7.33	7.00	7.78	7.94
D.2	6.06	6.06	6.06	6.06	6.06	6.06	6.06	6.06	6.06	5.66	6.06	6.06	6.06	5.71	5.71	6.06	6.06	6.06	6.06	6.06
D.3	9.44	9.44	9.31	9.07	9.21	9.40	9.07	9.21	9.21	9.07	9.72	9.21	9.17	9.21	9.21	9.72	9.40	9.31	9.72	9.72
D.4	5.97	6.40	5.91	5.48	6.08	6.40	6.02	6.45	6.83	5.81	5.97	6.02	6.18	6.18	6.08	6.18	6.29	6.02	6.61	6.08
D.5	6.61	7.05	6.56	6.89	6.72	6.83	6.78	6.94	6.50	7.21	6.94	6.45	6.56	6.72	6.72	6.89	6.39	6.67	6.83	6.67
NRE.1	3.45	3.45	3.45	3.45	3.45	3.45	3.45	3.45	3.45	3.45	3.45	3.45	3.45	3.45	3.45	3.45	3.45	3.45	3.45	3.45
NRE.2	14.89	15.00	15.00	15.44	15.33	13.33	15.00	15.11	14.56	14.67	15.11	14.89	15.67	14.67	15.44	15.11	15.11	15.33	14.89	15.44
NRE.3	22.72	22.35	22.47	23.46	22.59	23.95	22.35	22.72	23.58	22.59	21.60	22.59	22.47	22.72	22.96	22.22	23.21	22.35	18.02	23.09
NRE.4	17.86	17.86	17.86	17.86	17.86	17.86	17.86	17.86	17.86	17.86	17.86	17.86	17.86	17.86	17.86	17.86	17.86	17.86	16.90	17.86
NRE.5	7.14	7.14	7.14	7.14	7.14	7.14	7.14	7.14	7.14	7.14	7.14	7.14	7.14	7.14	7.14	7.14	7.14	7.14	7.14	7.14
NRF.1	21.43	21.43	21.43	21.43	21.43	21.43	21.43	21.43	21.43	21.43	21.43	21.43	21.43	21.43	21.43	21.43	21.43	21.43	21.43	21.43
NRF.2	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00	18.22	20.00
NRF.3	21.43	21.43	21.43	21.43	21.43	21.43	21.43	21.43	21.43	21.43	21.43	21.43	21.43	21.43	21.43	21.43	21.43	21.43	21.43	21.43
NRF.4	23.33	24.05	23.10	24.05	23.81	22.62	24.05	20.48	22.86	23.81	23.33	23.57	24.29	22.86	23.33	24.05	21.43	23.10	21.43	23.81
NRF.5	23.08	23.08	23.08	23.08	23.08	23.08	23.08	23.08	23.08	23.08	23.08	23.08	23.08	23.08	23.08	23.08	23.08	23.08	23.08	23.08
NRG.1	10.19	10.32	10.17	10.40	10.21	10.36	10.28	10.23	10.44	10.32	10.30	10.51	10.45	10.19	10.27	10.53	10.28	10.42	10.36	10.34
NRG.2	8.66	8.77	8.72	8.70	8.44	8.87	8.79	8.83	8.96	8.94	8.92	8.87	9.18	8.70	9.05	8.61	8.85	8.96	9.05	8.90
NRG.3	9.98	10.02	9.88	9.84	10.02	9.94	10.02	9.94	9.94	9.98	9.94	10.00	9.98	9.96	9.96	9.94	9.94	10.06	10.00	9.96
NRG.4	9.09	9.52	9.29	8.85	9.21	9.21	9.13	9.21	9.09	8.95	9.07	9.09	9.19	8.99	9.03	8.95	9.15	9.27	9.13	9.29
NRG.5	10.04	9.66	9.78	9.90	9.82	9.90	10.00	10.14	9.90	9.52	9.92	9.96	9.82	9.98	9.82	9.98	10.26	10.04	10.14	9.94
NRH.1	14.39	15.08	15.40	14.97	15.03	15.29	15.08	14.87	14.87	14.87	15.24	14.97	14.76	15.45	14.92	14.92	15.13	14.55	15.13	14.92
NRH.2	6.35	6.35	6.35	6.35	6.35	6.35	6.35	6.35	6.35	6.35	6.35	6.35	6.35	6.35	6.35	6.35	6.35	6.35	6.35	6.35
NRH.3	16.95	16.95	16.95	16.95	16.95	16.95	16.95	16.95	16.95	16.95	16.95	16.95	16.95	16.95	16.95	16.95	16.95	16.95	16.95	16.95
NRH.4	15.52	15.29	15.86	15.69	15.52	15.52	15.52	15.86	15.92	15.52	15.52	15.52	16.03	15.80	15.52	15.52	15.52	15.23	15.34	15.80
NRH.5	10.91	10.91	10.91	10.91	10.91	10.91	10.91	10.91	10.91	10.91	10.91	10.91	10.91	10.91	10.91	10.91	10.91	10.91	10.91	10.91
CYC.6	18.33	20.00	18.33	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00	18.33	20.00	20.00	21.67	20.00	20.00	20.00	20.00	20.00
CYC.7	24.31	25.00	24.31	23.61	25.00	25.69	23.61	25.00	25.00	25.69	25.00	25.00	24.31	24.31	25.00	25.00	24.31	24.31	25.69	25.00
CYC.8	24.42	25.00	24.71	25.00	24.71	24.42	24.42	24.42	24.42	25.29	25.00	24.71	24.13	23.55	25.29	24.42	24.71	24.42	23.84	24.42
CLR.10-4	20.00	16.00	20.00	16.00	16.00	16.00	16.00	20.00	16.00	20.00	16.00	16.00	20.00	16.00	20.00	20.00	16.00	20.00	16.00	16.00
CLR.11-4	30.43	34.78	30.43	30.43	30.43	34.78	30.43	30.43	30.43	30.43	30.43	34.78	30.43	30.43	34.78	30.43	30.43	30.43	30.43	30.43
CLR.12-4	34.78	34.78	34.78	34.78	34.78	34.78	34.78	34.78	34.78	34.78	34.78	34.78	34.78	34.78	34.78	34.78	34.78	34.78	34.78	34.78
CLR.13-4	47.83	52.17	52.17	52.17	52.17	47.83	52.17	47.83	52.17	52.17	52.17	47.83	52.17	47.83	47.83	52.17	52.17	47.83	52.17	47.83

Table 9
Statistical analysis. Percentage of cases in which a binarization technique offers the best significant performance compared to all others.

Discr.	Trans.	Instance set												
		4	5	6	A	B	C	D	NRE	NRF	NRG	NRH	CYC	CLR
D_1	**S_1**	1.87%	2.62%	2.46%	3.03%	1.97%	2.47%	2.08%	3.85%	0.00%	4.55%	2.36%	3.78%	0.35%
D_2		3.79%	2.06%	1.36%	2.90%	2.37%	2.68%	2.08%	0.38%	0.69%	1.22%	0.39%	3.24%	3.90%
D_3		2.72%	2.06%	1.86%	2.64%	2.50%	2.88%	1.77%	7.69%	20.49%	0.78%	0.39%	7.57%	2.48%
D_4		2.99%	2.62%	1.61%	3.03%	1.97%	2.88%	2.49%	0.00%	0.00%	0.44%	1.97%	3.24%	7.80%
D_5		2.08%	2.47%	1.61%	2.11%	4.34%	2.57%	2.28%	0.38%	0.00%	1.11%	0.39%	2.70%	1.42%
D_1	**S_2**	3.15%	2.77%	3.22%	1.71%	2.37%	2.47%	5.50%	0.77%	0.00%	4.88%	3.94%	0.00%	4.96%
D_2		0.11%	0.00%	0.08%	0.00%	0.26%	0.10%	2.28%	0.38%	0.00%	12.32%	3.15%	1.62%	0.00%
D_3		1.60%	1.54%	4.15%	1.84%	5.39%	3.30%	2.39%	0.00%	0.00%	1.55%	0.79%	1.08%	1.77%
D_4		2.40%	2.93%	3.05%	3.95%	2.11%	2.57%	3.01%	0.00%	22.57%	2.33%	1.97%	3.78%	0.00%
D_5		3.47%	4.01%	3.22%	3.29%	1.97%	2.47%	1.35%	0.38%	0.00%	0.44%	2.36%	2.70%	2.48%
D_1	**S_3**	3.15%	2.77%	3.14%	3.82%	2.24%	1.44%	0.00%	0.00%	0.00%	17.54%	16.93%	1.08%	2.13%
D_2		0.11%	0.00%	0.00%	0.13%	0.00%	0.00%	4.15%	5.00%	1.39%	12.10%	1.18%	0.00%	7.09%
D_3		3.21%	2.52%	3.05%	2.24%	2.37%	2.27%	1.77%	3.85%	0.35%	1.55%	0.39%	1.62%	7.09%
D_4		2.88%	2.62%	1.27%	1.98%	2.37%	2.37%	4.36%	2.69%	0.00%	1.00%	1.97%	3.78%	5.32%
D_5		1.98%	2.06%	1.69%	1.98%	4.21%	2.68%	1.04%	1.54%	0.00%	1.22%	2.76%	1.62%	0.71%
D_1	**S_4**	2.30%	2.57%	3.64%	2.50%	2.76%	3.81%	2.18%	0.38%	0.00%	1.00%	0.39%	0.00%	0.00%
D_2		0.27%	0.00%	0.08%	0.26%	0.00%	0.21%	1.66%	1.54%	0.00%	2.00%	1.97%	3.24%	5.67%
D_3		2.19%	4.16%	2.71%	1.98%	1.97%	2.57%	2.60%	0.38%	2.43%	1.66%	3.15%	0.00%	8.51%
D_4		2.94%	2.88%	3.31%	3.43%	2.37%	2.06%	3.12%	0.38%	0.00%	2.00%	0.39%	0.00%	0.71%
D_5		3.15%	2.26%	1.69%	1.71%	2.89%	1.96%	5.09%	0.00%	0.00%	2.33%	4.72%	2.16%	0.00%
D_1	**V_1**	2.51%	3.34%	3.05%	4.22%	2.11%	3.71%	1.56%	0.38%	0.69%	2.00%	7.48%	11.35%	2.13%
D_2		2.08%	1.85%	3.31%	4.08%	1.97%	2.06%	1.97%	1.15%	0.00%	1.11%	3.94%	2.16%	0.00%
D_3		2.24%	2.72%	1.61%	2.24%	2.37%	2.27%	2.39%	0.77%	1.39%	1.78%	0.39%	3.78%	0.00%
D_4		3.90%	2.93%	1.78%	1.84%	2.63%	3.71%	3.53%	0.00%	0.00%	3.44%	1.18%	3.24%	2.48%
D_5		2.56%	2.57%	2.46%	1.71%	2.24%	3.19%	2.18%	0.38%	0.00%	3.11%	1.97%	0.00%	0.71%
D_1	**V_2**	2.30%	3.55%	3.39%	2.77%	2.11%	3.81%	2.80%	15.00%	0.69%	0.44%	1.97%	0.00%	5.32%
D_2		2.35%	2.11%	3.73%	1.98%	3.82%	2.27%	2.28%	1.15%	0.00%	1.11%	1.57%	5.41%	1.42%
D_3		2.56%	1.80%	1.69%	3.95%	1.71%	2.78%	1.77%	0.38%	11.81%	1.44%	1.18%	0.54%	2.13%
D_4		3.10%	3.03%	1.69%	3.03%	2.11%	2.27%	2.39%	3.85%	0.00%	0.44%	0.79%	1.08%	2.48%
D_5		2.72%	2.47%	3.90%	2.37%	3.68%	2.27%	4.78%	3.08%	0.00%	4.33%	1.97%	1.62%	0.71%
D_1	**V_3**	4.11%	1.95%	3.47%	2.24%	1.18%	3.09%	0.52%	5.00%	0.69%	0.55%	1.97%	2.70%	7.09%
D_2		2.88%	2.57%	3.90%	5.53%	1.97%	2.78%	2.80%	0.77%	0.00%	0.44%	1.97%	3.78%	2.84%
D_3		2.30%	4.16%	2.63%	1.84%	3.16%	2.47%	3.12%	1.92%	0.00%	0.89%	0.39%	0.00%	0.00%
D_4		2.08%	2.57%	1.53%	3.29%	1.97%	3.09%	5.30%	2.69%	1.74%	1.33%	0.39%	6.49%	0.35%
D_5		2.94%	2.98%	4.07%	2.50%	2.50%	2.06%	4.67%	0.00%	0.69%	1.22%	2.36%	0.54%	0.00%
D_1	**V_4**	3.26%	1.75%	1.86%	1.98%	3.95%	2.06%	0.83%	0.77%	0.00%	2.33%	1.57%	1.08%	0.00%
D_2		1.50%	4.16%	1.78%	2.11%	3.42%	2.78%	2.28%	0.38%	11.11%	0.67%	1.97%	2.70%	2.84%
D_3		2.35%	3.03%	5.76%	3.29%	2.63%	2.37%	2.08%	1.92%	0.00%	0.33%	11.42%	2.70%	0.00%
D_4		2.67%	2.77%	2.12%	2.64%	4.21%	4.84%	1.04%	30.77%	23.26%	0.44%	3.54%	7.03%	0.00%
D_5		3.21%	2.77%	3.05%	1.84%	3.82%	2.37%	0.52%	0.00%	0.00%	0.55%	0.39%	0.54%	7.09%
		100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%

values are given by

$$rvp_t^f = \frac{1}{5} \sum_{d \in D_1, D_2, \ldots, D_5} rvp_{d,t}^f \quad .$$

Analyzing this table, we could recommend V_3 and V_4 transfer functions for solving small and medium problems and S_3 for solving large problems. Moreover, we check that as problem size grows, the variation observed in term of performance in the transfer functions is increased (*Avg.* field). Hence, it is especially significant to select an adequate transfer function for solving large problems.

Table 11 is sorted based on the indicative order defined for discretization functions in Section 5.2, *i.e.*, from more exploratory to more exploitative strategies. Studying this table, we find the following trend: as problem size grows, discretization techniques with a better exploration to exploitation ratio outperform the others. The same behavior is shown in Table 13, where RVP values are presented for each family regardless of the transfer function considered. To this end, the values are given by

$$rvp_d^f = \frac{1}{8} \sum_{t \in S_1, S_2, \ldots, S_4, V_1, V_2, \ldots, V_4} rvp_{d,t}^f \quad .$$

Analyzing this table, we could recommend D_1 and D_4 discretization functions for solving all the problems. Moreover and as for the transfer functions, we check that as problem size grows, the variation observed in term of performance in the discretization functions is increased. Hence, it is especially significant to select an adequate discretization function for solving large problems. Additionally, if we compare the *Avg.* field of Tables 12 and 13, we check that the values observed are similar. Hence, we conclude that both types of techniques influence the behavior of the solving method and therefore, we cannot focus only on one of them.

In terms of RPD, we study how affects using an adequate binarization technique. Table 14 compares the results obtained through the original BCSO to the binarization techniques analyzed in this work. In this table, $diff_{\overline{rpd}}$ is the difference between the RPD value obtained from the best binarization technique in this work, $\overline{rpd}$ field, and the original BCSO, $\overline{rpd}$(original) field. Analyzing this table, we note that the algorithm provides a clear better behavior when an adequate binarization technique is assumed. This way, the RPD value decreases up to 26.19% for the instance set 4, 16.18% for the instance set 5, 10.23% for the instance set 6, 8.32% for the instance set A, 10.25% for the instance set B, 6.56% for the instance set C, 6.37% for the instance set D, 12.43% for the instance set NRE, 5.90% for the instance set NRF, 5.74% for the instance set NRG, 4.55% for the instance set NRH, 6.20% for the instance set CYC, and 2.08% for the instance set CLR.

From this study and as a summary, we reach the following six major conclusions:

Table 10
RVP metric for each family of instance sets and binarization approach, where the values are grouped by transfer functions.

		Family of instance sets					
Discr.	**Trans.**	**4,5,6**	**A,B**	**C,D**	**NRE,NRF**	**CYC,CLR**	**NRG,NRH**
D_2		96.84%	96.49%	77.73%	97.15%	47.00%	88.49%
D_5		36.23%	38.63%	15.99%	100.00%	69.43%	79.53%
D_1	S_4	23.64%	29.85%	28.55%	99.29%	69.30%	95.96%
D_3		18.67%	47.38%	38.34%	94.79%	25.44%	86.03%
D_4		18.16%	22.82%	38.29%	99.29%	18.06%	93.06%
D_2		99.04%	98.25%	50.47%	88.18%	87.97%	61.47%
D_5		48.61%	17.59%	55.68%	97.15%	61.53%	88.46%
D_1	S_3	18.69%	19.31%	82.81%	100.00%	63.17%	0.00%
D_3		21.27%	38.62%	51.93%	92.24%	78.83%	94.35%
D_4		39.20%	42.13%	19.74%	95.02%	71.93%	91.39%
D_2		98.28%	96.49%	71.53%	99.29%	47.38%	55.12%
D_5		3.99%	29.83%	54.43%	99.29%	82.71%	91.86%
D_1	S_2	17.93%	45.64%	4.89%	98.58%	76.19%	74.41%
D_3		34.53%	3.57%	32.22%	100.00%	35.35%	93.21%
D_4		24.78%	19.31%	33.38%	58.23%	32.47%	87.53%
D_2		35.36%	29.84%	43.30%	98.00%	33.85%	95.32%
D_5		44.73%	14.08%	42.05%	99.29%	83.96%	95.64%
D_1	S_1	37.66%	33.34%	45.76%	92.88%	100.00%	79.94%
D_3		40.38%	31.60%	44.56%	47.85%	36.86%	96.60%
D_4		35.20%	33.34%	35.89%	100.00%	94.74%	93.00%
D_2		33.26%	26.35%	39.60%	78.72%	83.96%	92.36%
D_5		18.97%	24.60%	65.56%	100.00%	94.74%	97.25%
D_1	V_4	38.36%	21.09%	65.53%	98.58%	0.00%	88.67%
D_3		0.00%	21.07%	46.99%	96.44%	71.93%	65.91%
D_4		32.13%	8.81%	29.89%	0.00%	57.52%	88.43%
D_2		16.09%	0.00%	33.40%	98.58%	49.37%	93.00%
D_5		10.40%	33.35%	19.71%	98.71%	82.71%	89.60%
D_1	V_3	14.40%	54.39%	56.96%	89.46%	60.54%	92.68%
D_3		18.47%	33.37%	33.37%	96.44%	80.20%	96.28%
D_4		44.57%	29.83%	0.00%	91.80%	73.56%	94.99%
D_2		26.55%	22.85%	45.74%	97.86%	50.88%	92.21%
D_5		18.45%	19.33%	16.02%	94.31%	95.99%	81.73%
D_1	V_2	17.16%	35.10%	21.12%	70.95%	27.33%	93.00%
D_3		45.65%	24.57%	45.79%	77.44%	100.00%	92.39%
D_4		29.79%	31.59%	44.50%	92.88%	49.25%	96.43%
D_2		35.05%	19.30%	51.91%	97.86%	58.90%	85.36%
D_5		31.89%	47.39%	35.92%	99.29%	43.37%	85.27%
D_1	V_1	20.13%	15.80%	37.21%	98.00%	91.98%	72.50%
D_3		40.98%	38.62%	44.50%	96.01%	79.95%	93.71%
D_4		22.76%	40.38%	13.68%	100.00%	47.87%	86.59%

Table 11
RVP metric for each family of instance sets and binarization approach, where the values are grouped by discretization functions.

		Family of instance sets					
Discr.	**Trans.**	**4,5,6**	**A,B**	**C,D**	**NRE,NRF**	**CYC,CLR**	**NRG,NRH**
	S_4	96.84%	96.49%	77.73%	97.15%	47.00%	88.49%
	S_3	99.04%	98.25%	50.47%	88.18%	87.97%	61.47%
	S_2	98.28%	96.49%	71.53%	99.29%	47.38%	55.12%
D_2	S_1	35.36%	29.84%	43.30%	98.00%	33.85%	95.32%
	V_4	33.26%	26.35%	39.60%	78.72%	83.96%	92.36%
	V_3	16.09%	0.00%	33.40%	98.58%	49.37%	93.00%
	V_2	26.55%	22.85%	45.74%	97.86%	50.88%	92.21%
	V_1	35.05%	19.30%	51.91%	97.86%	58.90%	85.36%
	S_4	36.23%	38.63%	15.99%	100.00%	69.43%	79.53%
	S_3	48.61%	17.59%	55.68%	97.15%	61.53%	88.46%
	S_2	3.99%	29.83%	54.43%	99.29%	82.71%	91.86%
D_5	S_1	44.73%	14.08%	42.05%	99.29%	83.96%	95.64%
	V_4	18.97%	24.60%	65.56%	100.00%	94.74%	97.25%
	V_3	10.40%	33.35%	19.71%	98.71%	82.71%	89.60%
	V_2	18.45%	19.33%	16.02%	94.31%	95.99%	81.73%
	V_1	31.89%	47.39%	35.92%	99.29%	43.37%	85.27%
	S_4	23.64%	29.85%	28.55%	99.29%	69.30%	95.96%
	S_3	18.69%	19.31%	82.81%	100.00%	63.17%	0.00%
	S_2	17.93%	45.64%	4.89%	98.58%	76.19%	74.41%
D_1	S_1	37.66%	33.34%	45.76%	92.88%	100.00%	79.94%
	V_4	38.36%	21.09%	65.53%	98.58%	0.00%	88.67%
	V_3	14.40%	54.39%	56.96%	89.46%	60.54%	92.68%
	V_2	17.16%	35.10%	21.12%	70.95%	27.33%	93.00%
	V_1	20.13%	15.80%	37.21%	98.00%	91.98%	72.50%
	S_4	18.67%	47.38%	38.34%	94.79%	25.44%	86.03%
	S_3	21.27%	38.62%	51.93%	92.24%	78.83%	94.35%
	S_2	34.53%	3.57%	32.22%	100.00%	35.35%	93.21%
D_3	S_1	40.38%	31.60%	44.56%	47.85%	36.86%	96.60%
	V_4	0.00%	21.07%	46.99%	96.44%	71.93%	65.91%
	V_3	18.47%	33.37%	33.37%	96.44%	80.20%	96.28%
	V_2	45.65%	24.57%	45.79%	77.44%	100.00%	92.39%
	V_1	40.98%	38.62%	44.50%	96.01%	79.95%	93.71%
	S_4	18.16%	22.82%	38.29%	99.29%	18.06%	93.06%
	S_3	39.20%	42.13%	19.74%	95.02%	71.93%	91.39%
	S_2	24.78%	19.31%	33.38%	58.23%	32.47%	87.53%
D_4	S_1	35.20%	33.34%	35.89%	100.00%	94.74%	93.00%
	V_4	32.13%	8.81%	29.89%	0.00%	57.52%	88.43%
	V_3	44.57%	29.83%	0.00%	91.80%	73.56%	94.99%
	V_2	29.79%	31.59%	44.50%	92.88%	49.25%	96.43%
	V_1	22.76%	40.38%	13.68%	100.00%	47.87%	86.59%

- We find significant performance differences according to the binarization approach assumed when an SIA (the BCSO) is adapted to the discrete scope. Hence, we conclude that it is crucial to select an adequate binarization approach. Otherwise, it is possible that the algorithm does not reach its full potential as occurs with the original BCSO compared to the recommended configurations obtained in this work. As a direct result of this statement, it is possible that other algorithms could be improved by studying other binarization approaches.
- We conclude that both transfer and discretization functions greatly affect the behavior of the solving method and therefore, we cannot focus only on one of them.
- Regarding transfer functions, we find that v-shape functions are fit for solving limited search space problems and s-shape functions are fit for solving large search space problems. Concretely, we recommend V_3 and V_4 transfer functions for solving small and medium problems, and S_3 for solving large problems.
- Regarding discretization functions, we find that as problem size grows, discretization techniques with a better exploration to exploitation ratio outperform the others. Concretely, we recommend D_1 and D_4 discretization functions for solving all the problems.
- We reach that it is especially significant to select an adequate binarization approach for solving large problems as the variations observed in applying different techniques are higher.
- We appreciably increase the BCSO performance after selecting an adequate binarization approach for each instance. At this point, we cannot recommend this algorithm for solving the SCP,

Table 12
RVP metric for each family of instance sets regardless of the discretization function considered.

	Family of instance sets					
Trans.	**4,5,6**	**A,B**	**C,D**	**NRE,NRF**	**CYC,CLR**	**NRG,NRH**
S_4	22.63%	33.47%	15.55%	92.49%	44.38%	65.36%
S_3	31.03%	28.63%	32.86%	78.29%	16.57%	0.00%
S_2	19.08%	23.34%	14.86%	64.66%	49.56%	40.43%
S_1	22.57%	10.12%	19.10%	50.91%	0.00%	75.96%
V_4	4.75%	0.00%	29.20%	0.00%	34.29%	58.99%
V_3	0.00%	12.31%	0.00%	80.20%	34.80%	79.65%
V_2	8.50%	7.92%	8.33%	47.29%	43.27%	73.07%
V_1	11.84%	14.96%	11.15%	93.00%	29.15%	53.40%
Avg.	15.05%	16.34%	16.38%	63.35%	31.50%	55.86%

Table 13
RVP metric for each family of instance sets regardless of the transfer function considered.

	Family of instance sets					
Discr.	**4,5,6**	**A,B**	**C,D**	**NRE,NRF**	**CYC,CLR**	**NRG,NRH**
D_2	41.26%	28.64%	33.92%	72.75%	3.93%	32.62%
D_5	4.13%	0.00%	15.39%	92.65%	47.67%	55.30%
D_1	0.00%	5.17%	21.80%	67.90%	12.16%	0.00%
D_3	5.23%	2.43%	20.92%	39.31%	17.81%	59.81%
D_4	9.58%	0.59%	0.00%	0.00%	0.00%	66.19%
Avg.	12.04%	7.37%	18.41%	54.52%	16.31%	42.78%

Table 14
Comparing the results obtained through the original BCSO to the binarization techniques analyzed in this work.

Inst.	Trans.	Discr.	z_{opt}	z_{best}	z_{avg}	$\overline{rpd}$	$\overline{rpd}(original)$	$diff_{\overline{rpd}}$
4.1	S_2	D_1	429	432	440.07	2.58	6.44	59.94%
4.2	V_1	D_4	512	517	529.87	3.49	6.22	43.89%
4.3	V_4	D_5	516	531	552.77	7.13	7.93	10.09%
4.4	S_2	D_1	494	496	510.23	3.29	3.90	15.64%
4.5	S_2	D_5	512	514	523.23	2.19	2.75	20.36%
4.6	V_2	D_3	560	560	566.10	1.09	1.24	12.10%
4.7	V_3	D_1	430	434	437.53	1.75	2.18	19.72%
4.8	V_4	D_5	492	494	511.07	3.88	4.98	22.09%
4.9	V_3	D_5	641	660	674.37	5.21	6.04	13.74%
4.10	V_3	D_2	514	518	524.93	2.13	2.63	19.01%
Avg.	–	–	–	–	–	3.27	4.43	**26.19%**
5.1	V_1	D_1	253	258	261.54	3.37	3.77	10.61%
5.2	V_3	D_3	302	306	313.30	3.74	5.11	26.81%
5.3	S_2	D_4	226	229	232.73	2.98	3.58	16.76%
5.4	V_3	D_3	242	242	245.13	1.29	1.49	13.42%
5.5	S_1	D_3	211	216	219.43	4.00	4.31	7.19%
5.6	V_1	D_3	213	217	223.41	4.89	6.12	20.10%
5.7	V_2	D_1	293	294	303.40	3.55	4.60	22.83%
5.8	V_4	D_4	288	294	305.70	6.15	6.42	4.21%
5.9	S_2	D_5	279	280	280.42	0.51	1.49	65.77%
5.10	S_4	D_3	265	271	274.80	3.70	3.92	5.61%
Avg.	–	–	–	–	–	3.42	4.08	**16.18%**
6.1	V_2	D_2	138	143	146.20	5.94	6.57	9.59%
6.2	V_3	D_1	146	146	149.13	2.15	2.74	21.53%
6.3	V_3	D_3	145	148	151.77	4.67	5.15	9.32%
6.4	V_4	D_3	131	133	134.40	2.60	2.65	1.89%
6.5	V_1	D_5	161	165	168.07	4.39	4.87	9.86%
Avg.	–	–	–	–	–	3.95	4.40	1**0.23%**
A.1	V_1	D_1	253	271	274.67	8.56	9.16	6.55%
A.2	S_3	D_1	252	259	264.27	4.87	5.16	5.62%
A.3	V_3	D_2	232	238	242.53	4.54	5.19	12.52%
A.4	S_2	D_4	234	241	244.90	4.66	5.07	8.09%
A.5	V_2	D_3	236	237	238.47	1.05	1.27	17.32%
Avg.	–	–	–	–	–	4.74	5.17	**8.32%**
B.1	S_1	D_5	69	70	73.70	6.81	8.79	22.53%
B.2	S_2	D_3	76	80	83.80	10.26	10.26	0.00%
B.3	S_3	D_5	80	80	82.27	2.83	3.50	19.14%
B.4	V_3	D_3	79	81	83.63	5.86	6.33	7.42%
B.5	S_1	D_1	72	73	73.00	1.39	1.39	0.00%
Avg.	–	–	–	–	–	5.43	6.05	**10.25%**
C.1	V_1	D_5	227	232	234.30	3.22	3.39	5.01%
C.2	V_2	D_1	219	225	229.07	4.60	5.43	15.29%
C.3	S_1	D_2	243	251	264.07	8.67	9.42	7.96%
C.4	S_1	D_3	219	231	237.70	8.54	8.86	3.61%
C.5	V_4	D_4	215	222	228.60	6.33	6.47	2.16%
Avg.	–	–	–	–	–	6.27	6.71	**6.56%**
D.1	S_4	D_4	60	60	64.03	6.72	7.33	8.32%
D.2	S_4	D_5	66	69	69.70	5.61	6.06	7.43%
D.3	S_1	D_2	72	76	78.50	9.03	9.44	4.34%
D.4	S_3	D_3	62	63	65.37	5.43	5.91	8.12%
D.5	S_2	D_4	61	64	64.83	6.28	6.56	4.27%
Avg.	–	–	–	–	–	6.61	7.06	**6.37%**
NRE.1	S_1	D_1	29	30	30.00	3.45	3.45	0.00%
NRE.2	V_2	D_1	30	34	34.00	13.33	15.56	14.33%
NRE.3	V_4	D_4	27	29	31.87	18.02	23.21	22.36%
NRE.4	V_4	D_4	28	32	32.73	16.90	17.86	5.38%
NRE.5	S_1	D_1	28	30	30.00	7.14	7.14	0.00%
Avg.	–	–	–	–	–	11.77	13.44	**12.43%**
NRF.1	S_1	D_1	14	17	17.00	21.43	21.43	0.00%
NRF.2	S_1	D_3	15	16	17.70	18.00	20.00	10.00%
NRF.3	S_1	D_1	14	17	17.00	21.43	21.43	0.00%
NRF.4	V_2	D_3	14	15	16.87	20.48	25.00	18.08%
NRF.5	S_1	D_1	13	16	16.00	23.08	23.08	0.00%
Avg.	–	–	–	–	–	20.88	22.19	**5.90%**
NRG.1	S_1	D_1	176	191	193.10	9.72	10.30	5.63%
NRG.2	S_3	D_1	154	165	166.43	8.07	8.79	8.19%
NRG.3	S_2	D_2	166	182	182.00	9.64	9.92	2.82%
NRG.4	V_1	D_4	168	180	182.87	8.85	9.15	3.28%
NRG.5	S_3	D_1	168	183	183.00	8.93	9.80	8.88%
Avg.	–	–	–	–	–	9.04	9.59	**5.74%**
NRH.1	S_3	D_3	63	69	71.00	12.70	15.19	16.39%
NRH.2	S_1	D_1	63	67	67.00	6.35	6.35	0.00%
NRH.3	S_1	D_1	59	69	69.00	16.95	16.95	0.00%
NRH.4	S_2	D_5	58	64	66.73	15.06	15.52	2.96%

(*continued on next page*)

Table 14 (*continued*)

Inst.	Trans.	Discr.	z_{opt}	z_{best}	z_{avg}	$\overline{rpd}$	$\overline{rpd}$(*original*)	$diff_{\overline{rpd}}$
NRH.5	S_1	D_1	55	61	61.00	10.91	10.91	0.00%
Avg.	–	–	–	–	–	12.39	12.98	**4.55%**
CYC.6	S_1	D_3	60	66	72.00	18.33	20.00	8.35%
CYC.7	S_3	D_5	144	172	178.00	23.61	25.00	5.56%
CYC.8	V_3	D_4	344	418	435.00	23.55	24.71	4.69%
Avg.	–	–	–	–	–	21.83	23.24	**6.20%**
CLR.10	S_1	D_1	25	28	29.00	16.00	16.00	0.00%
CLR.11	S_1	D_1	23	28	30.00	30.43	30.43	0.00%
CLR.12	S_1	D_1	23	29	31.00	34.78	34.78	0.00%
CLR.13	S_1	D_5	23	31	34.00	47.83	52.17	8.32%
Avg.	–	–	–	–	–	32.26	33.35	**2.08%**

because it is far from other current state-of-the-art techniques in terms of performance. However, the BCSO has proven to be a good algorithm for studying a relevant aspect, such as binarization. Hence, we recommend this algorithm as a possible testing bench for future works.

7. Implementation details

Both the problem definition introduced in Section 3 and the solving methodology discussed in Section 4 were coded in Java assuming NetBeans IDE 7.1 and executed on a 2.53 GHz Intel Core i3 M380 processor with 3 GB RAM under Windows 7. Regarding the statistical tools, the Wilcoxon–Mann–Whitney's test was taken from Fonseca, Knowles, Thiele, and Zitzler and both Shapiro–Wilk's and Kolmogorov–Smirnov–Lilliefor's tests were taken from the IBM SPSS software.

8. Final remarks

The SCP is a traditional optimization problem widely considered for designing expert systems. We find many papers assuming metaheuristics for solving the SCP in the current literature. However, many metaheuristics are defined for solving continuous optimization problems, specially SIAs, while the SCP is a discrete problem. Hence, such algorithms should be adapted for working on the discrete scope. However, most authors did not perform any study to select a concrete binarization approach. This circumstance might lead to the conclusion that selecting a concrete binarization technique does not influence the behavior of the algorithm, but rather the general approach of the metaheuristic. This situation led us to write this paper focusing on the inherent difficulty in binarization of metaheuristics designed for continuous optimization, when solving a discrete optimization problem, concretely the SCP.

With the purpose of analyzing such difficulty, we consider a recent SIA which was later adapted to the discrete scope for solving the SCP, the BCSO algorithm. We change the original formulation of BCSO by combining eight transfer functions and five discretization functions from the current literature, *i.e.*, forty binarization techniques.

Based on an accepted statistical methodology, we analyze the results obtained while solving two problems sets: the standard OR-library and the unicost benchmark. As a result of this study, we reach six major conclusions: a)It is crucial to select an adequate binarization approach to guarantee that the solving algorithm reaches its full potential. b) Both transfer and discretization functions affect the behavior of the solving method. c) Regarding transfer functions, v-shape functions are fit for solving limited search space problems and s-shape functions are fit for solving large search space problems. We recommend V_3 and V_4 for solving small and medium problems and S_3 for solving large problems. d) As problem size grows, discretization techniques with a better exploration to exploitation ratio outperform the others. We recommend D_1 and D_4 for solving all the problems. e) it is especially significant to select an adequate binarization approach for solving large problems. f) We recommend BCSO as a possible testing bench for future works.

As future lines of research, it would be interesting to consider other metaheuristics, including some of which give the best results solving the SCP in the current literature. To this end, we should complete the benchmark by adding large SCP problems to analyze the differences observed, *e.g.*, the dataset available in http://people.sabanciuniv.edu/sibirbil/scp/.This is due to such techniques get optimal or near optimal solutions for the instances assumed in this work.

Acknowledgements

The authors thank the anonymous referees for comments and suggestions which have led to an improved version of this paper. Broderick Crawford is supported by grant CONICYT/ FONDECYT/ REGULAR/ 1140897. Ricardo Soto is supported by grant CONICYT/ FONDECYT/ REGULAR/ 1160455. This research was partially funded by CORFO Program Ingeniería 2030 PUCV - Consortium of Chilean Engineering Faculties.

References

Adamec, R. (1976). The interaction of hunger and prey in the domestic cat (felis catus): An adaptive hierarchy? *Behavioural Biology, 18*, 263–272.
Adler, H. (1995). Some factors of observation learning in cats. *The Journal of Genetic Psychology, 86*, 159–177.
Adulyasak, Y., Cordeau, J.-F., & Jans, R. (2015). The production routing problem: A review of formulations and solution algorithms. *Computers & Operations Research, 55*, 141–152.
Aickelin, U. (2004). An indirect genetic algorithm for set covering problems. *Computers & Operations Research, 31*, 1118–1126.
Bai, R., Xue, N., Chen, J., & Roberts, G. W. (2015). A set-covering model for a bidirectional multi-shift full truckload vehicle routing problem. *Transportation Research Part B: Methodological, 79*, 134–148.
Balas, E., & Carrera, M. C. (1996). A dynamic subgradient-based branch-and-bound procedure for set covering. *Operations Research, 44*, 875–890.
Bar-Yejuda, R., & Even, S. (1981). A linear-time approximation algorithm for the weighted vertex cover problem. *Journal of Algorithms, 2*, 198–203.
Beasley, J. (1990). A lagrangian heuristic for set covering problems. *Naval Research Logistics, 37*, 151–164.
Beasley, J., & Jornsten, K. (1992). Enhancing an algorithm for set covering problems. *European Journal of Operational Research, 58*, 293–300.
Beasley, J. E. (1987). An algorithm for set covering problem. *European Journal of Operation Research, 31*, 85–93.
Beasley, J. E. (2016). Or-library. http://people.brunel.ac.uk/~mastjjb/jeb/info.html.
Beasley, J. E., & Chu, P. C. (1996). A genetic algorithm for the set covering problem. *European Journal of Operation Research, 94*, 392–404.
Birbil, c. I., & Fang, S.-C. (2003). An electromagnetism-like mechanism for global optimization. *Journal of Global Optimization, 25*, 263–282.
Cacchiani, V., Hemmelmayr, V. C., & Tricoire, F. (2014). A set-covering based heuristic algorithm for the periodic vehicle routing problem. *Discrete Applied Mathematics, 163*, 53–64.
Caprara, A., Fischetti, M., & Toth, P. (1999). A heuristic method for the set covering problem. *Operations Research, 47*, 730–743.
Caprara, A., Fischetti, M., & Toth, P. (2000). Algorithms for the set covering problem. *Annals of Operations Research, 98*, 353–371.

Ceria, S., Nobili, P., & Sassano, A. (1998). A lagrangian-based heuristic for large-scale set covering problems. *Mathematical Programming, 81*, 215–228.
Chen, S., & Shen, Y. (2013). An improved column generation algorithm for crew scheduling problems. *Journal of Information and Computational Science, 10*, 175–183.
Chu, P. C., & Beasley, J. E. (1997). A genetic algorithm for the generalised assignment problem. *Computers & Operations Research, 24*, 17–23.
Chu, S.-C., Tsai, P.-W., & Pan, J.-S. (2006). Cat swarm optimization. In *PRICAI 2006: Trends in artificial intelligence* (pp. 854–858). Springer.
Chvatal, V. (1979). A greedy heuristic for the set-covering problem. *Mathematics of Operations Research, 4*, 233–235.
Colombo, F., Cordone, R., & Lulli, G. (2015). A variable neighborhood search algorithm for the multimode set covering problem. *Journal of Global Optimization, 63*, 461–480.
Crawford, B., & Castro, C. (2006). Integrating lookahead and post processing procedures with aco for solving set partitioning and covering problems. In *Artificial intelligence and soft computing–ICAISC 2006* (pp. 1082–1090). Springer.
Crawford, B., Soto, R., Aballay, F., Misra, S., Johnson, F., & Paredes, F. (2015a). A teaching-learning-based optimization algorithm for solving set covering problems. In *Computational science and its applications–ICCSA 2015* (pp. 421–430).
Crawford, B., Soto, R., Berríos, N., Johnson, F., Paredes, F., Castro, C., et al. (2015b). A binary cat swarm optimization algorithm for the non-unicost set covering problem. *Mathematical Problems in Engineering, 2015*.
Crawford, B., Soto, R., Cuesta, R., & Paredes, F. (2014a). Application of the artificial bee colony algorithm for solving the set covering problem. *The Scientific World Journal, 2014*, 1–8.
Crawford, B., Soto, R., Monfroy, E., Paredes, F., & Palma, W. (2011). A hybrid ant algorithm for the set covering problem. *International Journal of Physical Science, 6*, 4667–4673.
Crawford, B., Soto, R., Olivares-Suarez, M., Palma, W., Paredes, F., Olguin, E., et al. (2014b). A binary coded firefly algorithm that solves the set covering problem. *Romanian Journal of Information Science and Technology, 17*, 252–264.
Crawford, B., Soto, R., Peña, C., Riquelme-Leiva, M., Torres-Rojas, C., Johnson, F., et al. (2015c). Binarization methods for shuffled frog leaping algorithms that solve set covering problems. In *Software engineering in intelligent systems* (pp. 317–326).
Crawford, B., Soto, R., Riquelme-Leiva, M., Peña, C., Torres-Rojas, C., Johnson, F., et al. (2015d). Modified binary firefly algorithms with different transfer functions for solving set covering problems. In *Software engineering in intelligent systems* (pp. 307–315).
Crawford, B., Soto, R., Torres-Rojas, C., Peña, C., Riquelme-Leiva, M., Misra, S., et al. (2015e). A binary fruit fly optimization algorithm to solve the set covering problem. In *Computational science and its applications–ICCSA 2015* (pp. 411–420).
Dasgupta, D., & Michalewicz, Z. (2013). *Evolutionary algorithms in engineering applications*. Springer Science & Business Media.
Dorigo, M., & Gambardella, L. M. (1997). Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation, 1*, 53–66.
El-Darzi, E., & Mitra, G. (1990). Set covering and set partitioning: A collection of test problems. *Omega, 18*, 195–201.
Elizondo-Amaya, M. G., Rios-Mercado, R. Z., & Diaz, J. A. (2014). A dual bounding scheme for a territory design problem. *Computers & Operations Research, 44*, 193–205.
Eusuff, M., Lansey, K., & Pasha, F. (2006). Shuffled frog-leaping algorithm: A memetic meta-heuristic for discrete optimization. *Engineering Optimization, 38*, 129–154.
Farahani, R. Z., Asgari, N., Heidari, N., Hosseininia, M., & Goh, M. (2012). Covering problems in facility location: A review. *Computers & Industrial Engineering, 62*, 368–407.
Feo, T. A., & Resende, M. G. (1989). A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters, 8*, 67–71.
Fisher, M. L., & Kedia, P. (1990). Optimal solution of set covering/partitioning problems using dual heuristics. *Management Science, 36*, 674–688.
Fonseca, C., Knowles, J., Thiele, L., & Zitzler, E.. Performance assessment tool suite. http://www.tik.ee.ethz.ch/pisa/?page=assessment.php.
Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: A Guide to the theory of NP-Completeness*. W. H. Freeman & Co.
Grohmann, S., Urosevic, D., Carrizosa, E., & Mladenovic, N. (2016). Solving multifacility huff location models on networks using metaheuristic and exact approaches. *Computers & Operations Research*. -, –.
Haouari, M., & Chaouachi, J. (2002). A probabilistic greedy search algorithm for combinatorial optimisation with application to the set covering problem. *Journal of the Operational Research Society, 53*, 792–799.
Hays, W., & Winkler, R. (1970). *Statistics: Probability, inference, and decision*. Holt, Rinehart and Winston.
Juette, S., & Thonemann, U. W. (2012). Divide-and-price: A decomposition algorithm for solving large railway crew scheduling problems. *European Journal of Operational Research, 219*, 214–223.
Karaboga, D., & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (abc) algorithm. *Journal of Global Optimization, 39*, 459–471.
Karp, R. M. (1972). *Reducibility among combinatorial problems*. Springer.
Kennedy, J., & Eberhart, R. C. (1997). A discrete binary version of the particle swarm algorithm. In *Systems, man, and cybernetics, 1997. computational cybernetics and simulation., 1997 IEEE international conference on: vol. 5* (pp. 4104–4108).
Lessing, L., Dumitrescu, I., & Stutzle, T. (2004). A comparison between aco algorithms for the set covering problem. In *Ant colony optimization and swarm intelligence* (pp. 1–12).
Lilliefors, H. W. (1967). On the kolmogorov-smirnov test for normality with mean and variance unknown. *Journal of the American Statistical Association, 62*, 399–402.
Lu, Y., & Vasko, F. J. (2015). An or practitioner's solution approach for the set covering problem. *International Journal of Applied Metaheuristic Computing, 6*, 1–13.
Lust, T., & Tuyttens, D. (2014). Variable and large neighborhood search to solve the multiobjective set covering problem. *Journal of Heuristics, 20*, 165–188.
Mann, H. B., & Whitney, D. R. (1947). On a test of whether one of two random variables is stochastically larger than the other. *Annals of Mathematical Statistics, 1*, 50–60.
de Mare, R., Spliet, R., & Huisman, D. (2014). A branch-and-price approach for a ship routing problem with multiple products and inventory constraints. In *Operations research proceedings 2013* (pp. 97–103). Springer.
Mirjalili, S., & Hashim, S. Z. M. (2012). Bmoa: Binary magnetic optimization algorithm. *International Journal of Machine Learning and Computing, 2*, 204.
Mirjalili, S., & Lewis, A. (2013). S-Shaped versus v-shaped transfer functions for binary particle swarm optimization. *Swarm and Evolutionary Computation, 9*, 1–14.
Mirjalili, S., Mohd, S., Taherzadeh, G., Mirjalili, S., & Salehi, S. (2011). A study of different transfer functions for binary version of particle swarm optimization. In *Swarm and evolutionary computation* (pp. 169–174).
Naji-Azimi, Z., Toth, P., & Galli, L. (2010). An electromagnetism metaheuristic for the unicost set covering problem. *European Journal of Operational Research, 205*, 290–300.
Niroomand, S., & Vizvari, B. (2015). Exact mathematical formulations and metaheuristic algorithms for production cost minimization: A case study of the cable industry. *International Transactions in Operational Research, 22*, 519–544.
Pan, W.-T. (2012). A new fruit fly optimization algorithm: Taking the financial distress model as an example. *Knowledge-Based Systems, 26*, 69–74.
Rao, R. V., Savsani, V. J., & Vakharia, D. (2011). Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems. *Computer-Aided Design, 43*, 303–315.
Rashedi, E., Nezamabadi-Pour, H., & Saryazdi, S. (2010). Bgsa: Binary gravitational search algorithm. *Natural Computing, 9*, 727–745.
Reggia, J. A., Nau, D. S., & Wang, P. Y. (1983). Diagnostic expert systems based on a set covering model. *International Journal of Man-Machine Studies, 19*, 437–460.
Ren, Z., Feng, Z., Ke, L., & Zhang, Z. (2010). New ideas for applying ant colony optimization to the set covering problem. *Computers & Industrial Engineering, 58*, 774–784.
Shapiro, S. S., & Wilk, M. B. (1965). An analysis of variance test for normality (complete samples). *Biometrika, 52*, 591–611.
Sharafi, Y., Khanesar, M. A., & Teshnehlab, M. (2013). Discrete binary cat swarm optimization algorithm. In *3rd International Conference on Computer, Control & Communication (IC4)* (pp. 1–6). IEEE.
Simeone, B., Nouno, G., Mezzadri, M., & Lari, I. (2014). A boolean theory of signatures for tonal scales. *Discrete Applied Mathematics, 165*, 283–294.
Singh, A. (2009). An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem. *Applied Soft Computing, 9*, 625–631.
Soto, R., Crawford, B., Muñoz, A., Johnson, F., & Paredes, F. (2015). Pre-processing, repairing and transfer functions can help binary electromagnetism-like algorithms. In *Artificial intelligence perspectives and applications* (pp. 89–97). Springer.
Sundar, S., & Singh, A. (2012a). A hybrid heuristic for the set covering problem. *Operational Research, 12*, 345–365.
Sundar, S., & Singh, A. (2012b). A hybrid heuristic for the set covering problem. *Operational Research, 12*, 345–365.
Vasko, F. J. (1984). An efficient heuristic for large set covering problems. *Naval Research Logistics Quarterly, 31*, 163–171.
Vasko, F. J., Lu, Y., & Zyma, K. (2016). What is the best greedy-like heuristic for the weighted set covering problem? *Operations Research Letters, 44*, 366–369.
Vidal, T., Crainic, T. G., Gendreau, M., & Prins, C. (2013). Heuristics for multi-attribute vehicle routing problems: A survey and synthesis. *European Journal of Operational Research, 231*, 1–21.
Yang, X.-S. (2010). Firefly algorithm, stochastic test functions and design optimisation. *International Journal of Bio-Inspired Computation, 2*, 78–84.
Zhang, J., Wei, Q., & Chen, G. (2014). A heuristic approach for -representative information retrieval from large-scale data. *Information Sciences, 277*, 825–841.
Zyma, K., Lu, Y., & Vasko, F. J. (2015). Teacher training enhances the teaching-learning-based optimization metaheuristic when used to solve multiple-choice multidimensional knapsack problems. *International Journal of Metaheuristics, 4*, 268–293.