Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 5473501, 24 pages
https://doi.org/10.1155/2020/5473501

Research Article

Exploring Further Advantages in an Alternative Formulation for the Set Covering Problem

Jose M. Lanza-Gutierrez,[1] **N. C. Caballe**,[2] **Broderick Crawford**,[3] **Ricardo Soto**,[3] **Juan A. Gomez-Pulido**,[4] **and Fernando Paredes**[5]

[1]*Carlos III University of Madrid, Electronic Technology Department, Leganés, Spain*
[2]*University of Alcalá, Physics and Mathematics Department, Faculty of Sciences, Alcalá de Henares, Spain*
[3]*Pontificia Universidad Católica de Valparaíso, Escuela de Ingeniería Informática, Valparaíso, Chile*
[4]*University of Extremadura, School of Technology, Cáceres, Spain*
[5]*Universidad Diego Portales, Escuela de Ingeniería Industrial, Santiago de Chile, Chile*

Correspondence should be addressed to Jose M. Lanza-Gutierrez; jlanza@ing.uc3m.es

Received 28 February 2020; Accepted 1 June 2020; Published 15 July 2020

Guest Editor: Dilbag Singh

Copyright © 2020 Jose M. Lanza-Gutierrez et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The set covering problem (SCP) is an NP-complete optimization problem, fitting with many problems in engineering. The traditional SCP formulation does not directly address both solution unsatisfiability and set redundancy aspects. As a result, the solving methods have to control these aspects to avoid getting unfeasible and nonoptimized in cost solutions. In the last years, an alternative SCP formulation was proposed, directly covering both aspects. This alternative formulation received limited attention because managing both aspects is considered straightforward at this time. This paper questions whether there is some advantage in the alternative formulation, beyond addressing the two issues. Thus, two studies based on a metaheuristic approach are proposed to identify if there is any concept in the alternative formulation, which could be considered for enhancing a solving method considering the traditional SCP formulation. As a result, the authors conclude that there are concepts from the alternative formulation, which could be applied for guiding the search process and for designing heuristic feasibilit\y operators. Thus, such concepts could be recommended for designing state-of-the-art algorithms addressing the traditional SCP formulation.

1. Introduction

The set covering problem (SCP) is a classical problem shown to be NP-complete by Karp [1] and whose optimization version is NP-hard. Although this is a traditional problem, SCP is widely considered in the current scientific literature because it fits problems in relevant areas, such as engineering, vehicle routing, medical domain, and facilities allocation (see e.g., [2–6]).

Most contributions in the SCP field considered the traditional SCP formulation introduced by Chvatal [7] and defined as follows. Let E be a set of m objects and let S be a collection of n subsets of E, where each subset has a nonnegative cost associated. Then, the purpose of SCP is to get a minimum cost family of subsets $X \subseteq S$, such that each element of E belongs to at least one subset of the family X. This traditional formulation does not directly deal with two aspects: solution unsatisfiability and set redundancy. The solution unsatisfiability aspect is related to the possibility of generating unfeasible solutions during the search. The set redundancy aspect is related to the possibility of generating nonoptimized solutions in cost, including redundant components (subsets). The noninclusion of these two aspects in the formulation means that the solving method has to address them to ensure good performance.

In the last years, Bilal et al. [8] proposed an alternative SCP formulation. Its main contribution was that both redundant sets and unfeasible solutions were directly penalized in the fitness function. Therefore, the solving method does not have to control such aspects in contrast to the

traditional SCP formulation. Nevertheless, the contribution of the alternative formulation becomes questionable due to two main issues. First, Vasko et al. [9] demonstrated that the calculation effort to remove redundant components from an SCP solution is almost negligible. Thus, including a redundancy removal operator in a solving method addressing the traditional formulation does not increase excessively the computational cost. Second, there are simple methods for transforming unfeasible solutions into feasible ones, such as the one proposed by Beasley and Chu [10]. As a consequence, alternative formulation seems not to be advantageous.

Analysing the work proposed in the alternative formulation in Bilal et al. [8], they compared the alternative formulation to the traditional one. To this end, they solved the standard Beasley's OR library [11] through two algorithms: a simple descent heuristic (DH) addressing the alternative formulation and a standard greedy heuristic (GH) addressing the traditional one. As a result, DH outperformed GH, which is valuable for justifying the alternative formulation. However, Vasko et al. [9] later applied the same GH using the traditional formulation on the same instances, but included a simple redundancy removal operator, obtaining better results than the ones shown in Bilal et al. [8] for DH. Once again, the alternative formulation seems to have questionable merit. However, this comparison initiated in Bilal et al. [8] might have some limitations:

(i) The heuristic techniques considered might not be the most appropriate according to the current state of the art, in which metaheuristics, especially swarm intelligence algorithms (SIAs), provide the best results in general.

(ii) The authors independently compared the two formulations using different algorithms. This focus could be correct because the algorithm best suited for a formulation could be very different, even in type, from the algorithm for the other formulation. However, there is no study combining aspects from the two formulations that could provide some advantages for the same solving method.

(iii) The authors did not consider any statistical method for comparing both formulations. Instead, they only compare the average solution obtained.

On this basis, this paper questions whether there is any advantage from the concepts involved in the alternative formulation, beyond the novel problem formulation. This idea leads us to propose two studies focused on solution quality as a way to know if there is any concept in the alternative formulation, which could be considered for enhancing a method using the traditional formulation. To this end, the authors select two different metaheuristics adequate for the studies, although other metaheuristics could have been selected without loss of generality.

The only demonstration of the concept utility from the alternative formulation is valuable. This means that future solving methods could include these novel concept. This research focus implies that the authors are not focused on getting the best absolute results solving Beasley's OR library, but they understand that the solutions obtained should be reasonable, as will be discussed later.

The first study focuses on identifying if there is any concept in the alternative formulation, which could be applied for guiding the search process of a solving method addressing the traditional formulation. To this end, the authors generate two versions of the same SIA addressing the traditional formulation. In the first version, (i), the search process of SIA is guided by concepts from the traditional formulation. In the second version, (ii), the search process of SIA is guided by concepts from the alternative formulation. Further details of this first study are as follows:

(i) This study requires a solving method, whose search process is closely linked to the optimization problem. The ant colony optimization (ACO) algorithm meets this requirement, being very sensitive to the heuristic information operator designed based on the problem to solve. Thus, two heuristic information operators are considered: in (i), a usual operator based on the traditional formulation and in (ii), a novel operator based on concepts from the alternative formulation.

(ii) The two ACO approaches in (i) and (ii) include the same usual operator for removing redundant sets. No operator for transforming unfeasible solutions into feasible ones is considered because ACO does not generate them.

(iii) The two ACO approaches in (i) and (ii) are applied for solving Beasley's OR library. The results obtained were analysed through a widely accepted statistical method. Both approaches were tuned through the automatic method iterated F-Race, preventing errors from a manual method [12].

The second study focuses on identifying if there is any concept in the alternative formulation, which could be considered for designing the operators needed for transforming unfeasible solutions into feasible ones while removing redundant columns. Note that this type of operators is widely applied in most SIAs solving SCP. To this end, the authors generate two versions of the same SIA addressing the traditional formulation. In the first version, (iii), SIA considers the widely applied operator proposed in Beasley and Chu [10]. In the second version, (iv), SIA considers a novel operator inspired by concepts from the alternative formulation. Further details of this second study are as follows:

(i) The second study requires a solving method, which could generate unfeasible solutions. The artificial bee colony (ABC) is one of the many metaheuristics meeting this requirement. Thus, two different feasibility operators are considered in (iii) and (iv).

(ii) The two ABC approaches in (iii) and (iv) are applied for solving Beasley's OR library. The results obtained were analysed through a widely accepted statistical method. In this case, the parameter configuration is

taken from the literature (see [13]) because the feasibility operator is considered as an external tool.

To summarize, the motivation of this research is to identify if there is any concept from the alternative formulation, which could be used to solve the traditional SCP problem. To the best of our knowledge, this is the first work performing this research. Figure 1 summarizes the main tasks performed in the two studies. The contributions to the field are as follows:

(i) A first study is proposed to identify concepts of interest in the alternative SCP formulation, which could be applied for guiding a search method addressing the traditional SCP formulation. The study results in that the gain concept in the alternative SCP formulation is useful for guiding the search, outperforming the results obtained by a usual heuristic information operator from the literature.

(ii) A second study is proposed to identify concepts of interest in the alternative SCP formulation, which could be considered for designing feasibility operators. The study results in that the gain concept in the alternative SCP formulation is useful for designing this type of operators, outperforming a usual operator in the literature. This contribution is especially interesting because this type of operators is widely applied in the metaheuristic SCP field as a black-box method.

These two contributions are especially interesting for future works, implementing techniques for solving the traditional SCP formulation. From the first study, it is shown that the gain concept in the alternative SCP formulation is useful for guiding the search. That means that this concept could be considered during the design of novel solving methods for SCP. From the second study, it is shown that the gain concept is useful for designing feasibility operators. That means that this concept could be considered to improve techniques already shown to be useful for solving the problem, as well as proposing novel feasibility operators. As stated before, the second future scope line is especially interesting because feasibility operators are widely applied in the literature as black-box techniques outside the solving method.

The rest of this paper is structured as follows. Section 2 discusses related work. In Section 3, a formal statement of both SCP formulations is provided. In Section 4, the main aspects of the ACO algorithm in the first study are discussed. Section 5 discusses the main aspects of the ABC algorithm in the second study. In Section 6, the experimental methodology followed is discussed and the solution quality results are analysed. Finally, Section 7 concludes and introduces future works. Table 1 includes a summary of the notation considered throughout this work.

2. Related Work

The literature about SCP is extensive. Some authors considered exact algorithms, such as branch-and-bound and branch-and-cut techniques [14–16] or linear programming [17–19]. More recently, Caprara et al. [20] compared several exact algorithms, concluding that the best exact technique was CPLEX.

It is well known that exact techniques require excessive computer resources on large problems. Therefore, much effort was focused on exploring heuristic and metaheuristic algorithms, which could find near-optimal (or even optimal) solutions for large problems in reasonable computing time.

Starting from heuristic methods, Chvatal [7] applied a classical GH. Although GHs are simple and fast to implement, they seldom produce good quality solutions. Some researchers tried to improve GHs by adding randomness (see e.g., [9, 21–24]). Highly sophisticated heuristics based on Lagrangian relaxation were also considered, yielding very good solutions (see e.g., [20, 25–27]). From this brief review, it is shown that the number of proposals considering heuristic methods is limited for SCP in the last years. Note that in other optimization problems, the proposal of heuristics is usual (e.g., [28]). This situation is opposite for metaheuristics, acquiring a great relevance during the last decades [29]. Thus, metaheuristics were applied to fields as networking [30], biological ontology alignment [31], shop scheduling [32], chemical analysis [33], and image encryption [34].

Metaheuristics combine effectiveness exploring the search space and basic heuristic methods. Such techniques are usually split into three large groups: evolutionary algorithms (EAs), trajectory algorithms (TAs), and SIAs. To solve SCP, some authors considered EAs (see e.g., [10, 35–37]). Other authors applied TAs (see e.g., [38, 39]). However, the most widely applied metaheuristics for solving SCP are SIAs. Some examples are the artificial bee colony (ABC) algorithm [13, 40, 41], the ant colony optimization (ACO) algorithm [42–44], the firefly algorithm (FA) [45, 46], the teaching-learning-based optimization (TLBO) algorithm [47, 48], the electromagnetism-like (EM-like) algorithm [49, 50], the shuffled frog leaping algorithm (SFLA) [51], the fruit fly optimization algorithm (FFOA) [52], the cuckoo search algorithm (CSA) [53, 54], the cat swarm optimization (CSO) algorithm [55, 56], the jumping particle swarm optimization (JPSO) method [57], the black hole optimization [54, 58], and the monkey search algorithm [59].

Analysing the previous contributions according to the results obtained, exact algorithms provided excellent results, solving reduced SCP problems. Focusing on larger SCP problems addressed by approximate techniques (heuristics and metaheuristics), the authors check that heuristics do not provide as good results as the more sophisticated metaheuristics. Thus, the best results were usually obtained by SIAs. In this line, we should mention the valuable contributions of Naji-Azimi et al. [48, 49] and Balaji and Revathi [57] who got optimal or near-optimal solutions for classical SCP benchmarks.

All the works listed before have in common that they considered the traditional SCP formulation. On the contrary, the alternative formulation received limited attention. As far as the authors know, there are only two works

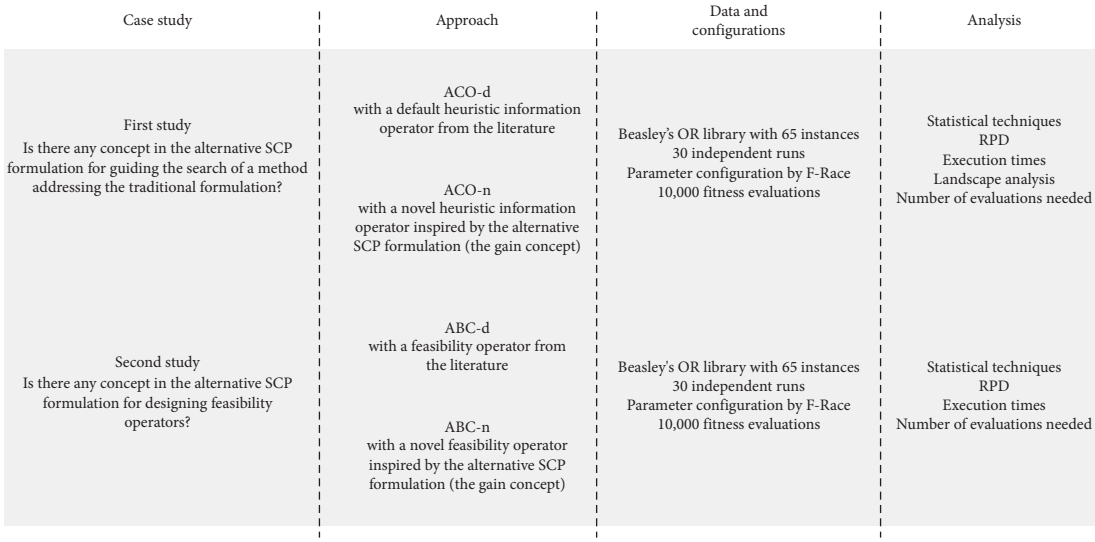

Figure 1: Summary of the main tasks performed in the two studies in this paper.

Table 1: Notation.

$\|\cdot\|$	Cardinal of a given set.
α	Relative importance of pheromone trails, $\alpha \geq 0$.
α_i	The set of columns that row $i \in I$ covers, $\alpha_i \subseteq J$.
A	Binary matrix of m-rows and n-columns. The rows are the elements E of the universe and the columns are the subsets S.
abc_{add}	Percentage of columns to be added during the generation of a solution in ABC.
abc_{eli}	Percentage of columns to be removed during the generation of a solution in ABC.
$a\eta_j^t$	Heuristic factor of column $j \in R_{t-1}$ at step $t \geq 0$ for the alternative formulation, $\eta(a)_j^t > 0$.
$a_{i,j}$	Value in the cell (i, j) of A. It equals 1 if the j-th column covers the i-th row and 0 otherwise, $i \in I$, $j \in J$.
argmax{·}	Point/points in which a function gets its maximum value/values.
argmin{·}	Point/points in which a function gets its minimum value/values.
$a\phi_j^t$	The sum of the gains of covering the noncovered rows which could be covered by column $j \in R_{t-1}$ at $t \geq 0$.
β	Relative importance of heuristic information, $\beta \geq 0$.
C	Set of costs, $C = \{c_1, c_2, \ldots, c_n\}$.
$c\eta_j^t$	Heuristic factor of column $j \in R_{t-1}$ at step $t \geq 0$ for the classical formulation, $\eta(c)_j^t > 0$.
c_j	Cost associated to the j-th column, $c_j \in \mathbb{R}^+$, $j \in J$.
$c_{\min}(e_i)$	Cost of the cheapest set among the sets covering e_i, $c_{\min}(e_i) \in C$, $i \in I$.
$c\phi_j^t$	Number of noncovered rows which could be covered by column $j \in R_{t-1}$ at step $t \geq 0$.
Δ_t	Solution generated by an ant at step $t \geq 0$, $\Delta_t \subseteq J$.
Δ	A solution to the problem, $\Delta \subseteq J$.
ϵ	Ratio coefficient, $\epsilon \in (0, 1)$.
η_j^t	Heuristic factor of column $j \in R_{t-1}$ at step $t \geq 0$, $\eta_j^t > 0$.
E	Universe of elements, $E = \{e_1, e_2, \ldots, e_m\}$.
e_i	i-th element of E, $e_i \in E$, $i \in I$.
evals	Average fitness evaluations needed for reaching the best solution found during the exploration.
f_1	Fitness function of the classical SCP formulation, $f_1 \in \mathbb{R}^+$.
f_1'	Fitness function of the alternative SCP formulation, $f_1' \in \mathbb{R}$.
g_i	Gain of covering an element e_i, $g_i \in \mathbb{R}^+$, $i \in I$.
I	Row set, $I = \{1, 2, \ldots, m\}$.
I_j	Row set covered by column $j \in J$, $I_j \subseteq I$.
ipv	Percentage of improvement by considering the alternative approach of the algorithm instead of the default version.
J	Column set, $J = \{1, 2, \ldots, n\}$.
L	The best solution found from the beginning of the algorithm, $L \subseteq J$.
limit	Threshold value based on trials for deciding if a worker bee is transformed into a scout one in ABC, limit > 0.
m	Cardinal of E or number of rows.
n	Cardinal of S or number of columns.
n_w	Number of worker bees in ABC, $n_w \leq \text{ps}_{abc}$.
ω_j	Amount of pheromone put on column $j \in R_{t-1}$, $\omega_j \geq 0$.
ψ	Very small positive constant, $\psi \in \mathbb{R}^+$.
ps_{abc}	Population size in ACO.

Table 1: Continued.

ps_{aco}	Population size in ABC.
QMetric	Landscape solution quality metric, QMetric $\in [0,1]$.
q	Random number uniformly distributed in $[0,1]$.
q_0	Parameter determining the relative importance of exploitation versus exploration, $q_0 \in [0,1]$.
ρ	Pheromone persistence, $\rho \in [0,1)$.
$\underline{R_t}$	Set of unselected columns in Δ_t, $\underline{R_t} \subseteq J$, $t \geq 0$.
$\overline{\text{rpd}}$	Average RPD from a distribution, $\overline{\text{rpd}} \in [0,1]$.
rpd_a	Average RPD of algorithm a, with $a \in \{\text{d} - \text{ACO}, \text{n} - \text{ACO}\}$.
rpd_b	Average RPD of algorithm a, with $b \in \{\text{d} - \text{ACO}, \text{n} - \text{ACO}\}$.
rpd_c	Average RPD of algorithm a, with $c \in \{\text{d} - \text{ABC}, \text{n} - \text{ABC}\}$.
rpd_d	Average RPD of algorithm a, with $c \in \{\text{d} - \text{ABC}, \text{n} - \text{ABC}\}$.
$\text{rpd}_{\min}$	Minimum RPD from a distribution, $\text{rpd}_{\min} \in [0,1]$.
$\text{rpd}_{\max}$	Maximum RPD from a distribution, $\text{rpd}_{\max} \in [0,1]$.
S	Set of subsets, $S = \{s_1, s_2, \ldots, s_n\}$, $\cup S = E$.
s_j	j-th subset of S, $s_j \in S$, $s_j \subseteq E$, $j \in J$.
SRate	Landscape rate of success, SRate $\in [0,1]$.
SSpeed	Landscape speed of reaching a solution, SSpeed $\in [0,1]$.
τ_j	Pheromone trail of column $j \in R_{t-1}$, $\tau_j > 0$.
t	Construction step in ACO, $t \geq 1$.
$\overline{\text{time(s)}}_a$	Average execution time of algorithm a, with $a \in \{\text{d} - \text{ACO}, \text{n} - \text{ACO}\}$.
$\overline{\text{time(s)}}_b$	Average execution time of algorithm b, with $a \in \{\text{d} - \text{ACO}, \text{n} - \text{ACO}\}$.
$\overline{\text{time(s)}}_c$	Average execution time of algorithm c, with $c \in \{\text{d} - \text{ABC}, \text{n} - \text{ABC}\}$.
$\overline{\text{time(s)}}_d$	Average execution time of algorithm d, with $d \in \{\text{d} - \text{ABC}, \text{n} - \text{ABC}\}$.
V	Number of uncovered rows in a solution, $V \leq m$.
V_t	Set of uncovered rows in Δ_t, $V_t \subseteq I$, $t \geq 0$.
ξ_i	Number of columns in a solution Δ that cover the row $i \in I$, $\xi_i \leq n$.
X	An SCP solution expressed as binary vector, $X = \{x_1, x_2, \ldots, x_n\}$.
x_j	j-th element of X. It equals 1 if s_j is part of the solution X and 0 otherwise, $j \in J$.
y_i	Variable equaling 1 if the element e_i is covered in X and 0 otherwise, $i \in I$.
ζ_t	Column provided by SROM at step $t \geq 0$, $\zeta_t \in R_{t-1}$.
$\overline{z}$	Average cost obtained from a distribution, $\overline{z} \in \mathbb{R}^+$.
$z_{\max}$	Maximum cost obtained from a distribution, $z_{\max} \in \mathbb{R}^+$.
$z_{\min}$	Minimum cost obtained from a distribution, $z_{\min} \in \mathbb{R}^+$.
z_{opt}	Optimum solution of a given instance, $z_{\text{opt}} \in \mathbb{R}^+$.
z_t	Column selected at step $t \geq 0$, $z_t \in R_{t-1}$.

considering the alternative formulation. In Bilal et al. [60], they solved an SCP variant through an iterated tabu-search metaheuristic. In Crawford et al. [61], they compared the results obtained solving the traditional and alternative formulations through the ACO algorithm.

The research presented in this paper was inspired by a very preliminary work discussed before (see Crawford et al. [61]). In this contribution, there is no study regarding the existence of concepts in the alternative formulation, which could be considered for solving methods addressing the traditional formulation. In Lanza-Gutierrez et al. [56], the authors applied an SIA to solve SCP by a CSO algorithm but with a completely different approach.

3. Set Covering Problem Statements

Let $I = \{1, 2, \ldots, m\}$ and $J = \{1, 2, \ldots, n\}$ be the row and column sets, respectively. Let $E = \{e_1, e_2, \ldots, e_m\}$ be a universe of m elements and let $S = \{s_1, s_2, \ldots, s_n\}$ be a collection of n subsets of E, such that $s_j \subseteq E$ and $\cup S = E$, with $j \in J$. Each subset s_j has a non-negative cost associated $c_j \in C$, where $C = \{c_1, c_2, \ldots, c_n\}$.

The optimization problem is formally defined by assuming a binary matrix A of m-rows and n-columns, where the rows are the elements of the universe and the columns are the subsets. Let a_{ij} be the value in the cell (i, j) of A given by

$$a_{ij} = \begin{cases} 1, & \text{if } e_i \in s_j, \\ 0, & \text{otherwise,} \end{cases} \tag{1}$$

for $i \in I$ and $j \in J$, where $e_i \in E$. Thus,

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix}. \tag{2}$$

The objective of SCP is to find a subset of S covering (containing) all the elements of E at a minimal cost. A solution to SCP is usually expressed as a binary vector $X = \{x_1, x_2, \ldots, x_n\}$, where

$$x_j = \begin{cases} 1, & \text{if the set } s_j \text{ is part of the solution,} \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Then, the cost of the solution X is

$$\sum_{j \in J} c_j x_j. \tag{4}$$

Next, we give a formal statement of the two SCP formulations.

3.1. Traditional Formulation. The SCP fitness function is

$$f_1 = \sum_{j \in J} c_j x_j, \quad f_1 \in \mathbb{R}^+. \tag{5}$$

Then, given m elements and n subsets, the objective is to find a collection of subsets to

$$\min(f_1), \tag{6}$$

subject to

$$\sum_{j \in J} a_{ij} x_j \geq 1, \quad \forall i \in I, \tag{7}$$

$$x_j \in \{0, 1\}, \quad \forall j \in J. \tag{8}$$

The constraint in equation (7) ensures that each row is covered by at least one column. If this constraint is not satisfied, the solution is considered unfeasible. The constraint in equation (8) is only for the integrity of the mathematical programming. Hence, this equation does not need to be addressed as a constraint in heuristic approaches.

3.2. Alternative Formulation. In this formulation, covering an element is identified with collecting a gain at a given cost. Let $c_{\min}(e_i) \in C$ be the cost of the cheapest set among the sets covering the element e_i given by

$$c_{\min}(e_i) = \arg\min_{c_j \in C} \{a_{ij} x_j c_j > 0\}, \quad \forall j \in J, \tag{9}$$

where argmin$\{\cdot\}$ provides the point/points in which a function gets its minimum value/values. Then, the gain $g_i \in \mathbb{R}^+$ of covering an element e_i is

$$g_i = c_{\min}(e_i) + \psi, \tag{10}$$

where $\psi \in \mathbb{R}^+$ is a very small positive constant. Based on this gain concept, the SCP fitness function is

$$f_1' = \sum_{i \in I} g_i y_i - \sum_{j \in J} c_j x_j, \quad f_1' \in \mathbb{R}, \tag{11}$$

where

$$y_i = \begin{cases} 1, & \text{if } e_i \text{ is covered in the solution } X, \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

Then, given m elements and n subsets, the objective is to find a collection of subsets to

$$\max(f_1'), \tag{13}$$

subject to

$$y_i \leq \sum_{j \in J} a_{i,j} x_j, \quad \forall i \in I, \tag{14}$$

$$x_j, y_i \in \{0, 1\}, \quad \forall i \in I, \forall j \in J. \tag{15}$$

The constraints in equations (14) and (15) are only for the integrity of the mathematical programming. According to this formulation, there are no unfeasible solutions as happens with the traditional formulation. Note that unfeasible solutions still exist for the problem. However, the alternative formulation penalizes such issue instead of discarding the solution. Moreover, it also penalizes directly redundant sets beyond having a higher cost as occurs for the traditional formulation. Thus, the use of redundancy removal operators is not needed, in contrast to the traditional formulation, where it is highly recommended.

4. Ant Colony Optimization

The ACO algorithm is inspired by ant colony behaviours. The ACO process is focused on the search of the optimal path in a graph based on an artificial ant colony. Thus, ants work cooperatively and communicate through heuristic information depending on the problem and pheromone trails. Pheromone trails are a type of distributed information, which is dynamically updated by the ants. Pheromones keep the experience gained during the search process while remarking promising areas of the search space.

Let $\Delta_{t-1} \subseteq J$ be the solution generated by an ant at construction step $t - 1 \geq 0$. Let $V_{t-1} \subseteq I$ be the set of uncovered rows in Δ_{t-1}. Let $R_{t-1} \subseteq J$ be the set of unselected columns in Δ_{t-1}. Reviewing the scientific literature [42, 62], a usual heuristic information expression for a column $j \in R_{t-1}$ at step t is

$$c\eta_j^t = \frac{c\phi_j^t}{c_j}, \tag{16}$$

where $c\phi_j^t$ is the number of noncovered rows in Δ_{t-1}, which could be covered by column j at step t. This value is

$$c\phi_j^t = \left\| I_j \cap V_{t-1} \right\|, \tag{17}$$

where $\|\cdot\|$ is the cardinal of a set and I_j denotes the row set covered by column j, for $j \in J$ and $I_j \subseteq I$.

In this work, we propose a heuristic information inspired by the gain concept from the alternative formulation introduced in Section 3.2 as

$$a\eta_j^t = \frac{a\phi_j^t}{c_j}, \tag{18}$$

where $a\phi_j^t$ is the sum of the gains of covering the noncovered rows in Δ_{t-1} by column j at step t. Thus,

$$a\phi_j^t = \sum_{i \in \{I_j \cap V_{t-1}\}} g_i, \tag{19}$$

where g_i is given in equation (10).

To simplify the notation, we define the heuristic information for a column j at step t based on whether we consider the traditional formulation or the alternative one. That is,

$$\eta_j^t = \begin{cases} c\eta_j^t, & \text{if traditional formulation,} \\ a\eta_j^t, & \text{if alternative formulation.} \end{cases} \tag{20}$$

Algorithm 1 shows the procedure of a general ACO. Next, the main steps are detailed.

(1) *Initialization*: in the beginning, we propose to preprocess the SCP instances by using column domination and column inclusion [18]. Next, the algorithm parameters are initialized. Traditionally, ACO algorithms do not include an initialization step to generate the ps_{aco} solutions in the population. Instead, pheromone trails are randomly assigned and then solutions are generated according to this random information. That means that the algorithm could need to run some iterations before having the right information about the solution component quality. At this point, we propose to include a greedy population initialization step in ACO based on Lu and Vasko [48]. This step corresponds to line 1 of Algorithm 1.

(2) *Solution construction method*: each ant starts with an empty solution where columns are added iteratively until all rows are covered. Consequently, this strategy causes all solutions generated to be feasible. Most ACO-based algorithms consider a similar state transition rule, preferring solution components with high pheromone and heuristic values (see e.g., [42, 62]. A possible way to generate solutions is the single row oriented method (SROM) proposed by Ren et al. [43]. In that work, it was demonstrated that SROM reduces the computation burden compared to other methods. Thus, SROM is used in this paper as the solution construction method. Additionally, we also consider the ant colony system (ACS) proposed by Dorigo and Gambardella [63] as an extension of the ACO algorithm. ACS includes a pseudo-random-proportional rule, providing a direct way to balance between exploration and exploitation during the selection of the solution component. If $z_t \in R_{t-1}$ denotes the column selected at step t, then the ACS rule is

$$z_t = \begin{cases} \arg\max_{j \in R_{t-1}} \left\{ \left(\tau_j\right)^{\alpha} \left(\eta_j^t\right)^{\beta} \right\}, & \text{if } q \le q_o \text{ (exploitation)}, \\ \zeta_t, & \text{otherwise (exploration)}, \end{cases} \tag{21}$$

where q is a random number uniformly distributed in $[0, 1]$, $q_0 \in [0, 1]$ is a parameter determining the relative importance of exploitation versus exploration, $\zeta_t \in R_{t-1}$ is the column provided by SROM at step t, and $\arg\max\{\cdot\}$ provides the point/points in which a function gets its maximum value/values. Thus, if $q \le q_0$, then it returns the nonselected column having the highest value of $(\tau_j)^{\alpha}\ (\eta_j^t)^{\beta}$ at step t, where $\tau_j > 0$ denotes the pheromone trail of column j and $\alpha \ge 0$ and $\beta \ge 0$ denote the relative importance of pheromone trails and heuristic information, respectively. This step corresponds to line 4 of Algorithm 1.

(3) *Local search*: it is well known that local search is effective to improve ACO performance. We consider the local search proposed by Ren et al. [43], where for each column in Δ_t, the algorithm determines if the column should be removed or replaced by one or more columns while keeping solution feasibility. This step corresponds to line 5 of Algorithm 1.

(4) *Update pheromone trails*: we consider that pheromone trails are updated based on the max-min ant system (MMAS) approach proposed by Stützle and Hoos [64]. In this method, after each ant generates a full solution, all pheromone trails are decreased uniformly to simulate evaporation, forgetting part of the historical experience. Next, a small amount of pheromone is deposited on the columns corresponding to the best solution found. To this end, MMAS considers the best solution found in the current iteration, instead of the best solution found from the beginning of the algorithm. We opted for the second option as did Ren et al. [43]. Thus, the search can concentrate fast around the best solution found. This strategy could result in a bad performance if the algorithm is trapped in bad solution areas. However, this risk is reduced due to the ACS strategy detailed in Step 2. Formally, pheromone trails are updated following

$$\tau_j = \rho\, \tau_j + \omega_j, \quad \forall j \in J, \tag{22}$$

where $\rho \in [0, 1)$ is the pheromone persistence and $\omega_j \ge 0$ is the amount of pheromone put on column j provided in Stützle and Hoos [64]. Additionally, they also proposed that the range of pheromone trails is in $[1/[(1-\rho)(\sum_{j \in L} c_j)], \epsilon/[(1-\rho)(\sum_{j \in L} c_j)]]$, where $\epsilon \in (0, 1)$ denotes a ratio coefficient and L is the best solution found from the beginning of the algorithm, $L \subseteq J$. This step corresponds to line 7 of Algorithm 1.

5. Artificial Bee Colony

The ABC algorithm is inspired by honey bee behaviours, the search process being guided by three types of artificial bees: workers, onlookers, and scouts. The general procedure of ABC is shown in Algorithm 2. It starts by generating an initial population of ps_{abc} solutions. For every row, a random column with covering possibilities is selected until all rows are covered. Next, along iterations, the population is managed by $n_w - ps_{abc}$ workers and $1 - n_w$ onlookers, which are randomly recruited in each iteration. The behaviour of each bee is as follows:

(i) A worker takes a random solution from the population to generate a new solution by adding a random number of columns between 0 and abc_{add} (in percentage) of columns in the SCP instance. This step is followed by an elimination of random columns between 0 and abc_{eli} (in percentage) of columns in the SCP instance. The fitness value of the individual generated by the worker is obtained. In the case that the fitness value of the new individual is better than the previous individual assigned to the worker, then the new individual replaces the previous one. In the opposite case, the counter is increased for the number of trials for improving the current solution. Otherwise, the counter is set to zero. If such counter reaches the limit threshold, the worker is transformed into a scout bee.

```
Initialize
while not stop condition do
    for each ant in the nest do
        generate a new solution
        apply local search to the new solution
    end for
    update pheromone trails
end while
return the best solution found
```

Algorithm 1: The ACO algorithm.

```
Generate ps_abc initial solutions
while not stop condition do
    Recruit n_w − ps_abc worker bees
    Recruit 1 − n_w onlooker bees
    recruit scout bees if needed
    update the best solution found
end while
return the best solution found
```

Algorithm 2: ABC algorithm.

```
initialize ξ_i = ‖Δ ∩ α_i‖, ∀i ∈ I
initialize V = {i | ξ_i = 0, ∀i ∈ I}
for each row i ∈ V in increasing order of i do
    find the first column j in α_i minimizing equation (23)-default or (24)-proposal
    add j to Δ
    set ξ_i = ξ_i + 1, ∀i ∈ I_j, and V = V − I_j
end for
for each column j ∈ Δ in decreasing order of j do
    if ξ_i ≥ 2, ∀i ∈ I_j then
        set Δ = Δ − j and ξ_i = ξ_i − 1, ∀i ∈ I_j
    end if
    end for
return Δ, which is a feasible SCP solution, without redundant columns
```

Algorithm 3: Heuristic operator for solution feasibility in ABC.

(ii) An onlooker generates a new solution following a similar procedure as for workers, but selecting the solution with probability to its quality, instead of randomly. The concept of the limit threshold is not used in onlookers.

(iii) A scout discards its current solution and generates a new one by following the same strategy as for generating the initial population. As expected, the counter of trials is initialized to zero.

As both workers and onlookers can generate unfeasible solutions because of the random elimination of columns, it is mandatory to manage this issue. Crawford et al. [13] proposed to consider the usual heuristic by Beasley and Chu [10] for transforming unfeasible solutions into feasible ones while reducing the cost of the solution in a later step. This heuristic is shown in Algorithm 3. Here, the first stage in lines 3 – 7 transforms an unfeasible solution into a feasible one. The second stage in lines 8 – 12 removes redundant columns. In this algorithm, note that Δ, α_i, and ξ_i are the set of columns in a solution, the set of columns that row $i \in I$ covers, and the number of columns in Δ that cover the row i, respectively.

Focusing on the first stage, the steps required to make a solution feasible include the identification of uncovered rows and the addition of columns to the solution so that all rows are covered. The search for the missing columns in the proposal of Beasley and Chu [10] is guided by

$$\min \frac{c_j}{\left\| I_j \cap V \right\|}, \tag{23}$$

where V is the number of noncovered rows in the solution, i.e., the ratio between the cost of a column and the number of noncovered rows, which could be covered by such column.

As an alternative strategy, this paper proposes to guide the search based on the concepts from the alternative SCP formulation, that is,

$$\min \frac{c_j}{\sum_{i \in \{I_j \cup V\}} g_i}, \tag{24}$$

i.e., the ratio between the cost of a column and the sum of the gains of covering the noncovered rows by such column.

6. Experimentation

This section discusses the experimental methodology and analyses the results obtained in the first and second studies.

6.1. Experimental Methodology. We apply the two approaches in each study for ACO and ABC algorithms to solve Beasley's OR library. This dataset is widely used to report empirical results in the current literature (see e.g. [9, 40, 48]). This library includes 65 non-unicost instances generated randomly, as detailed in Table 2. For further details about the random generation of these instances, see [18, 65]. For each instance in the library, the number of rows, the number of columns, and the cost of each column are provided. Additionaly, for each row, the number of columns that covers and also the list of columns which cover that row are also provided. For a complexity study of the search space in this benchmark, we refer readers to the work by Finger et al. [66], which considered the fitness-distance correlation landscape metric to this end. In Table 2, "*Density* (%)" contains the percentage of $1'$s in the A matrix in equation (2). "*Optimal solution*" shows two possible values, known and unknown, according to whether the instances have a solution tested to be optimal, or instead it could not be checked because of problem complexity. Thus, we only know the best historical solutions found for the sets nrg and nrh.

We combine two stop condition criteria for performing the experimentation: reaching a given number of fitness evaluations or getting the optimal solution. If at least a condition holds, the algorithm ends. For ACO, we assume 10,000 fitness evaluations as a stop condition. As we will discuss later, this value is enough for performing the experimentation. For ABC, we consider 500 iterations based on Crawford et al. [13].

Before running the experimentation, we should configure both algorithms. In the case of ABC, we can assume the parameters provided in Crawford et al. [13] for the two approaches of ABC considered here because (i) the authors also solved SCP and (ii) the approach based on the alternative formulation only modifies the heuristic operator for solution feasibility and the operators guiding the search are not modified. In the case of ACO, we should configure the two approaches of ACO considered here because (i) we do not have any set of parameters from previous works for the approach of ACO used and (ii) the approach based on the alternative formulation modifies how the search is performed in comparison to the traditional one, and then we should configure the two approaches independently.

Thus, for the first study, we consider $\text{ps}_{abc} = 200$, $n_w = 100$, limit = 50, $abc_{\text{add}} = 0.5\%$, and $abc_{\text{eli}} = 1.2\%$ of Crawford et al. [13]. For the second study, we get the parameters of the two ACO approaches using F-Race. This method configures a metaheuristic starting with a set of candidate values for each parameter. Then, it discards bad performance configurations as soon as statistically sufficient evidence is reached against them, focusing on the most promising ones.

Concretely, we consider the iterated F-Race implementation for R software by López-Ibáñez et al. [67]. Following the authors' recommendations, we divided the benchmark into three groups according to the problem size ($m \times n$) to get a consistent configuration. Thus, Group A includes instance sets 4, 5, and 6; Group B includes instance sets a, b, c, and d; and, finally, Group C includes instance sets nre, nrf, nrg, and nrh. Table 3 shows the candidate values for each parameter based on previous works [43] and the configurations obtained for each group and ACO approach. Note that d-ACO denotes ACO with the traditional heuristic information expression and n-ACO denotes ACO with the alternative heuristic information expression.

Once both ACO approaches are configured, 30 independent runs are performed for each instance and algorithm. Next, we analyse if there are significant differences between the behaviour of the two algorithms regarding solution quality and execution time for each instance. To this end, the authors consider the Wilcoxon–Mann–Whitney test [68] to validate several hypotheses. The implementation of this test is the one provided in the assessment performance tool described in Knowles et al. [69] and available in Fonseca et al. [70].

However, indirectly the RPD metrics used for the assessment evaluation consider the optimal solution for the instances, which can be considered as the solutions provided by the corresponding exact techniques. Thus, the RPD metric evaluates how far the solution found by the metaheuristic is from the optimal solution provided by an exact technique.

As a solution quality metric, we consider the relative percentage deviation (RPD), which evaluates how far the solution found by the metaheuristic is from the optimal solution known in the literature. The lower the RPD value, the better the solution obtained. Thus, indirectly, this metric evaluates the performance of the technique in comparison with the corresponding generic exact technique solving the same problem instance. Three RPD metrics are included: the average RPD, $\overline{\text{rpd}} \in [0, 1]$; the minimum RPD, $\text{rpd}_{\text{min}} \in [0, 1]$; and the maximum RPD, $\text{rpd}_{\text{max}} \in [0, 1]$. They are calculated as

Table 2: Beasley's OR library description.

Instance set	Number of instances	m	n	Cost range	Density (%)	Optimal solution
4	10	200	1,000	[1, 100]	2.00	Known
5	10	200	2,000	[1, 100]	2.00	Known
6	5	200	1,000	[1, 100]	5.00	Known
a	5	300	3,000	[1, 100]	2.00	Known
b	5	300	3,000	[1, 100]	5.00	Known
c	5	400	4,000	[1, 100]	2.00	Known
d	5	400	4,000	[1, 100]	5.00	Known
nre	5	500	5,000	[1, 100]	10.00	Known
nrf	5	500	5,000	[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 3: F-Race settings and configurations obtained for each ACO approach.

Parameter	Candidate values	d-ACO			n-ACO		
		A	B	C	A	B	C
α	0.25, 0.50, 1.00, 2.00, 5.00, 8.00	1.00	1.00	1.00	1.00	1.00	1.00
β	1.00, 2.00, 5.00, 8.00, 10.00, 13.00	8.00	8.00	5.00	8.00	8.00	8.00
ρ	0.80, 0.85, 0.90, 0.95, 0.98, 0.99, 0.995, 0.999	0.95	0.90	0.90	0.98	0.99	0.90
q_0	0.10, 0.25, 0.50, 0.75, 0.90, 0.95, 0.99	0.50	0.75	0.50	0.90	0.90	0.25
ϵ	0.0001, 0.0005, 0.001, 0.002, 0.005, 0.010	0.001	0.005	0.005	0.005	0.005	0.001
ps_{aco}	5, 10, 20, 50, 100, 150, 200, 250	20	10	100	150	150	50

$$\overline{\text{rpd}} = \left(\frac{\overline{z} - z_{\text{opt}}}{z_{\text{opt}}}\right), \tag{25}$$

$$\text{rpd}_{\text{min}} = \left(\frac{z_{\text{min}} - z_{\text{opt}}}{z_{\text{opt}}}\right), \tag{26}$$

$$\text{rpd}_{\text{max}} = \left(\frac{z_{\text{max}} - z_{\text{opt}}}{z_{\text{opt}}}\right), \tag{27}$$

where $\overline{z}$, z_{min}, and z_{max} denote the average solution cost, the minimum solution cost, the maximum solution cost from a distribution of 30 samples solving a instance, respectively, and z_{opt} is the optimum solution cost of the instance. Note that the cost of the best solution found during one run is given by equation (4). Thus, although both SCP fitness formulations are different, we can compare the results obtained without loss of generality.

Regarding the computing platform considered to perform the experimentation, the authors used two computing nodes in a computing cluster. Each node has two 2.33 GHz Intel Xeon E5410 with four cores each and a 1600 MHz DDR3 16 GB RAM, running a Linux operating system. All executions were performed in a single core without parallelism because the goal of this paper is not to explore parallelism. The reason for considering such unconventional infrastructure is the possibility of performing many independent executions because of the needs for the statistical test required to validate the proposal. That means that for a single execution or a reduced set of them, a conventional computer could be considered. To avoid the operating system tasks affecting the total computing time obtained during the experimentation, one core in each computing node was idle. Additionally, the authors also checked that the RAM in the computing node was enough to not apply memory swap. As expected, the computing power capacity of the processor definitively affects the time required to find the solution to the problem, most of the operations being related to CPU computing and accessing the principal memory (RAM). Note the same computing nodes are considered for all the experiments in this work to not bias the conclusions reached regarding computing time.

Regarding programming languages, ACO algorithm was fully implemented in Java for Java Development Kit (JDK) 1.7. ABC algorithm was fully implemented in C. The scripts for managing the executions and collecting the results were implemented in bash. Note that the usage of two different programming languages for implementing ABC and ACO does not affect the conclusions reached in computing time. This fact is because ABC and ACO computing times are not compared in this work.

6.2. Analysis of the Experimental Results. Tables 4 and 5 show for each instance and case study, the RPD metrics (rpd, rpd_{min}, rpd_{max}), average execution time reaching the stop condition (time (s)), and average fitness evaluations needed for reaching the best solution found during the exploration (evals). In both tables, lower $\overline{\text{rpd}}$ and time (s) values are given in bold for each instance. In Table 5, d-ABC denotes ABC with the default heuristic feasibility operator and n-ABC denotes ABC with the heuristic feasibility operator based on the alternative SCP formulation.

Analysing both tables regarding RPD metrics, we check that (i) n-ACO seems to outperform or match d-ACO in most instances and (ii) n-ABC seems to outperform or match d-ABC in most instances. Focusing on computing

Table 4: RPD metrics, the average total execution time, and average number of evaluations obtained during the first study for ACO algorithm.

Instance	z_{opt}	d-ACO					n-ACO				
		$\overline{\text{evals}}$	$\overline{\text{time (s)}}$	$\overline{\text{rpd}}$ (%)	rpd_{min} (%)	rpd_{max} (%)	$\overline{\text{evals}}$	$\overline{\text{time (s)}}$	$\overline{\text{rpd}}$ (%)	rpd_{min} (%)	rpd_{max} (%)
4_1	429	1880	845	0.23	0.23	0.23	1350	**109**	**0.00**	**0.00**	**0.00**
4_2	512	3950	900	0.20	**0.00**	0.59	2700	**224**	**0.00**	**0.00**	**0.00**
4_3	516	1600	916	0.97	**0.19**	1.16	4200	**704**	**0.19**	**0.00**	**0.78**
4_4	494	4850	880	0.40	**0.20**	0.81	2100	**829**	**0.20**	**0.20**	**0.20**
4_5	512	1950	680	0.10	**0.00**	**0.39**	600	**340**	**0.00**	**0.00**	**0.39**
4_6	560	3780	574	**0.00**	**0.00**	0.36	2625	**240**	**0.00**	**0.00**	**0.00**
4_7	430	1430	858	0.70	**0.00**	1.16	1425	**118**	**0.00**	**0.00**	**0.00**
4_8	492	2440	415	**0.00**	**0.00**	0.20	1350	**97**	**0.00**	**0.00**	**0.00**
4_9	641	5120	932	1.25	0.31	2.34	5550	**848**	**0.31**	**0.00**	**0.78**
4_10	514	1390	115	**0.00**	**0.00**	**0.00**	750	**60**	**0.00**	**0.00**	**0.00**
5_1	253	4860	891	0.40	**0.00**	1.58	3000	**526**	**0.00**	**0.00**	**1.19**
5_2	302	4970	945	1.49	0.66	2.32	2700	**870**	**1.32**	**0.33**	**1.99**
5_3	226	2080	753	0.22	**0.00**	1.33	1725	**125**	**0.00**	**0.00**	**0.00**
5_4	242	2780	245	**0.00**	**0.00**	**0.00**	1500	**110**	**0.00**	**0.00**	**0.00**
5_5	211	1160	857	0.47	0.47	**0.47**	750	**223**	**0.00**	**0.00**	**0.47**
5_6	213	640	59	**0.00**	**0.00**	**0.00**	300	**25**	**0.00**	**0.00**	**0.00**
5_7	293	5490	901	0.51	**0.34**	1.02	1575	**837**	**0.34**	**0.34**	**0.34**
5_8	288	3820	653	0.35	**0.00**	0.69	1650	**122**	**0.00**	**0.00**	**0.00**
5_9	279	20	823	**0.00**	**0.00**	0.36	975	**78**	**0.00**	**0.00**	**0.00**
5_10	265	2370	617	0.38	**0.00**	0.75	1350	**112**	**0.00**	**0.00**	**0.00**
6_1	138	3050	802	1.45	**0.00**	**1.45**	2175	**332**	**0.00**	**0.00**	**1.45**
6_2	146	1180	**122**	**0.00**	**0.00**	**0.00**	2025	164	**0.00**	**0.00**	**0.00**
6_3	145	1530	128	**0.00**	**0.00**	**0.00**	900	**70**	**0.00**	**0.00**	**0.00**
6_4	131	930	70	**0.00**	**0.00**	**0.00**	300	**22**	**0.00**	**0.00**	**0.00**
6_5	161	1130	831	1.24	**0.00**	2.48	2025	**378**	**0.00**	**0.00**	**1.86**
a_1	253	2725	**904**	0.79	0.79	0.79	4875	976	**0.40**	**0.40**	**0.40**
a_2	252	4700	**908**	1.19	1.19	1.19	3000	967	**0.40**	**0.00**	**0.79**
a_3	232	2925	**903**	**0.43**	**0.43**	**0.43**	5700	971	**0.43**	**0.43**	**0.43**
a_4	234	1575	895	0.43	0.43	**0.43**	3150	**444**	**0.00**	**0.00**	**0.43**
a_5	236	1815	902	0.42	0.42	**0.42**	3675	**478**	**0.00**	**0.00**	0.85
b_1	69	360	**45**	**0.00**	**0.00**	**0.00**	450	**45**	**0.00**	**0.00**	**0.00**
b_2	76	370	**37**	**0.00**	**0.00**	**0.00**	900	93	**0.00**	**0.00**	**0.00**
b_3	80	320	34	**0.00**	**0.00**	**0.00**	300	**33**	**0.00**	**0.00**	**0.00**
b_4	79	380	**38**	**0.00**	**0.00**	**0.00**	450	44	**0.00**	**0.00**	**0.00**
b_5	72	10	**1**	**0.00**	**0.00**	**0.00**	150	16	**0.00**	**0.00**	**0.00**
c_1	227	2690	**973**	1.32	**0.88**	1.76	6225	1034	**0.88**	**0.88**	**0.88**
c_2	219	2780	**979**	**0.91**	**0.91**	**0.91**	3975	1056	**0.91**	**0.91**	**0.91**
c_3	243	3305	**993**	1.65	0.82	2.47	5775	1067	**0.82**	**0.41**	**1.23**
c_4	219	2400	**235**	**0.00**	**0.00**	**0.00**	3900	380	**0.00**	**0.00**	**0.00**
c_5	215	1305	**979**	**0.47**	**0.00**	0.93	2925	1052	**0.47**	0.47	**0.47**
d_1	60	415	1174	1.67	**0.00**	1.67	2025	**473**	**0.00**	**0.00**	**0.00**
d_2	66	1675	**236**	**0.00**	**0.00**	**0.00**	1800	350	**0.00**	**0.00**	**0.00**
d_3	72	775	1198	**1.39**	1.39	**1.39**	975	**1019**	**1.39**	**0.00**	**1.39**
d_4	62	235	25	**0.00**	**0.00**	**0.00**	150	**20**	**0.00**	**0.00**	**0.00**
d_5	61	600	**63**	**0.00**	**0.00**	**0.00**	675	75	**0.00**	**0.00**	**0.00**
nre_1	29	100	29	**0.00**	**0.00**	**0.00**	50	**18**	**0.00**	**0.00**	**0.00**
nre_2	30	1650	902	**0.00**	**0.00**	3.33	1400	**423**	**0.00**	**0.00**	**0.00**
nre_3	27	900	216	**0.00**	**0.00**	**0.00**	700	**170**	**0.00**	**0.00**	**0.00**
nre_4	28	1200	452	**0.00**	**0.00**	3.57	1000	**260**	**0.00**	**0.00**	**0.00**
nre_5	28	200	49	**0.00**	**0.00**	**0.00**	100	**31**	**0.00**	**0.00**	**0.00**
nrf_1	14	200	118	**0.00**	**0.00**	**0.00**	200	**112**	**0.00**	**0.00**	**0.00**
nrf_2	15	100	64	**0.00**	**0.00**	**0.00**	75	**40**	**0.00**	**0.00**	**0.00**
nrf_3	14	100	5464	**7.14**	**0.00**	**7.14**	50	**4966**	**7.14**	**0.00**	**7.14**
nrf_4	14	500	271	**0.00**	**0.00**	**0.00**	200	**122**	**0.00**	**0.00**	**0.00**
nrf_5	13	100	**5511**	**7.69**	**7.69**	**7.69**	100	5614	**7.69**	**7.69**	**7.69**

Table 4: Continued.

Instance	z_{opt}	d-ACO					n-ACO				
		$\overline{\text{evals}}$	$\overline{\text{time (s)}}$	$\overline{\text{rpd}}$ (%)	rpd_{min} (%)	rpd_{max} (%)	$\overline{\text{evals}}$	$\overline{\text{time (s)}}$	$\overline{\text{rpd}}$ (%)	rpd_{min} (%)	rpd_{max} (%)
nrg_1	176	5750	1905	0.28	**0.00**	2.27	5350	**1250**	**0.00**	**0.00**	**0.00**
nrg_2	154	4800	**2166**	1.95	1.95	**1.95**	3425	2171	**1.30**	**0.65**	**1.95**
nrg_3	166	4350	**2253**	3.61	3.01	4.82	2775	2253	**3.01**	**2.41**	**4.22**
nrg_4	168	5250	**2207**	2.68	**1.19**	3.57	4100	2217	**2.38**	**1.19**	**2.98**
nrg_5	168	6450	2258	0.60	0.60	**0.60**	5500	**1830**	**0.00**	**0.00**	**0.60**
nrh_1	63	3650	**5499**	**3.17**	**1.59**	4.76	3525	5768	**3.17**	3.17	**3.17**
nrh_2	63	3300	**5437**	**3.17**	**3.17**	**3.17**	2100	5706	**3.17**	**3.17**	**3.17**
nrh_3	59	4200	**5511**	**3.39**	**3.39**	**3.39**	2750	5776	**3.39**	**3.39**	**3.39**
nrh_4	58	3600	**5434**	**3.45**	**1.72**	**3.45**	2200	5664	**3.45**	**1.72**	**3.45**
nrh_5	55	4250	2329	**0.00**	**0.00**	**0.00**	2975	**1708**	**0.00**	**0.00**	**0.00**

Table 5: RPD metrics, average total execution time, and average number of evaluations obtained during the second study for ABC algorithm.

Instance	z_{opt}	d-ABC					n-ABC				
		$\overline{\text{evals}}$	$\overline{\text{time (s)}}$	$\overline{\text{rpd}}$ (%)	rpd_{min} (%)	rpd_{max} (%)	$\overline{\text{evals}}$	$\overline{\text{time (s)}}$	$\overline{\text{rpd}}$ (%)	rpd_{min} (%)	rpd_{max} (%)
4_1	429	125,808	485	0.82	**0.23**	0.93	124,122	447	**0.23**	**0.23**	**0.23**
4_2	512	9998	93	**0.00**	**0.00**	**0.00**	8194	76	**0.00**	**0.00**	**0.00**
4_3	516	56027	390	**0.00**	**0.00**	0.19	11,191	103	**0.00**	**0.00**	**0.00**
4_4	494	128,399	535	**0.20**	**0.00**	0.61	126,209	488	**0.20**	0.20	**0.20**
4_5	512	11,161	102	**0.00**	**0.00**	**0.00**	118,961	354	0.39	**0.00**	**0.39**
4_6	560	127,988	537	0.36	**0.00**	0.54	28,428	254	**0.00**	**0.00**	**0.36**
4_7	430	110,922	207	0.93	**0.00**	1.16	110,224	194	**0.70**	0.47	**0.70**
4_8	492	126,117	491	0.20	0.20	**0.20**	34,292	283	**0.00**	**0.00**	**0.20**
4_9	641	128,322	526	0.94	**0.00**	2.18	128,276	523	**0.47**	**0.00**	**0.78**
4_10	514	18,162	164	**0.00**	**0.00**	**0.00**	55,670	333	**0.00**	**0.00**	0.58
5_1	253	125,020	1797	**0.40**	**0.00**	1.58	123,792	1706	**0.40**	**0.00**	**0.40**
5_2	302	127,193	1956	1.99	1.66	2.98	125,859	1856	**0.99**	**0.00**	**1.32**
5_3	226	123,812	1713	1.33	1.33	1.33	123,627	1696	**0.88**	**0.88**	**0.88**
5_4	242	17,751	650	**0.00**	**0.00**	**0.00**	6760	239	**0.00**	**0.00**	**0.00**
5_5	211	14,945	521	**0.00**	**0.00**	**0.00**	7927	277	**0.00**	**0.00**	**0.00**
5_6	213	19,299	684	**0.00**	**0.00**	**0.00**	5575	200	**0.00**	**0.00**	**0.00**
5_7	293	126,401	1924	0.34	0.34	1.02	14,753	547	**0.00**	**0.00**	**0.00**
5_8	288	17,153	652	**0.00**	**0.00**	**0.00**	23,534	814	**0.00**	**0.00**	**0.00**
5_9	279	124,770	1754	0.36	0.36	0.36	8961	316	**0.00**	**0.00**	**0.00**
5_10	265	86,539	1149	0.19	**0.00**	1.13	12,431	407	**0.00**	**0.00**	**0.00**
6_1	138	124,417	1198	1.45	**0.00**	2.17	91,269	916	**0.36**	**0.00**	**1.45**
6_2	146	8784	217	**0.00**	**0.00**	**0.00**	6381	156	**0.00**	**0.00**	**0.00**
6_3	145	14,699	359	**0.00**	**0.00**	**0.00**	4776	115	**0.00**	**0.00**	**0.00**
6_4	131	16,928	400	**0.00**	**0.00**	**0.00**	3961	95	**0.00**	**0.00**	**0.00**
6_5	161	13,760	338	**0.00**	**0.00**	**0.00**	4377	106	**0.00**	**0.00**	**0.00**
a_1	253	123,573	5729	**0.40**	**0.40**	0.79	122,222	5381	**0.40**	**0.40**	**0.40**
a_2	252	126,016	6309	1.39	0.79	2.38	126,212	6346	**0.79**	**0.00**	**1.19**
a_3	232	126,103	6334	1.29	0.86	1.72	125,826	6189	**0.43**	**0.00**	**0.86**
a_4	234	125,908	6287	**0.43**	**0.43**	**0.43**	124,897	6004	**0.43**	**0.43**	**0.43**
a_5	236	124,785	5948	0.42	0.42	0.42	11,144	1346	**0.00**	**0.00**	**0.00**
b_1	69	120,832	13,012	1.45	**0.00**	1.45	3970	1228	**0.00**	**0.00**	**0.00**
b_2	76	6962	2054	**0.00**	**0.00**	**0.00**	3372	1043	**0.00**	**0.00**	**0.00**
b_3	80	4170	1296	**0.00**	**0.00**	**0.00**	3572	1106	**0.00**	**0.00**	**0.00**
b_4	79	6166	1932	**0.00**	**0.00**	**0.00**	3965	1227	**0.00**	**0.00**	**0.00**
b_5	72	3172	988	**0.00**	**0.00**	**0.00**	2774	857	**0.00**	**0.00**	**0.00**
c_1	227	121,978	12,589	**0.88**	**0.44**	1.76	116,551	9431	**0.88**	0.88	**0.88**
c_2	219	110,309	13,899	0.23	**0.00**	1.37	21,664	6492	**0.00**	**0.00**	**0.00**
c_3	243	126,728	15,246	1.65	0.41	2.47	126,018	14,881	**0.82**	**0.00**	**2.06**
c_4	219	126,196	14,951	0.46	**0.00**	1.83	28,355	7658	**0.00**	**0.00**	**0.00**
c_5	215	19,075	5434	**0.00**	**0.00**	**0.00**	14,546	4086	**0.00**	**0.00**	**0.00**

Table 5: Continued.

Instance	z_{opt}	d-ABC					n-ABC				
		$\overline{\text{evals}}$	$\overline{\text{time(s)}}$	$\overline{\text{rpd}}$ (%)	rpd_{min} (%)	rpd_{max} (%)	$\overline{\text{evals}}$	$\overline{\text{time(s)}}$	$\overline{\text{rpd}}$ (%)	rpd_{min} (%)	rpd_{max} (%)
d_1	60	5970	4455	**0.00**	**0.00**	**0.00**	7572	5603	**0.00**	**0.00**	**0.00**
d_2	66	121,272	31,555	1.52	1.52	1.52	11,343	8371	**0.00**	**0.00**	**0.00**
d_3	72	122,829	34,063	1.39	1.39	2.78	18,808	13,510	**0.00**	**0.00**	**0.00**
d_4	62	121,976	32,383	1.61	1.61	1.61	4768	3510	**0.00**	**0.00**	**0.00**
d_5	61	109,457	14,077	1.64	1.64	1.64	5090	3690	**0.00**	**0.00**	**0.00**
nre_1	29	27,684	68,066	**0.00**	**0.00**	**0.00**	3479	9634	**0.00**	**0.00**	**0.00**
nre_2	30	119,319	111,213	6.67	6.67	6.67	18,666	51,982	**0.00**	**0.00**	**3.33**
nre_3	27	120,070	115,028	7.41	7.41	7.41	19,339	53,470	**0.00**	**0.00**	**0.00**
nre_4	28	119,063	109,812	10.71	3.57	14.29	10,559	30,055	**0.00**	**0.00**	**0.00**
nre_5	28	118,585	106,781	3.57	3.57	7.14	4053	11,203	**0.00**	**0.00**	**0.00**
nrf_1	14	119,200	230,417	14.29	14.29	14.29	2353	13,741	**0.00**	**0.00**	**0.00**
nrf_2	15	119,831	238,705	13.33	13.33	13.33	1950	11,337	**0.00**	**0.00**	**0.00**
nrf_3	14	119,332	236,611	14.29	14.29	21.43	3929	23,123	**0.00**	**0.00**	**0.00**
nrf_4	14	119,910	241,666	14.29	14.29	14.29	2746	16,074	**0.00**	**0.00**	**0.00**
nrf_5	13	106,580	79,141	15.38	15.38	23.08	120,731	248,033	**7.69**	**7.69**	**7.69**
nrg_1	176	128,776	257,664	2.84	2.27	3.41	126,471	235,636	**1.14**	**0.57**	**1.70**
nrg_2	154	128,678	253,564	3.25	2.60	3.90	126,160	233,168	**1.95**	**1.30**	**2.60**
nrg_3	166	134,516	243,567	3.61	3.01	4.22	124,477	217,781	**2.41**	**1.81**	**3.01**
nrg_4	168	135,789	223,456	3.57	2.98	4.17	125,268	224,757	**1.79**	**1.19**	**2.38**
nrg_5	168	126,782	227,346	4.17	3.57	4.76	126,077	232,047	**2.38**	**2.38**	**2.98**
nrh_1	63	126,421	228,326	3.17	3.17	4.76	145,678	253,467	**1.59**	**1.59**	**3.17**
nrh_2	63	134,282	217,456	3.17	4.76	4.76	155,671	243,457	**1.59**	**1.59**	**3.17**
nrh_3	59	123,481	237,222	3.39	5.08	5.08	132,451	233,245	**1.69**	**1.69**	**3.39**
nrh_4	58	143,562	242,456	3.45	3.45	5.17	142,457	231,002	**1.72**	**1.72**	**3.45**
nrh_5	55	134,612	232,216	3.64	3.64	3.64	143,417	224,452	**1.82**	**1.82**	**3.64**

times, we reach a similar behaviour, where n-ACO appears to need a shorter time than d-ACO, except for b, c, and nrh instances, and n-ABC appears to need a shorter time than d-ABC in general. Focusing on evaluations, the $\overline{\text{evals}}$ field is related to the number of iterations reached as follows. In ACO, ps_{aco} evaluations are performed for each iteration. In ABC, a number of evaluations varying between ps_{abc} and two times ps_{abc} are performed for each iteration. Thus, for ABC, the maximum number of evaluations will be a value in the range $[100,000, 200,000]$. For ACO, the maximum number of evaluations will be 10,000 as defined before. Analysing the $\overline{\text{evals}}$ field, the authors reach that the number of evaluations needed is distant from the stop condition defined, and then the stop condition is adequate in both studies.

Table 6 shows the average RPD metrics for each instance group, where "ipv" field denotes the percentage of improvement by considering the alternative approach of the algorithm instead of the default version. Analysing this table, it is observed that (i) n-ACO provides better RPD values than d-ACO for all the groups and (ii) n-ABC also provides better RPD values than its default version. The RPD metrics obtained are in line with other works from the literature, with RPD values lower than 1.0%. In this regard, Table 7 shows the values of some recent successful approaches solving the problem. However, we should remark that the purpose of this work is not to outperform other techniques solving the standard SCP benchmark.

At this point, it seems that the alternative approach of the algorithms provides better performance in both cases. However, we do not know if the differences observed are significant. To this end, the statistical methodology procedure described by Lanza-Gutierrez et al. [56] was applied. First, we removed all possible outliers. Then, we analysed the normality of data, obtaining that we cannot assume normal distribution in any case. Consequently, the median should be considered as average value for calculating $\overline{z}$ in equation (25).

Next, we study if there are significant differences in the solution quality of the algorithms. Starting with the first study, we consider the Wilcoxon–Mann–Whitney test with hypotheses H_0: $\overline{\text{rpd}}_a \geq \overline{\text{rpd}}_b$ and H_1: $\overline{\text{rpd}}_a < \overline{\text{rpd}}_b$, $\forall a, b \in \{\text{d} - \text{ACO}, \text{n} - \text{ACO}\}$ with $a \neq b$, where $\overline{\text{rpd}}_a$ and $\overline{\text{rpd}}_b$ are the average RPD of the algorithm a and b for a given instance, respectively. The p values obtained for each instance and ACO approach are shown in Table 8 under the title *RPD analysis*, where p values lower than the significance level $\alpha = 0.05$ are given in bold, i.e., the confidence level is 0.95. Note that the unilateral test performed between the two possibilities was the one that matches with the descriptive analysis, the other test being marked with a dash in the table. Also note that in case of equality between the average RPD values, the two unilateral tests are performed. For the second study, we consider the Wilcoxon–Mann–Whitney test with similar hypotheses as before H_0: $\overline{\text{rpd}}_c \geq \overline{\text{rpd}}_d$ and H_1: $\overline{\text{rpd}}_c < \overline{\text{rpd}}_d$, $\forall c, d \in \{\text{d} - \text{ABC}, \text{n} - \text{ABC}\}$ with $c \neq d$. The p values obtained are also shown in Table 9 with the same notation as in Table 8.

Table 6: RPD metrics for each instance group and algorithm in both studies.

Instance group	$\overline{\text{rpd}}$			rpd_{min}			rpd_{max}		
	d-ACO (%)	n-ACO (%)	ipv	d-ACO (%)	n-ACO (%)	ipv	d-ACO (%)	n-ACO (%)	ipv
First study									
A	0.43	**0.08**	81.80	0.08	**0.03**	63.83	0.79	**0.43**	46.03
B	0.53	**0.28**	46.60	0.36	**0.17**	51.89	0.62	**0.39**	37.24
C	1.86	**1.74**	6.54	1.22	**1.17**	3.74	2.49	**1.89**	24.06
ALL	0.94	**0.70**	25.69	0.55	**0.46**	17.20	1.30	**0.90**	30.60
Second study									
Instance group	$\overline{\text{rpd}}$			rpd_{min}			rpd_{max}		
	d-ABC (%)	n-ABC (%)	ipv	d-ABC (%)	n-ABC (%)	ipv	d-ABC (%)	n-ABC (%)	ipv (%)
A	0.38	**0.19**	51.25	0.16	**0.07**	56.65	0.66	**0.30**	54.25
B	0.74	**0.19**	74.56	0.50	**0.09**	82.81	1.11	**0.29**	73.77
C	10.71	**1.07**	90.01	9.91	**0.90**	90.95	12.75	**1.48**	88.39
ALL	3.94	**0.48**	87.80	3.52	**0.35**	90.04	4.84	**0.69**	85.73

Table 7: Review of recent approaches solving Beasley's OR library.

Author/s	SIA	$\overline{\text{rpd}}$ (%)	rpd_{min} (%)	rpd_{max} (%)
Farahani et al. [41]	ABC	0.16	0.01	—
Ren et al. [43]	ACO	0.20	0.03	0.39
Lu and Vasko [48]	TLBO20	—	0.06	—
Lu and Vasko [48]	TLBO10	—	0.09	—
Naji-Azimi et al. [49]	EM-like	—	0.20	—
Lu and Vasko [48]	TLBO	—	0.28	—

Table 8: p values obtained analysing the differences observed regarding execution time and solution quality during the first study for the ACO algorithm.

	RPD analysis (H_1)		Execution time analysis (H_1)	
	d-ACO < n-ACO	n-ACO < d-ACO	d-ACO < n-ACO	n-ACO < d-ACO
Inst				
4_1	—	**≤0.001**	—	**≤ 0.001**
4_2	—	**≤0.001**	—	**≤0.001**
4_3	—	**≤0.001**	—	**≤0.001**
4_4	—	**≤0.001**	—	**≤0.001**
4_5	—	**0.003**	—	**0.006**
4_6	—	**≤0.001**	—	**≤0.001**
4_7	—	**≤0.001**	—	**≤0.001**
4_8	—	**≤0.001**	—	**≤0.001**
4_9	—	**≤0.001**	—	**≤0.001**
4_10	0.500	0.500	—	**≤0.001**
5_1	—	**0.004**	—	**≤0.001**
5_2	—	0.131	—	**≤0.001**
5_3	—	**≤0.001**	—	**≤0.001**
5_4	0.500	0.500	—	**≤0.001**
5_5	—	**≤0.001**	—	**≤0.001**
5_6	0.500	0.500	—	**≤0.001**
5_7	—	**≤0.001**	—	**≤0.001**
5_8	—	**≤0.001**	—	**≤0.001**
5_9	—	0.060	—	**≤0.001**
5_10	—	**≤0.001**	—	**≤0.001**
6_1	—	**0.002**	—	0.150
6_2	0.500	0.500	0.117	—
6_3	0.500	0.500	—	**≤0.001**
6_4	0.500	0.500	—	**≤0.001**
6_5	—	**0.005**	—	0.054
a_1	—	**≤0.001**	**≤ 0.001**	—

Table 8: Continued.

	RPD analysis (H_1)		Execution time analysis (H_1)	
	d-ACO < n-ACO	n-ACO < d-ACO	d-ACO < n-ACO	n-ACO < d-ACO
a_2	—	≤ **0.001**	**0.006**	—
a_3	0.500	0.500	≤ **0.001**	—
a_4	—	≤ **0.001**	—	**0.001**
a_5	—	≤ **0.001**	—	≤ **0.001**
b_1	0.500	0.500	0.291	—
b_2	0.500	0.500	≤ **0.001**	—
b_3	0.500	0.500	—	0.565
b_4	0.500	0.500	0.106	—
b_5	0.500	0.500	≤ **0.001**	—
c_1	—	≤ **0.001**	≤ **0.001**	—
c_2	0.500	0.500	≤ **0.001**	—
c_3	—	≤ **0.001**	≤ **0.001**	—
c_4	0.500	0.500	**0.013**	—
c_5	—	**0.002**	≤ **0.001**	—
d_1	—	≤ **0.001**	—	**0.002**
d_2	0.500	0.500	0.124	—
d_3	—	≤ **0.001**	—	**0.003**
d_4	0.500	0.500	—	**0.017**
d_5	0.500	0.500	0.059	—
nre_1	0.500	0.500	—	≤ **0.001**
nre_2	—	≤ **0.001**	—	≤ **0.001**
nre_3	0.500	0.500	—	0.112
nre_4	—	**0.002**	—	**0.006**
nre_5	0.500	0.500	—	≤ **0.001**
nrf_1	0.500	0.500	—	0.108
nrf_2	0.500	0.500	—	≤ **0.001**
nrf_3	0.500	0.500	—	0.292
nrf_4	0.500	0.500	—	≤ **0.001**
nrf_5	0.500	0.500	≤ **0.001**	—
nrg_1	—	≤ **0.001**	—	≤ **0.001**
nrg_2	—	≤ **0.001**	0.261	—
nrg_3	—	≤ **0.001**	0.378	—
nrg_4	—	**0.038**	0.574	—
nrg_5	—	≤ **0.001**	—	≤ **0.001**
nrh_1	—	0.001	≤ **0.001**	—
nrh_2	0.500	0.500	≤ **0.001**	—
nrh_3	0.500	0.500	≤ **0.001**	—
nrh_4	—	0.261	≤ **0.001**	—
nrh_5	0.500	0.500	—	**0.003**

Table 9: p values obtained analysing the differences observed regarding execution time and solution quality during the second study for the ABC algorithm.

Inst	RPD analysis (H_1)		Execution time analysis (H_1)	
	d-ABC < n-ABC	n-ABC < d-ABC	d-ABC < n-ABC	n-ABC < d-ABC
4_1	—	≤**0.001**	—	≤**0.001**
4_2	0.500	0.500	—	≤**0.001**
4_3	—	≤**0.001**	—	≤**0.001**
4_4	**0.037**	—	—	0.054
4_5	≤**0.001**	—	≤**0.001**	—
4_6	—	≤**0.001**	—	≤**0.001**
4_7	—	≤**0.001**	—	≤**0.001**
4_8	—	≤**0.001**	—	≤**0.001**
4_9	—	≤**0.001**	—	0.459
4_10	≤**0.001**	—	≤**0.001**	—

Table 9: Continued.

Inst	RPD analysis (H_1)		Execution time analysis (H_1)	
	d-ABC < n-ABC	n-ABC < d-ABC	d-ABC < n-ABC	n-ABC < d-ABC
5_1	—	**≤0.001**	—	**0.001**
5_2	—	**≤0.001**	—	**≤0.001**
5_3	—	**≤0.001**	0.415	—
5_4	0.500	0.500	—	**≤0.001**
5_5	0.500	0.500	—	**≤0.001**
5_6	0.500	0.500	—	**≤0.001**
5_7	—	**≤0.001**	—	**≤0.001**
5_8	0.500	0.500	0.348	—
5_9	—	**≤0.001**	—	**≤0.001**
5_10	—	**≤0.001**	—	**≤0.001**
6_1	—	**0.001**	—	**≤0.001**
6_2	0.500	0.500	—	**≤0.001**
6_3	0.500	0.500	—	**≤0.001**
6_4	0.500	0.500	—	**≤0.001**
6_5	0.500	0.500	—	**≤0.001**
a_1	—	**0.001**	—	**≤0.001**
a_2	—	**≤0.001**	0.479	—
a_3	—	**≤0.001**	—	0.135
a_4	0.500	0.500	—	**0.025**
a_5	—	**≤0.001**	—	**≤0.001**
b_1	—	**≤0.001**	—	**≤0.001**
b_2	0.500	0.500	—	**≤0.001**
b_3	0.500	0.500	—	**0.012**
b_4	0.500	0.500	—	**≤0.001**
b_5	0.500	0.500	—	**≤0.001**
c_1	—	**0.001**	—	**≤0.001**
c_2	—	**≤0.001**	—	**≤0.001**
c_3	—	**≤0.001**	—	**0.022**
c_4	—	**≤0.001**	—	**≤0.001**
c_5	0.500	0.500	—	**0.001**
d_1	0.500	0.500	0.097	—
d_2	—	**≤0.001**	—	**≤0.001**
d_3	—	**≤0.001**	—	**≤0.001**
d_4	—	**≤0.001**	—	**≤0.001**
d_5	—	**≤0.001**	—	**≤0.001**
nre_1	0.500	0.500	—	**≤0.001**
nre_2	—	**≤0.001**	—	**0.003**
nre_3	—	**≤0.001**	—	**≤0.001**
nre_4	—	**≤0.001**	—	**≤0.001**
nre_5	—	**≤0.001**	***x*0.001**	—
nrf_1	—	**≤0.001**	—	**≤0.001**
nrf_2	—	**≤0.001**	—	**≤0.001**
nrf_3	—	**≤0.001**	—	**≤0.001**
nrf_4	—	**≤0.001**	—	**≤0.001**
nrf_5	—	**≤0.001**	**≤0.001**	—
nrg_1	—	**≤0.001**	0.500	0.500
nrg_2	—	**≤0.001**	0.500	0.500
nrg_3	—	**≤0.001**	0.500	0.500
nrg_4	—	**≤0.001**	0.500	0.500
nrg_5	—	**≤0.001**	0.500	0.500
nrh_1	—	**≤0.001**	0.500	0.500
nrh_2	—	**≤0.001**	0.500	0.500
nrh_3	—	**≤0.001**	0.500	0.500
nrh_4	—	**≤0.001**	0.500	0.500
nrh_5	—	**≤0.001**	0.500	0.500

Table 10: Percentage of cases where an algorithm provides the best significant performance compared to the other regarding solution quality and execution time.

	RPD analysis					Execution time analysis			
	A (%)	B (%)	C (%)	ALL (%)		A (%)	B (%)	C (%)	ALL (%)
First study									
n-ACO > d-ACO	26.15	13.85	12.31	**52.31**	n-ACO > d-ACO	33.85	7.69	13.85	**55.38**
d-ACO > n-ACO	0.00	0.00	0.00	0.00	d-ACO > n-ACO	0.00	15.38	7.69	23.08
n-ACO = d-ACO	12.31	16.92	18.46	47.69	n-ACO = d-ACO	4.62	7.69	9.23	21.54
Percentage	38.46	30.77	30.77	100.00	Percentage	38.46	30.77	30.77	100.00
Second study									
n-ABC > d-ABC	23.64	23.64	16.36	**63.64**	n-ABC > d-ABC	29.23	26.15	13.85	**69.23**
d-ABC > n-ABC	5.45	0.00	1.82	7.27	d-ABC > n-ABC	3.08	0.00	1.54	4.62
n-ABC = d-ABC	16.36	12.73	0.00	29.09	n-ABC = d-ABC	6.15	4.62	15.38	26.15
Percentage	45.45	36.36	18.18	100.00	Percentage	38.46	30.77	30.77	100.00

Through the Wilcoxon–Mann–Whitney test, the differences observed regarding execution time for both algorithms are also analysed. For the first study, we define the hypotheses H_0: $\overline{\text{time(s)}}_a \geq \overline{\text{time(s)}}_b$ and H_1: $\overline{\text{time(s)}}_a < \overline{\text{time(s)}}_b$, $\forall a, b \in \{\text{d} - \text{ACO}, \text{n} - \text{ACO}\}$ with $a \neq b$, where $\overline{\text{time(s)}}_a$ and $\overline{\text{time(s)}}_b$ are the average execution time of the algorithm a and b for a given instance, respectively. The p values obtained for each instance and algorithm are shown in Table 8 under the title *Execution time analysis*, where p values are given in bold with the same criterion as before. For the second study, we consider the Wilcoxon–Mann–Whitney test with similar hypotheses as before H_0: $\overline{\text{time(s)}}_c \geq \overline{\text{time(s)}}_d$ and H_1: $\overline{\text{time(s)}}_c < \overline{\text{time(s)}}_d$, $\forall c, d \in \{\text{d} - \text{ABC}, \text{n} - \text{ABC}\}$ with $c \neq d$. The p values obtained are also shown in Table 9.

Based on the previous statistical analysis, Table 10 shows the percentage of cases where an algorithm provides the best significant performance compared to another for each study in terms of RPD and execution time. Focusing on the first study and RPD values, we verify that n-ACO provides better behaviour than d-ACO in 52.31% of cases. However, it is important to remark that d-ACO never provides better results than n-ACO, meaning that n-ACO clearly outperforms d-ACO. For execution time, n-ACO needs lower execution times than d-ACO in 55.38% of cases and d-ACO needs lower execution times than n-ACO in 23.08% of cases. This fact could mean that the alternative heuristic information needs higher execution times under certain conditions. However, most cases in which d-ACO needs lower execution times correspond with instances whose optimal solution is not reached, and then the 10,000 evaluations are performed for each algorithm, e.g., for instances a_1, a_2, a_3, c_1, c_2, c_3, c_5, nrh_1, nrh_2, nrh_3, and nrh_4. In such unfavourable cases, the differences observed are not of concern as shown in Figure 2(a). On the other hand, most cases in which n-ACO needs lower execution times correspond with instances whose optimal solution is reached. In such cases, a greater difference is observed favoring n-ACO. This fact is because n-ACO reaches the stop condition before d-ACO, e.g., for instance sets 4, 5, 6, d, nre, and nrg. Focusing on the second study, we verify that n-ABC provides better behaviour than d-ABC in 63.64% of cases, where d-ABC outperforms n-ABC in 7.27% of cases. This unfavourable situation occurs in small instances, where the search space is reduced. For execution time, n-ABC needs lower execution times than d-ABC in 69.23% of cases and d-ABC needs lower execution times than n-ABC in 4.62% of cases. As before, this unfavourable situation mainly occurs when n-ABC does not reach the optimal solution, penalizing the additional computation of the gain concept in the alternative SCP formulation. This behaviour is shown in Figure 2(b).

The previous analysis is completed with the landscape study in Tables 11 and 12 for the solutions obtained solving the instances through ACO and ABC, respectively. The metrics in such tables quantify solution quality (QMetric $\in [0, 1]$), the rate of success (SRate $\in [0, 1]$), and speed of reaching a solution (SSpeed $\in [0, 1]$). QMetric follows an exponential formulation which allows distinguishing between the performance of two algorithms, which obtained solutions close to the optimum fitness. SRate is defined as the number of successful runs that the algorithm reaches the optimum fitness divided by the total number of runs. SSpeed quantifies the number of evaluations taken to reach the optimum fitness. For the three metrics, the value 1 indicates the highest quality. More details about the three metrics, as well as formulation, are listed in [71]. Analysing Tables 11 and 12, we check that, in general, both n-ACO and n-ABC provide a higher or equal QMetric than d-ACO and d-ABC approaches. For SSpeed, we check that both d-ACO and d-ABC need a lower or equal number of evaluations than n-ACO and d-ABC to reach the optimum fitness. For SRate, we check that both d-ACO and d-ABC reach the optimum fitness a greater or equal number of times than n-ACO and d-ABC. That means that n-ACO and n-ABC obtained better quality solutions, are better in convergence, and provide a more robust performance than the default approaches.

Up to this point, we know that the concepts included in the alternative formulation positively affect the search process in the first study, where the concepts from the alternative formulation are considered for guiding purposes. A mapping of the solutions visited by n-ACO and d-ACO could help to effectively show how the two approaches explore the search space. To this end, we consider the mapping method (MaM) proposed by Autuori et al. [72], where a mapping function converts a multidimensional space solution in one dimensional space through two steps: (i) a binary conversion is applied to the solution (for SCP,

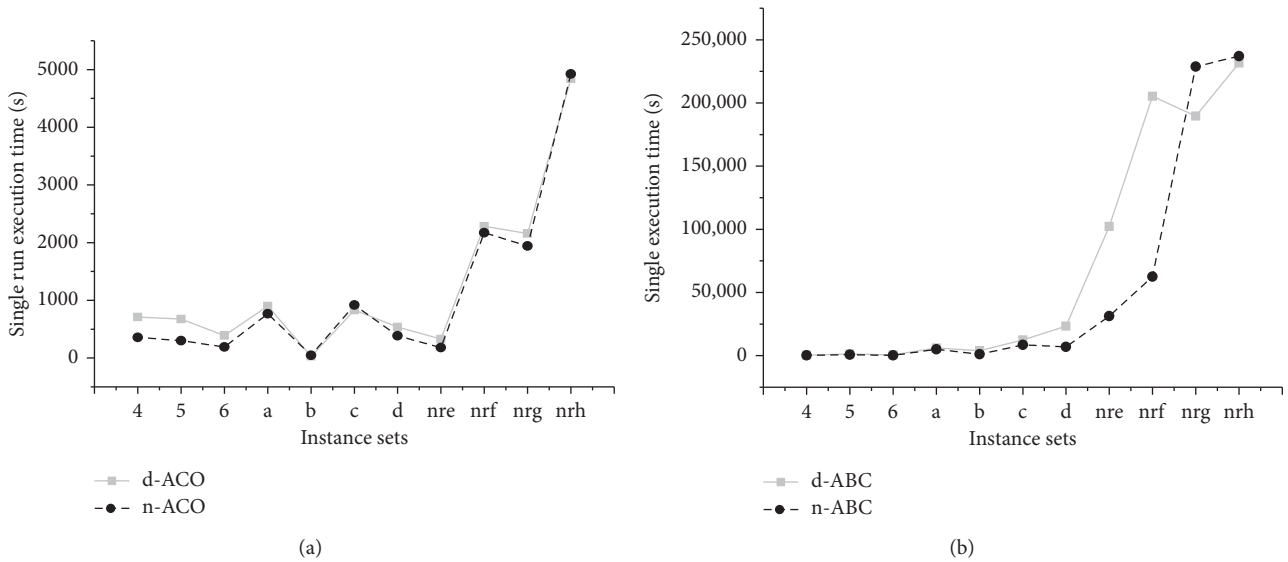

Figure 2: Analysis for the average execution times in each study. (a) Average execution time for each ACO approach and instance set. (b) Average execution time for each ABC approach and instance set.

Table 11: Landscape metrics for each instance and ACO approach.

Inst	d-ACO			n-ACO		
	QMetric	SSpeed	SRate	QMetric	SSpeed	SRate
4_1	0.0008	0.5107	0.1000	**1.0000**	**0.8366**	**0.9667**
4_2	0.1777	0.4248	0.1667	**1.0000**	**0.7190**	**1.0000**
4_3	0.0019	0.0000	0.0000	**0.4467**	**0.5119**	**0.4333**
4_4	0.0027	0.0000	0.0000	**0.0114**	**0.6300**	**0.1000**
4_5	0.5029	0.6151	0.5000	**0.5695**	**0.7979**	**0.5667**
4_6	0.6473	0.5445	0.6333	**1.0000**	**0.6931**	**0.8667**
4_7	0.0669	0.8310	0.0667	**1.0000**	**0.8382**	**0.9333**
4_8	0.6751	0.6144	0.6667	**1.0000**	**0.8425**	**1.0000**
4_9	0.0005	0.0000	0.0000	**0.1045**	**0.4400**	**0.1000**
4_10	**1.0000**	0.8497	**1.0000**	**1.0000**	**0.9080**	**1.0000**
5_1	0.3336	0.4622	0.3333	**0.6335**	**0.5903**	**0.6333**
5_2	0.0000	**0.0000**	**0.0000**	**0.0007**	**0.0000**	**0.0000**
5_3	0.4334	0.5965	0.4333	**1.0000**	**0.8098**	**0.9333**
5_4	**1.0000**	0.6987	**1.0000**	**1.0000**	**0.8650**	**1.0000**
5_5	0.0000	**0.8640**	0.0333	**0.6333**	**0.8066**	**0.6333**
5_6	**1.0000**	0.9183	**1.0000**	**1.0000**	**0.9615**	**1.0000**
5_7	0.0009	0.0000	0.0000	**0.0017**	**0.7800**	**0.2000**
5_8	0.5340	0.5448	0.5333	**1.0000**	**0.8215**	**1.0000**
5_9	0.4017	0.5528	0.4000	**1.0000**	**0.8775**	**1.0000**
5_10	0.4669	0.6416	0.4667	**1.0000**	**0.8216**	**0.9667**
6_1	0.2359	0.4803	0.2333	**0.5696**	**0.7115**	**0.5667**
6_2	**1.0000**	**0.8268**	0.8667	**1.0000**	0.7477	**0.9333**
6_3	**1.0000**	0.7902	0.9667	**1.0000**	**0.9130**	**1.0000**
6_4	**1.0000**	0.8791	**1.0000**	**1.0000**	**0.9645**	**1.0000**
6_5	0.2355	**0.7840**	0.2333	**0.5350**	0.7516	**0.5333**
a_1	0.0037	**0.0000**	**0.0000**	**0.0517**	**0.0000**	**0.0000**
a_2	0.0001	0.0000	0.0000	**0.2202**	**0.7075**	**0.2000**
a_3	**0.0337**	**0.0000**	**0.0000**	**0.0337**	**0.0000**	**0.0000**
a_4	0.0306	0.0610	0.0333	**0.6769**	**0.5440**	**0.6667**
a_5	0.0350	0.5955	0.0667	**0.6772**	**0.6153**	**0.6667**
b_1	**1.0000**	0.9180	**1.0000**	**1.0000**	**0.9510**	**1.0000**
b_2	**1.0000**	**0.9584**	**1.0000**	**1.0000**	0.8810	**1.0000**
b_3	**1.0000**	0.9531	**1.0000**	**1.0000**	**0.9595**	**1.0000**
b_4	**1.0000**	**0.9487**	**1.0000**	**1.0000**	0.9425	**1.0000**
b_5	**1.0000**	**0.9990**	**1.0000**	**1.0000**	0.9850	**1.0000**

Table 11: Continued.

Inst	d-ACO			n-ACO		
	QMetric	SSpeed	SRate	QMetric	SSpeed	SRate
c_1	0.0021	0.0000	0.0000	**0.0086**	**0.9700**	**0.0333**
c_2	**0.0120**	**0.0000**	**0.0000**	**0.0120**	**0.0000**	**0.0000**
c_3	0.0055	**0.0000**	**0.0000**	**0.0785**	**0.0000**	**0.0000**
c_4	**1.0000**	**0.6953**	0.9667	**1.0000**	0.6230	**1.0000**
c_5	**0.0925**	0.6810	**0.0333**	0.0918	**0.8950**	**0.0333**
d_1	0.5069	0.5334	0.3000	**1.0000**	**0.6635**	**0.7667**
d_2	**1.0000**	0.6861	**0.9333**	**1.0000**	**0.7275**	0.8000
d_3	0.3280	**0.8020**	0.1000	**0.5968**	0.6775	**0.4000**
d_4	**1.0000**	0.9768	**1.0000**	**1.0000**	**0.9850**	**1.0000**
d_5	**1.0000**	**0.9288**	**1.0000**	**1.0000**	0.9115	**1.0000**
nre_1	**1.0000**	0.9900	**1.0000**	**1.0000**	**0.9950**	**1.0000**
nre_2	0.9319	0.7268	0.6333	**1.0000**	**0.7906**	**0.8333**
nre_3	**1.0000**	0.8629	0.9333	**1.0000**	**0.8947**	**1.0000**
nre_4	0.9499	**0.8336**	0.7333	**1.0000**	0.8086	**0.9667**
nre_5	**1.0000**	0.9800	**1.0000**	**1.0000**	**0.9877**	**1.0000**
nrf_1	**1.0000**	0.9717	**1.0000**	**1.0000**	**0.9782**	**1.0000**
nrf_2	**1.0000**	0.9847	**1.0000**	**1.0000**	**0.9898**	**1.0000**
nrf_3	**0.9621**	**0.8770**	**0.3333**	**0.9621**	0.7620	**0.3333**
nrf_4	**1.0000**	0.9407	0.9667	**1.0000**	**0.9683**	**1.0000**
nrf_5	**0.9441**	0.6980	**0.1667**	**0.9441**	**0.8650**	0.0667
nrg_1	0.7528	0.3613	0.5000	**1.0000**	**0.4760**	**0.8667**
nrg_2	0.2463	**0.0000**	**0.0000**	**0.3555**	**0.0000**	**0.0000**
nrg_3	0.0808	**0.0000**	**0.0000**	**0.1123**	**0.0000**	**0.0000**
nrg_4	0.1762	**0.0000**	**0.0000**	**0.2026**	**0.0000**	**0.0000**
nrg_5	0.6374	0.2350	0.2000	**0.8550**	**0.3203**	**0.6000**
nrh_1	0.7880	**0.0000**	**0.0000**	**0.8135**	**0.0000**	**0.0000**
nrh_2	0.8136	**0.0000**	**0.0000**	**0.8398**	**0.0000**	**0.0000**
nrh_3	**0.8148**	0.0000	0.0000	**0.8148**	**0.2700**	**0.0333**
nrh_4	0.8392	0.0000	0.0000	**0.8484**	**0.5900**	**0.0333**
nrh_5	**1.0000**	0.5407	0.9667	**1.0000**	**0.6670**	**1.0000**

Table 12: Landscape metrics for each instance and ABC approach.

Inst	d-ABC			n-ABC		
	QMetric	SSpeed	SRate	QMetric	SSpeed	SRate
4_1	0.0001	0.0000	0.0000	**0.0008**	**0.7873**	**0.1333**
4_2	**1.0000**	0.9486	**1.0000**	**1.0000**	**0.9598**	**1.0000**
4_3	0.5791	0.8366	0.5667	**1.0000**	**0.9454**	**1.0000**
4_4	**0.3068**	0.8238	**0.3000**	0.0114	**0.8441**	0.1333
4_5	**1.0000**	**0.9482**	**1.0000**	0.4337	0.8642	0.4333
4_6	0.0846	**0.8985**	0.0667	**0.7432**	0.8656	**0.7333**
4_7	**0.0333**	**0.9187**	**0.0333**	0.0000	0.0000	0.0000
4_8	0.0254	0.0000	0.0000	**0.7076**	**0.8662**	**0.7000**
4_9	0.0335	0.8215	0.0333	**0.0671**	**0.8849**	**0.0667**
4_10	**1.0000**	**0.9050**	**1.0000**	0.5385	0.8514	0.5333
5_1	0.0349	0.8071	0.0333	**0.3338**	**0.8912**	**0.3333**
5_2	0.0000	0.0000	0.0000	**0.1013**	**0.8183**	**0.1000**
5_3	0.0000	0.0000	0.0000	**0.0000**	**0.9069**	**0.1000**
5_4	**1.0000**	0.8848	0.9000	**1.0000**	**0.9674**	**1.0000**
5_5	**1.0000**	0.9134	**1.0000**	**1.0000**	**0.9575**	**1.0000**
5_6	**1.0000**	0.9108	**1.0000**	**1.0000**	**0.9727**	**1.0000**
5_7	0.0011	0.0000	0.0000	**1.0000**	**0.9068**	**0.9667**
5_8	**1.0000**	**0.8952**	**0.9333**	**1.0000**	0.8698	0.8667
5_9	0.0029	0.0000	0.0000	**1.0000**	**0.9517**	**1.0000**
5_10	0.5002	0.8888	0.5000	**1.0000**	**0.9296**	**0.8333**

Table 12: Continued.

Inst	d-ABC			n-ABC		
	QMetric	SSpeed	SRate	QMetric	SSpeed	SRate
6_1	0.2335	**0.9274**	0.2333	**0.5008**	0.8686	**0.5000**
6_2	**1.0000**	0.9509	**1.0000**	**1.0000**	**0.9681**	**1.0000**
6_3	**1.0000**	0.9239	0.9000	**1.0000**	**0.9764**	**1.0000**
6_4	**1.0000**	0.9193	**1.0000**	**1.0000**	**0.9807**	**1.0000**
6_5	**1.0000**	0.9206	**1.0000**	**1.0000**	**0.9778**	**1.0000**
a_1	0.0373	**0.0000**	**0.0000**	**0.0517**	**0.0000**	**0.0000**
a_2	0.0006	0.0000	0.0000	**0.2109**	**0.8496**	**0.2000**
a_3	0.0006	0.0000	0.0000	**0.1476**	**0.8056**	**0.1333**
a_4	**0.0306**	0.8544	**0.2333**	**0.0306**	**0.8577**	0.1667
a_5	0.0350	0.0000	0.0000	**1.0000**	**0.9298**	**1.0000**
b_1	0.5415	0.8758	0.4667	**1.0000**	**0.9792**	**1.0000**
b_2	**1.0000**	0.9646	**1.0000**	**1.0000**	**0.9829**	**1.0000**
b_3	**1.0000**	0.9791	**1.0000**	**1.0000**	**0.9824**	**1.0000**
b_4	**1.0000**	0.9682	**1.0000**	**1.0000**	**0.9796**	**1.0000**
b_5	**1.0000**	0.9843	**1.0000**	**1.0000**	**0.9871**	**1.0000**
c_1	0.0075	0.0000	0.0000	**0.0086**	**0.7975**	**0.0333**
c_2	0.5250	0.7752	0.5000	**1.0000**	**0.8852**	**0.9000**
c_3	0.0094	0.0000	0.0000	**0.1069**	**0.5973**	**0.0667**
c_4	0.1301	0.7586	0.0667	**1.0000**	**0.8601**	**0.7667**
c_5	**1.0000**	0.9049	**1.0000**	**1.0000**	**0.9239**	**1.0000**
d_1	**1.0000**	**0.9653**	**1.0000**	**1.0000**	0.9578	**1.0000**
d_2	0.3285	0.0000	0.0000	**1.0000**	**0.9389**	**1.0000**
d_3	0.2665	0.0000	0.0000	**1.0000**	**0.9094**	**0.8333**
d_4	0.2820	0.5737	0.0667	**1.0000**	**0.9716**	**1.0000**
d_5	0.2963	0.9176	0.0333	**1.0000**	**0.9736**	**1.0000**
nre_1	**1.0000**	0.8874	0.8000	**1.0000**	**0.9818**	**1.0000**
nre_2	0.6684	0.0000	0.0000	**0.9381**	**0.8912**	**0.6667**
nre_3	0.6712	0.0000	0.0000	**1.0000**	**0.8703**	**0.8000**
nre_4	0.5555	0.0000	0.0000	**1.0000**	**0.9228**	**0.9667**
nre_5	0.0000	0.0000	0.0000	**1.0000**	**0.9796**	**1.0000**
nrf_1	0.8880	0.0000	0.0000	**1.0000**	**0.9875**	**1.0000**
nrf_2	0.7632	0.0000	0.0000	**1.0000**	**0.9900**	**1.0000**
nrf_3	0.6578	0.0000	0.0000	**1.0000**	**0.9799**	**1.0000**
nrf_4	0.7354	0.0000	0.0000	**1.0000**	**0.9852**	**1.0000**
nrf_5	0.7624	0.0000	0.0000	**0.9441**	**0.9702**	**0.0333**
nrg_1	0.2134	**0.0000**	**0.0000**	**0.4013**	**0.0000**	**0.0000**
nrg_2	0.2100	**0.0000**	**0.0000**	**0.2295**	**0.0000**	**0.0000**
nrg_3	0.1345	**0.0000**	**0.0000**	**0.2122**	**0.0000**	**0.0000**
nrg_4	0.1234	**0.0000**	**0.0000**	**0.2529**	**0.0000**	**0.0000**
nrg_5	0.1245	**0.0000**	**0.0000**	**0.2343**	**0.0000**	**0.0000**
nrh_1	0.1876	**0.0000**	**0.0000**	**0.2342**	**0.0000**	**0.0000**
nrh_2	0.1984	**0.0000**	**0.0000**	**0.2311**	**0.0000**	**0.0000**
nrh_3	0.1976	**0.0000**	**0.0000**	**0.2542**	**0.0000**	**0.0000**
nrh_4	0.1764	**0.0000**	**0.0000**	**0.2103**	**0.0000**	**0.0000**
nrh_5	0.1648	**0.0000**	**0.0000**	**0.2231**	**0.0000**	**0.0000**

the binary encoding is straightforward, where a column takes the value 1 if it is considered in the solution and 0 otherwise) and (ii) the binary Hamming distance is calculated between such binary solution and a reference solution randomly generated, resulting in a new representation of the solution. This representation is used for identifying the different zones explored by the algorithms. Note that the number of zones corresponds with the number of elements in the binary encoding (the number of columns). To this end, all the binary representations are added, resulting in a frequency diagram, showing how usually the algorithm includes a column in a solution.

Table 13 shows three metrics analysing the frequency diagrams previously generated for each ACO approach and an instance from each group 4, 5, 6, a, b, c, d, nre, nrf, nrg, and nrh. Note that the frequency diagrams were generated using all the solutions built in the 30 runs for each algorithm. The metrics used were also proposed by Autuori et al. [72] and are (i) the number of unexplored zones ("uz"), the number of explored zones ("ez"), and the number of large explored zones ("lez"). The metrics are related as follows. Convergence is considered high if lez is few in number. Diversity is considered good if the number of ez is large. Analysing Table 13, we check that ez is usually higher for n-ACO than for d-ACO, meaning that

Table 13: Mapping analysis of the two ACO approaches, when solving a instance from each group.

Inst	n-ACO			d-ACO		
	uz	ez	lez	uz	ez	lez
4_1	51	122	**41**	50	**123**	43
5_1	52	171	**58**	49	**174**	64
6_1	58	174	**46**	51	**181**	50
a_1	109	**297**	**59**	113	293	71
b_1	329	**235**	76	351	213	**60**
c_1	129	**417**	99	150	396	**97**
d_1	569	**299**	81	586	282	**60**
nre_1	2286	164	**91**	2197	**253**	106
nrf_1	4658	145	**81**	4543	**260**	98
nrg_1	1433	915	**172**	728	**1620**	200
nrh_1	4408	602	**140**	3826	**1184**	163

d-ACO improves diversity during the search compared to the traditional approach. This fact is especially relevant for large instances, as occurs for nrg_1 and nrh_1, where ez metric is significantly large. Focusing on lez, we check that the differences observed between both approaches are not as pronounced as for ez. However, such differences could mean that d-ACO has a lack of convergence during the search, and then future authors should manage this fact.

From these two studies solving the traditional SCP, we verify that (i) the concepts from the alternative formulation are useful in guiding the search process of a metaheuristic and (ii) the concepts from the alternative formulation are useful in updating a usual heuristic feasibility operator from the literature. The improvement in both studies was observed in terms of the solution quality, the rate of success, and the speed of reaching a solution. The conclusion in (ii) is especially interesting because this type of operators is generically applied in metaheuristics (generating unfeasible solutions), and the proposal could be directly incorporated into many solving methods.

7. Conclusions and Future Scope

Traditionally, SCP is formulated without addressing two issues: solution unsatisfiability and set redundancy, meaning that the solving method has to implement mechanisms to control such aspects. In recent years, an alternative SCP formulation was proposed, whose main contribution was that both issues were directly addressed by including penalties in the fitness function.

Reviewing the current scientific literature, we check that the alternative SCP formulation has received limited attention. Hence, we question whether there is any advantage of using this formulation beyond addressing set redundancy and feasibility aspects. This idea led us to propose two studies based on a metaheuristic approach. The aim is to identify if there is any concept in the alternative formulation, which could be considered for enhancing a solving method using the traditional formulation. The first study considers an ACO algorithm in two contexts: (i) solving the problem by addressing the traditional SCP formulation and (ii) solving SCP addressing the traditional formulation but using concepts from the alternative one for guiding the search. The second study considers an ABC algorithm in two contexts: (i) solving SCP addressing the traditional formulation and (ii) solving SCP addressing the traditional formulation but including concepts from the alternative one for updating a usual heuristic feasibility operator from the literature.

As a result of the first study, the authors conclude that it is possible to consider the gain concept from the alternative SCP formulation to successfully guide ACO search addressing the traditional SCP formulation. The benefits of the novel guide are shown in terms of solution quality, convergence, execution time, and diversity. From the second study, the authors conclude that it is possible to consider the gain concept from the alternative SCP formulation to update a feasibility operator from the literature. The benefits of the novel feasibility operator are shown in terms of solution quality, execution time, and convergence.

The first conclusion is interesting for designing novel guide strategies for solving the traditional SCP formulation. The second conclusion is especially interesting because feasibility operators are widely considered in metaheuristic approaches solving the SCP because of the usual generation of unfeasible solutions. This type of operators is integrated into the solving method as black-box methods. That means that it is straightforward to interchange one method for another. This situation implies that the feasibility operator based on the alternative SCP formulation presented here could be integrated with a reduced effort in already published works from the literature, as well as in future works to evaluate each specific use case.

As future lines of research, it would be interesting to consider additional metaheuristics to this study, as well as a larger dataset with bigger problems. Additionally, it could be interesting to extend this work by taking into account the performance of the solving methods and the search space complexity of the instances based on the landscape metrics.

Data Availability

The results shown in this paper were obtained by solving some freely available datasets in the literature. They can be found in http://people.brunel.ac.uk/~mastjjb/jeb/orlib/scpinfo.html.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors would like to thank the following grants: Juan A. Gomez-Pulido is supported by grant IB16002 (Junta Extremadura, Spain), Broderick Crawford is supported by grant CONICYT/FONDECYT/REGULAR/1171243, and Ricardo Soto is supported by grant CONICYT/FONDECYT/REGULAR/1190129. The authors also thank "Proyecto CORFO 14ENI2-26905 Nueva Ingeniería para el 2030."

References

[1] R. M. Karp, *Reducibility among Combinatorial Problems*, Springer, Berlin, Germany, 1972.

[2] L. Liang, D. Cool, N. Kakani, G. Wang, H. Ding, and A. Fenster, "Automatic radiofrequency ablation planning for liver tumors with multiple constraints based on set covering," *IEEE Transactions on Medical Imaging*, vol. 39, no. 5, pp. 1459–1471, 2020.

[3] I.-J. Jeong, "An optimal approach for a set covering version of the refueling-station location problem and its application to a diffusion model," *International Journal of Sustainable Transportation*, vol. 11, no. 2, pp. 86–97, 2017.

[4] H. Ye and H. Kim, "Locating healthcare facilities using a network-based covering location problem," *GeoJournal*, vol. 81, no. 6, pp. 875–890, 2016.

[5] A. O. Adewumi and O. J. Adeleke, "A survey of recent advances in vehicle routing problems," *International Journal of System Assurance Engineering and Management*, vol. 9, no. 1, pp. 155–172, 2018.

[6] S. S. V. Vianna, "The set covering problem applied to optimisation of gas detectors in chemical process plants," *Computers & Chemical Engineering*, vol. 121, pp. 388–395, 2019.

[7] V. Chvatal, "A greedy heuristic for the set-covering problem," *Mathematics of Operations Research*, vol. 4, no. 3, pp. 233–235, 1979.

[8] N. Bilal, P. Galinier, and F. Guibault, "A new formulation of the set covering problem for metaheuristic approaches," *ISRN Operations Research*, vol. 2013, pp. 1–10, 2013.

[9] F. J. Vasko, Y. Lu, and K. Zyma, "What is the best greedy-like heuristic for the weighted set covering problem?" *Operations Research Letters*, vol. 44, no. 3, pp. 366–369, 2016.

[10] J. E. Beasley and P. C. Chu, "A genetic algorithm for the set covering problem," *European Journal of Operational Research*, vol. 94, no. 2, pp. 392–404, 1996.

[11] J. E. Beasley, *Or-library*, http://people.brunel.ac.uk/mastjjb/jeb/orlib/scpinfo.html, 2016.

[12] M. Birattari, Z. Yuan, P. Balaprakash, and T. Stützle, "F-race and iterated f-race: an overview," *Experimental Methods for the Analysis of Optimization Algorithms*, Springer, Berlin, Germany, 2010.

[13] B. Crawford, R. Soto, R. Cuesta, and F. Paredes, "Application of the artificial bee colony algorithm for solving the set covering problem," *The Scientific World Journal*, vol. 2014, Article ID 189164, 8 pages, 2014.

[14] M. L. Fisher and P. Kedia, "Optimal solution of set covering/partitioning problems using dual heuristics," *Management Science*, vol. 36, no. 6, pp. 674–688, 1990.

[15] J. E. Beasley and K. Jørnsten, "Enhancing an algorithm for set covering problems," *European Journal of Operational Research*, vol. 58, no. 2, pp. 293–300, 1992.

[16] E. Balas and M. C. Carrera, "A dynamic subgradient-based branch-and-bound procedure for set covering," *Operations Research*, vol. 44, no. 6, pp. 875–890, 1996.

[17] R. Bar-Yejuda and S. Even, "A linear-time approximation algorithm for the weighted vertex cover problem," *Journal of Algorithms*, vol. 2, no. 2, pp. 198–203, 1981.

[18] J. E. Beasley, "An algorithm for set covering problem," *European Journal of Operational Research*, vol. 31, no. 1, pp. 85–93, 1987.

[19] E. El-Darzi and G. Mitra, "Set covering and set partitioning: a collection of test problems," *Omega*, vol. 18, no. 2, pp. 195–201, 1990.

[20] A. Caprara, P. Toth, and M. Fischetti, "Algorithms for the set covering problem," *Annals of Operations Research*, vol. 98, no. 1, pp. 353–371, 2000.

[21] F. J. Vasko, "An efficient heuristic for large set covering problems," *Naval Research Logistics Quarterly*, vol. 31, no. 1, pp. 163–171, 1984.

[22] T. A. Feo and M. G. C. Resende, "A probabilistic heuristic for a computationally difficult set covering problem," *Operations Research Letters*, vol. 8, no. 2, pp. 67–71, 1989.

[23] M. Haouari and J. S. Chaouachi, "A probabilistic greedy search algorithm for combinatorial optimisation with application to the set covering problem," *Journal of the Operational Research Society*, vol. 53, no. 7, pp. 792–799, 2002.

[24] J. H. Ablanedo-Rosas and C. Rego, "Surrogate constraint normalization for the set covering problem," *European Journal of Operational Research*, vol. 205, no. 3, pp. 540–551, 2010.

[25] J. E. Beasley, "A Lagrangian heuristic for set-covering problems," *Naval Research Logistics*, vol. 37, no. 1, pp. 151–164, 1990.

[26] S. Ceria, P. Nobili, and A. Sassano, "A Lagrangian-based heuristic for large-scale set covering problems," *Mathematical Programming*, vol. 81, no. 2, pp. 215–228, 1998.

[27] A. Caprara, M. Fischetti, and P. Toth, "A heuristic method for the set covering problem," *Operations Research*, vol. 47, no. 5, pp. 730–743, 1999.

[28] P. Du and Y. Zhang, "A new distributed approximation algorithm for the maximum weight independent set problem," *Mathematical Problems in Engineering*, vol. 2016, Article ID 9790629, 10 pages, 2016.

[29] R. Soto, E. Rodriguez-Tello, and E. Monfroy, "Recent advances on swarm intelligence for solving complex engineering problems," *Mathematical Problems in Engineering*, vol. 2018, Article ID 5642786, 1 pages, 2018.

[30] X. Zhao, R. Li, and X. Zuo, "Advances on qos-aware web service selection and composition with nature-inspired computing," *CAAI Transactions on Intelligence Technology*, vol. 4, no. 3, pp. 159–174, 2019.

[31] X. Xue, J. Lu, and J. Chen, "Using nsga-iii for optimising biomedical ontology alignment," *CAAI Transactions on Intelligence Technology*, vol. 4, no. 3, pp. 135–141, 2019.

[32] L. R. Rodrigues and J. P. P. Gomes, "Tlbo with variable weights applied to shop scheduling problems," *CAAI Transactions on Intelligence Technology*, vol. 4, no. 3, pp. 148–158, 2019.

[33] H. S. Pannu, D. Singh, and A. K. Malhi, "Improved particle swarm optimization based adaptive neuro-fuzzy inference system for benzene detection," *CLEAN-Soil, Air, Water*, vol. 46, no. 5, Article ID 1700162, 2018.

[34] M. Kaur, V. Kumar, and L. Li, "Color image encryption approach based on memetic differential evolution," *Neural Computing and Applications*, vol. 31, no. 11, pp. 7975–7987, 2019.

[35] P. C. Chu and J. E. Beasley, "A genetic algorithm for the generalised assignment problem," *Computers & Operations Research*, vol. 24, no. 1, pp. 17–23, 1997.

[36] U. Aickelin, "An indirect genetic algorithm for set covering problems," *Computers & Operations Research*, vol. 31, no. 5, pp. 1118–1126, 2004.

[37] M. Deveci and N. Ç. Demirel, "Evolutionary algorithms for solving the airline crew pairing problem," *Computers & Industrial Engineering*, vol. 115, pp. 389–406, 2018.

[38] T. Lust and D. Tuyttens, "Variable and large neighborhood search to solve the multiobjective set covering problem," *Journal of Heuristics*, vol. 20, no. 2, pp. 165–188, 2014.

[39] F. Colombo, R. Cordone, and G. Lulli, "A variable neighborhood search algorithm for the multimode set covering problem," *Journal of Global Optimization*, vol. 63, no. 3, pp. 461–480, 2015.

[40] S. Sundar and A. Singh, "A hybrid heuristic for the set covering problem," *Operational Research*, vol. 12, no. 3, pp. 345–365, 2012.

[41] R. Z. Farahani, A. Hassani, S. M. Mousavi, and M. B. Baygi, "A hybrid artificial bee colony for disruption in a hierarchical maximal covering location problem," *Computers & Industrial Engineering*, vol. 75, pp. 129–141, 2014.

[42] L. Lessing, I. Dumitrescu, and T. Stützle, "A comparison between aco algorithms for the set covering problem," in *Ant Colony Optimization and Swarm Intelligence*, pp. 1–12, Springer, Berlin, Germany, 2004.

[43] Z.-G. Ren, Z.-R. Feng, L.-J. Ke, and Z.-J. Zhang, "New ideas for applying ant colony optimization to the set covering problem," *Computers & Industrial Engineering*, vol. 58, no. 4, pp. 774–784, 2010.

[44] B. Crawford, R. Soto, E. Monfroy, F. Paredes, and W. Palma, "A hybrid ant algorithm for the set covering problem," *International Journal of Physical Science*, vol. 6, no. 19, pp. 4667–4673, 2011.

[45] B. Crawford, R. Soto, M. Olivares-Suarez et al., "A binary coded firefly algorithm that solves the set covering problem," *Romanian Journal of Information Science and Technology*, vol. 17, no. 3, pp. 252–264, 2014.

[46] B. Crawford, R. Soto, M. Riquelme-Leiva et al., "Modified binary firefly algorithms with different transfer functions for solving set covering problems," in *Software Engineering in Intelligent Systems*, pp. 307–315, Springer, Berlin, Germany, 2015.

[47] B. Crawford, R. Soto, F. Aballay, S. Misra, F. Johnson, and F. Paredes, "A teaching-learning-based optimization algorithm for solving set covering problems," in *Computational Science and its Applications-ICCSA 2015*, Springer, Berlin, Germany, 2015.

[48] Y. Lu and F. J. Vasko, "An OR practitioner's solution approach for the set covering problem," *International Journal of Applied Metaheuristic Computing*, vol. 6, no. 4, pp. 1–13, 2015.

[49] Z. Naji-Azimi, P. Toth, and L. Galli, "An electromagnetism metaheuristic for the unicost set covering problem," *European Journal of Operational Research*, vol. 205, no. 2, pp. 290–300, 2010.

[50] R. Soto, B. Crawford, A. Muñoz, F. Johnson, and F. Paredes, "Pre-processing, repairing and transfer functions can help binary electromagnetism-like algorithms," in *Artificial Intelligence Perspectives and Applications*, Springer, Berlin, Germany, 2015a.

[51] B. Crawford, R. Soto, C. Peña et al., "Binarization methods for shuffled frog leaping algorithms that solve set covering problems," in *Software Engineering in Intelligent Systems*, pp. 317–326, Springer, Berlin, Germany, 2015.

[52] B. Crawford, R. Soto, C. Torres-Rojas et al., "A binary fruit fly optimization algorithm to solve the set covering problem," in *Proceedings of the International Conference on Computational Science and its Applications-ICCSA 2015*, pp. 411–420, Saint Petersburg, Russia, July 2015.

[53] R. Soto, B. Crawford, J. Barraza, F. Johnson, and F. Paredes, "Solving pre-processed set covering problems via cuckoo search and lévy flights," in *Proceedings of the 10th Iberian Conference on Information Systems and Technologies (CISTI)*, pp. 1–6, Agueda, Aveiro, Portugal, June 2015.

[54] R. Soto, B. Crawford, R. Olivares et al., "Solving the non-unicost set covering problem by using cuckoo search and black hole optimization," *Natural Computing*, vol. 16, no. 2, pp. 213–229, 2017.

[55] B. Crawford, R. Soto, N. Berríos et al., "A binary cat swarm optimization algorithm for the non-unicost set covering problem," *Mathematical Problems in Engineering*, vol. 2015, Article ID 578541, 8 pages, 2015.

[56] J. M. Lanza-Gutierrez, B. Crawford, R. Soto, N. Berrios, J. A. Gomez-Pulido, and F. Paredes, "Analyzing the effects of binarization techniques when solving the set covering problem through swarm optimization," *Expert Systems with Applications*, vol. 70, pp. 67–82, 2017.

[57] S. Balaji and N. Revathi, "A new approach for solving set covering problem using jumping particle swarm optimization method," *Natural Computing*, vol. 15, no. 3, pp. 503–517, 2016.

[58] R. Soto, B. Crawford, R. Olivares et al., "Adaptive black hole algorithm for solving the set covering problem," *Mathematical Problems in Engineering*, vol. 2018, Article ID 2183214, 23 pages, 2018.

[59] B. Crawford, R. Soto, R. Olivares et al., "A binary monkey search algorithm variation for solving the set covering problem," *Natural Computing*, pp. 1–17, 2019.

[60] N. Bilal, P. Galinier, and F. Guibault, "An iterated-tabu-search heuristic for a variant of the partial set covering problem," *Journal of Heuristics*, vol. 20, no. 2, pp. 143–164, 2014.

[61] B. Crawford, R. Soto, W. Palma, F. Paredes, F. Johnson, and E. Norero, "The impact of a new formulation when solving the set covering problem using the aco metaheuristic," in *Modelling, Computation and Optimization in Information Systems and Management Sciences*, pp. 209–218, Springer, Berlin, Germany, 2015.

[62] B. Crawford and C. Castro, "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, Berlin, Germany, 2006.

[63] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, 1997.

[64] T. Stützle and H. H. Hoos, "Ant system," *Future Generation Computer Systems*, vol. 16, no. 8, pp. 889–914, 2000.

[65] E. Balas, "Cutting planes from conditional bounds: a new approach to set covering," in *Combinatorial Optimization*, pp. 19–36, Springer, Berlin, Germany, 1980.

[66] M. Finger, T. Stützle, and H. Lourenço, "Exploiting fitness distance correlation of set covering problems," in *Workshops on Applications of Evolutionary Computation*, pp. 61–71, Springer, Berlin, Germany, 2002.

[67] M. López-Ibáñez, J. Dubois-Lacoste, L. Pérez Cáceres, M. Birattari, and T. Stützle, "The irace package: iterated racing for automatic algorithm configuration," *Operations Research Perspectives*, vol. 3, pp. 43–58, 2016.

[68] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *The Annals of Mathematical Statistics*, vol. 18, no. 1, pp. 50–60, 1947.

[69] J. Knowles, L. Thiele, and E. Zitzler, "A tutorial on the performance assessment of stochastic multiobjective optimizers," *Computer Engineering and Networks Laboratory (TIK)*, ETH Zurich, Switzerland, 2006.

[70] C. Fonseca, J. Knowles, L. Thiele, and E. Zitzler, "Performance assessment package," 2006, https://sop.tik.ee.ethz.ch/pisa/?page=assessment.php.

[71] K. M. Malan and A. P. Engelbrecht, "Fitness landscape analysis for metaheuristic performance prediction," in *Recent Advances in the Theory and Application of Fitness Landscapes*, pp. 103–132, Springer, Berlin, Germany, 2014.

[72] J. Autuori, F. Hnaien, and F. Yalaoui, "A mapping technique for better solution exploration: NSGA-II adaptation," *Journal of Heuristics*, vol. 22, no. 1, pp. 89–123, 2016.