Journal of Universal Computer Science, vol. 26, no. 2 (2020), 293-316
submitted: 11/12/18, accepted: 5/2/20, appeared: 28/2/20

Ant-Set: A Subset-Oriented Ant Colony Optimization Algorithm for the Set Covering Problem

Murilo Falleiros Lemos Schmitt
(Department of Informatics, Federal University of Paraná
Curitiba, Brazil
muriloschmitt@gmail.com)

Mauro Henrique Mulati
(Computer Science Department, Midwestern State University of Paraná
Guarapuava, Brazil
mhmulati@unicentro.br)

Ademir Aparecido Constantino
(Computer Science Department, State University of Maringá
Maringá, Brazil
ademir@din.uem.br)

Fábio Hernandes
(Computer Science Department, Midwestern State University of Paraná
Guarapuava, Brazil
hernandes@unicentro.br)

Tony Alexander Hild
(Computer Science Department, Midwestern State University of Paraná
Guarapuava, Brazil
thild@unicentro.br)

Abstract: This paper proposes an algorithm for the set covering problem based on the metaheuristic Ant Colony Optimization (ACO) called Ant-Set, which uses a line-oriented approach and a novelty pheromone manipulation based on the connections between components of the construction graph, while also applying a local search. The algorithm is compared with other ACO-based approaches. The results obtained show the effectiveness of the algorithm and the impact of the pheromone manipulation.
Key Words: Ant Colony Optimization, Ant-Set, Set covering problem, Pheromone manipulation, Line-orientation
Category: G.1, G.1.6

1 Introduction

$\mathcal{NP}$-hard combinatorial optimization problems demand exact algorithms in superpolynomial time to obtain an optimal solution, unless $\mathcal{P} = \mathcal{NP}$ [Cormen et al., 2009]. This is the case of the Set Covering Problem (SCP) [Garey and Johnson, 1979], in which given a binary matrix, consists in finding a subset of its columns, where each column has an associated cost, that cover all its lines while minimizing the cost. The SCP is applicable to several practical situations such as crew scheduling problems [Desrochers and Soumis, 1989, Housos and Elmroth, 1997]. In these practical cases, the instance problems can reach huge sizes, making the use of exact algorithms impracticable.

An alternative to obtain satisfactory solutions to $\mathcal{NP}$-hard problems in acceptable time is the use of metaheuristics, such as Ant Colony Optimization (ACO) [Dorigo et al., 2006]. Metaheuristics are a set of algorithmic and data structure concepts to develop and apply heuristic algorithms to satisfactory resolution of $\mathcal{NP}$-hard problems.

In the present paper, we propose the Ant-Set algorithm, an ACO-based algorithm applied to the SCP with two main features: (1) line-orientation, that is, the construction of solutions is based on the selection of uncovered lines by a solution of a SCP instance; and (2) pheromone manipulation associated with the connections between components of the construction graph, which indicates the desirability of including sets of components together to a solution. Ant-Set is also applied with a local search based on the work of [Jacobs and Brusco, 1995] that alters the obtained solution according to a column-cost threshold, in order to improve the solutions obtained by the artificial ants.

The proposed algorithm is compared with other line-orientation ACO-based approaches. The results of the experiments show the effectiveness of the proposed approach, which is capable of obtaining the optimal solutions to several instance classes. The results also suggest that the proposed pheromone manipulation explore more combinations of sets of columns in the solution space.

The rest of the paper is organized as follows: Sections 2, 3 and 4 present the theoretical foundations of this work. Section 5 presents related work. Section 6 presents the proposed approach. Section 7 reports the experiments and results of the proposed approach and comparisons with other ACO-based approaches. Conclusions and future works are presented in the Section 8.

2 The Set Covering Problem

The integer linear optimization problem SCP is a NP-hard problem [Cormen et al., 2009], that belongs to the category of *subset problems*, in which a solution to the problem is given by a subset of available items, subject to the specific constraints of the problem.

The SCP can be defined by a binary $m \times n$ matrix $A = [a_{ij}]$, where each element can be 0 or 1. Each column of the matrix A is associated with a non-negative cost c_j. A column j covers a line i if $a_{ij} = 1$. The objective is to choose a subset of columns - which can be represented by the indexes $s \subseteq \{1, \ldots, n\}$ - minimizing the sum of their costs, in a way that each line is covered by at least one column. The SCP can be formulated according to the Equations (1), (2) and (3):

$$\min \sum_{j=1}^{n} c_j x_j \tag{1}$$

s.t.

$$\sum_{j=1}^{n} a_{ij} x_j \geq 1 \quad i = 1..m, \tag{2}$$

$$x_j \in \{0, 1\} \quad j = 1..n, \tag{3}$$

where the constrains in Equation (2) define that each row must be covered by at least one column, and by Equation (3), the decision variables x_j can assume only the values of 0 or 1. We also define M as the set of all the lines and N as the set of all the columns.

The SCP is important due to the fact that is applicable to several practical situations, such as: urban transit crew scheduling problem [Desrochers and Soumis, 1989], crew scheduling in airlines [Housos and Elmroth, 1997], location of emergency services facilities [Toregas et al., 1971], information retrieval [Al-Sultan et al., 1996], test set compaction [Flores et al., 1999] and location of components of an integrated circuit [Francis and White, 1977].

3 Ant Colony Optimization

Ant Colony Optimization (ACO) is a metaheuristic based on the behavior of ant colonies and has the objective of finding solutions to discrete combinatorial optimization problems. The behavior of different species of ants is based on the indirect communication through a chemical substance called pheromone. While an ant makes a route in search for sources of food, it deposits pheromone on the surface, making a pheromone trail. Other ants smells the deposited pheromone and tend to choose the path with a higher amount of pheromone [Dorigo and Stützle, 2004].

In ACO algorithms, artificial ants cooperate between them in order to obtain good solutions to optimization problems. An artificial ant can be seen as a procedure that constructs a solution iteratively by adding components to a partial solution. The ACO metaheuristic can be applied to any combinatorial optimization problem in which a constructive heuristic can be used [Dorigo and Stützle, 2004].

In ACO algorithms the time is usually discrete and the artificial ants have the capacity of memorizing the constructed path. During the solution construction, the ants guide themselves by the heuristic information and by the existent pheromone on the path.

After the solutions are constructed, it occurs the evaporation of a portion of the current pheromone and, after that, the deposit of pheromone by a subset of ants is made. The generic ACO metaheuristic is presented in Algorithm 1.

Algorithm 1 Generic ACO metaheuristic for static problems [Dorigo and Stützle, 2004]

```
ACO metaheuristic
Initialize parameters
Initialize pheromone trails
while termination condition not met
    Construct ants solutions
    Apply local search  (optional)
    Update pheromone trails
```

The entire process results in a system that exhibits autocatalytic behavior: the artificial ants reinforce the pheromone deposited on the best solutions, resulting in convergence. ACO can be considered simultaneously a metaheuristic with both improving and constructive features, since it constructs new solutions at each iteration using information from previously constructed solutions.

The *Ant System* [Colorni et al., 1991, Dorigo, 1992, Dorigo et al., 1996] is the first proposed ACO algorithm, which was firstly applied to the Travelling Salesman Problem (TSP). After that, many variations were created for different problems, such as the *Max-Min* Ant System (MMAS) [Stutzle and Hoos, 1998], Ant Colony System (ACS) [Dorigo and Gambardella, 1997], a hybrid of MMAS and ACS (MMACS) [Stützle and Hoos, 2000], Approximated Nondeterministic Tree Search (ANTS) [Maniezzo, 1999], beyond many others. There is also the application of ACO to diverse categories of problems, such as SCP [Leguizamon and Michalewicz, 2000], the Multidimensional Knapsack Problem (MKP) [Solnon and Bridge, 2006] and the Quadratic Assignment Problem (QAP) [Maniezzo and Colorni, 1999].

In order to apply ACO algorithms to combinatorial optimization problems, a few important aspects must be considered [Dorigo and Stützle, 2004]: the construction graph, which represents the paths where the artificial ants walk; the constraints of the problem; the definition of the pheromone trails and heuristic information; the way the solutions are built; the pheromone update; and when required, the local search.

4 The JB Local Search

The JB Local Search used in this paper is an adaptation of a local search scheme called *SEARCH Module* presented in [Jacobs and Brusco, 1995]. In that work, the authors propose a heuristic algorithm for the SCP based on Simulated Annealing (SA), which uses the SEARCH Module to find a neighbor solution required by the main algorithm.

The SEARCH Module consists of removing a number of columns from the solution, based on a cost threshold obtained through the multiplication of the total number of columns in the current solution by the parameter ρ_1^{JB}. Columns which exceed the cost threshold are removed from the current solution. This action will likely result in several lines being uncovered. The algorithm then performs the covering of the newly uncovered lines according to a criteria based on the cost of the worst column on the solution multiplied by the parameter ρ_2^{JB}. The result of this multiplication is the maximum cost allowed to a column to be included. The feasibility of the solution is regained by adding candidate columns based on its cover cost.

5 Related Work

One of the first papers about the use of heuristic algorithms to SCP was the work of [Chvatal, 1979], that implemented a greedy algorithm. [Balas and Ho, 1980] also developed a greedy algorithm, based on five new greedy functions. The paper of [Vasko, 1984] explored the heuristics of [Chvatal, 1979] and [Balas and Ho, 1980], with new greedy functions and a procedure to remove redundant columns of a feasible solution.

The SCP has been explored with several different approaches. In [Feo and Resende, 1989], the authors proposed a non-deterministic variation of the [Chvatal, 1979] heuristic where a local search was developed. [Beasley, 1990a] and [Haddadi, 1997] developed algorithms based on Lagrangian heuristics. The work of [Jacobs and Brusco, 1994] implemented a Simulated Annealing algorithm. [Beasley and Chu, 1996], [Al-Sultan et al., 1996] and [Aickelin, 2002] developed procedures based on Genetic Algorithms. [Yagiura et al., 2006] developed a 3-flip neighborhood local search, with the application of Lagrangian relaxation to reduce the problem size through the use of the subgradient method. In [Sundar and Singh, 2012], the authors proposed an algorithm based on the Artificial Bee Colony metaheuristic, with the application of a local search based on the work by [Ren et al., 2010] to improve the quality of the solutions. The authors compared their results with several different approaches. The work of [Lanza-Gutierrez et al., 2017] analysed the effects of binarization methods of metaheuristics. To that end, the authors implemented a Binary Cat Swarm Optimization algorithm for the SCP, and applied several different binarizarion approaches. For comprehensive reviews to the SCP, we refer the reader to the works of [Caprara et al., 2000] and [Farahani et al., 2012].

Several ACO-based algorithms were proposed to the SCP. The initial ones followed the Ant System algorithm guidelines, proposed by [Leguizamon and Michalewicz, 2000] and [Hadji et al., 2000], as showed by [Dorigo and Stützle, 2004]. The results presented by [Leguizamon and Michalewicz, 2000] and [Hadji et al., 2000] are not competitive with other approaches such as of [Caprara et al., 1999] and [Marchiori and Steenbeek, 2000]. Another study derived from [Hadji et al., 2000] is presented in the work of [Rahoual et al., 2002].

An application of the MMAS to the SCP is made by [Lessing et al., 2004], comparing the application of the ACS, MMACS and ANTS to the SCP with seven types of heuristic information. They also utilized a local search based on [Yagiura et al., 2006]. The best results were obtained with the heuristic information that uses Lagrangian relaxation [Fisher, 1981].

The work of [Aickelin, 2002] suggests choosing columns to the partial solution through *line-orientation*. In that sense, [Ren et al., 2008] proposed the ACO-based algorithm Ant-Cover (AC). In the work of [Ren et al., 2010], the AC with a local search is compared with other metaheuristics approaches and with an ACO approach from [Crawford and Castro, 2006], with the AC obtaining better results. In [Mulati and Constantino, 2011], the authors proposed the *Ant-Line* algorithm, an ACO-based algorithm with line-orientation that uses a greedy heuristic information [Balas and Ho, 1980] with the application of a local search based on the work of [Jacobs and Brusco, 1995]. The authors compared the Ant-Line with the Ant-Cover, obtaining competitive results. In [Al-Shihabi et al., 2015] the authors proposed the LP-MMAS-LS algorithm. The LP-MMAS-LS is a hybrid of the MMAS [Stützle and Hoos, 2000], with the construction graph and the local search used by [Ren et al., 2010], while also using information from Lagrangian relaxation to reduce the size of the problem. The LP-MMAS-LS is compared with the approaches of [Ren et al., 2010] and [Sundar and Singh, 2012], obtaining superior results.

6 Proposed Method

In this section, we present the proposed method, the Ant-Set algorithm. The algorithm is based on the Ant-Line algorithm, proposed by [Mulati and Constantino, 2011], which constructs its solution based on line-orientation. We first present an overview of the Ant-Line, followed by the Ant-Set.

6.1 Ant-Line

The *Ant-Line* [Mulati and Constantino, 2011] is an ACO based algorithm applied to the SCP. The main novelty of the algorithm is a feature called *line-orientation*, which differs from other ACO approaches since instead of constructing solutions by adding sets of columns to the partial solution, the Ant-Line first chooses a line uncovered by the current solution, and after that, chooses a column to cover this specific line.

In the Ant-Line, the columns of SCP are mapped as the solution components. All these components are in the set C and composes the vertices of the construction graph, which is a complete graph. The pheromone value of a component j is denoted by τ_j and indicates some desirability developed by the ants of including j in a solution, thus the ants deposit and consult pheromone over a component.

Each ant k is guided by a value called *information choice*: this value is given by $\tau_j^\alpha \eta_j^\beta$, which indicates the pheromone trails and heuristic information combined desirability of including the component j in the solution s_k, where α and β are parameters that controls the importance of pheromone (τ_j) and heuristic information (η_j), respectively. The heuristic information used by the Ant-Line is the dynamic heuristic information *cover cost* (CC) [Balas and Ho, 1980]. CC is related to each column, and is given by Equation (4):

$$\eta_j = \frac{card_j(s)}{c_j} \tag{4}$$

where c_j is the cost of column j, $card_j(s)$ is the cardinality of the column j, that is, the number of lines that are covered by the column j but are not covered by any column in the partial solution s.

In a construction step, the Ant-Line takes into account information beyond the information choice. Each ant k is a constructive method that at each step: (1) randomly chooses a line e which is not covered yet by any column in the partial solution and (2) chooses a column to cover this line by using a deterministic decision rule based on the information choice.

The probability of selecting a line e is given by a uniform distribution, as defined in Equation (5):

$$p_e^k = \begin{cases} \dfrac{1}{|M \setminus R(s_k)|} & if\, e \notin R(s_k) \\ 0 & if\, e \in R(s_k) \end{cases} \qquad \forall e \in M \tag{5}$$

where $R(s_k)$ is the set of all lines that are already covered by a column in s_k. Following the selection of a line e, the algorithm defines a set of candidate components, given by the Equation (6):

$$N(e, s_k) = \{j \in C\colon j \notin s_k,\ e \notin R(s_k),\ a_{ej} = 1\} \tag{6}$$

The algorithm then deterministically selects a component from $N(e, s_k)$ following the Equation (7):

$$j = \arg\max_{h \in N(e,s_k)} \{\tau_h^{\alpha} \eta_h^{\beta}\} \tag{7}$$

The solution constructed by an artificial ant k repeats this process until the feasibility of the solution s_k is achieved. After the construction of the solution, a local search can be applied, followed by the elimination of redundant components.

After the construction of feasible solutions by all the ants, the pheromone trails are updated by respectively evaporation and deposit operations. The evaporation is done according to Equation (8):

$$\tau_j = (1 - \rho)\tau_j \quad \forall j \in C \tag{8}$$

where ρ is the evaporation rate, a parameter of the algorithm.

Following the evaporation, the deposit of pheromone is performed. The solution to be reinforced is chosen between the best solution made by an artificial ant in the current iteration (s') or the best solution so far made by an ant (s^*). Initially, s' is reinforced more often than s^*. A gradual exchange on this frequency is done based on the maximum number of iterations of the algorithm. Assuming that the value of s is already selected between s' and s^*, the deposit of pheromone is done by only one ant, according to the Equation (9):

$$\tau_j = \tau_j + \left(\frac{f(s^*)}{f(s)}\right)^y \quad \forall j \in s \tag{9}$$

where y is a parameter which regulates the value to be deposited in the components of the solution.

6.2 Ant-Set

Different ways to manipulate pheromone for the SCP are proposed in [Mulati, 2009]. One of the proposed manipulation is the use of consult, evaporation and deposit operations of pheromone on the connections (relationship) between all column pairs of a solution of the problem. In the present paper, we apply this pheromone manipulation and use the Ant-Line constructive procedure, resulting in the *Ant-Set* algorithm.

In the Ant-Set, the pheromone trails are associated with the edges (connections between components) of the constructive graph, which indicates the desirability of including sets of components together in the solution.

The Ant-Set presents similar behavior to the Ant-Line, with differences on the pheromone manipulation, and consequently in the construction of the solutions by the artificial ants. The manipulation of pheromone between all column pairs results in each existent column in the partial solution under construction having direct influence in the evaluation of the candidate columns, and consequently in the selection of the next column to be added to the solution.

In the Ant-Set decision rule, after selecting the line, according to Equation (5), the artificial ant must evaluate each candidate column considering the existent pheromone in the connections (edges) between all columns existent in the partial solution and the candidate column. In the first iteration of the algorithm, given that the pheromone deposited between all connections is the same, the first column to be added to the solution is selected randomly. The pheromone consult operation is defined according to Equation (10):

$$\tau_{s_k j} = \sum_{i \in s_k} \tau_{ij} \tag{10}$$

where j is the candidate column and s_k is the partial solution under construction by the ant k.

The pheromone evaporation in the Ant-Set is made over the connections between all existing column pairs in the instance of the problem, as given in Equation (11):

$$\tau_{ij} = (1-\rho)\tau_{ij} \quad \forall i, j \in C \tag{11}$$

In the pheromone deposit, each artificial ant assigned to effectuate the deposit must deposit pheromone in all the connections between the existent columns in the obtained solution, as defined in Equation (12):

$$\tau_{ij} = \tau_{ij} + \left(\frac{f(s^*)}{f(s)}\right)^y \quad \forall i, j \in s \tag{12}$$

The pheromone manipulation has a direct impact on the way of choosing a component to a partial solution. In the Ant-Line, the pheromone deposit is made on the solution columns, that is, the algorithm evaluate the desirability of adding a specific component to the solution. Conversely, the manipulation of pheromone between connections of all solution column pairs refers to the desirability of choosing certain columns together to the same solution. A visualization of the pheromone deposits for algorithms Ant-Line and Ant-Set, respectively, is presented in Figure 1.

The Ant-Set is presented in Algorithm 2. The algorithm has three stopping criteria as stated in the *while* loop in line 4: maximum time (mt), maximum number of iterations (mi) and maximum number of iterations without improving the solution s^*, which is defined by the procedure ReachMaxItsWithoutImprov() and the parameter $miwi$. The *for* loop in the lines 6-15 makes the artificial ants construct their solutions considering the number of ants defined by the parameter $nants$, following the Equations (5), (6), (7) and (10).

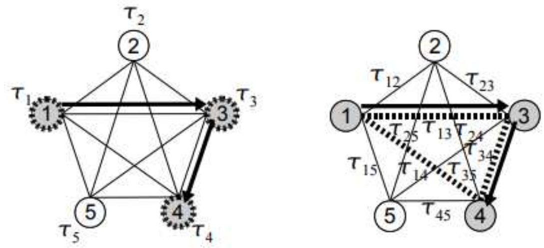

Figure 1: Pheromone deposit operation for the Ant-Line and Ant-Set algorithms, respectively. In the Ant-Line, the pheromone is deposited on the vertices of a solution in the construction graph, while in the Ant-Set the pheromone is deposited on all the edges among the vertices of a solution [Mulati, 2009].

The rest of the algorithm, defined in the lines 16-21, refers to the pheromone manipulation, following Equations (11) and (12), and the selection of the best found solution. The procedure ChooseByFrequency(), presented in the line 19, selects the solution to be reinforced according to the Equation (12).

The procedure uses the parameter *nich*, which indicates the number of iterations of an interval used to change the updating solution. The value of s is selected as follows. In the first interval, s' is used for all the iterations, while s^* is not used. At each succeeding interval, the number of iterations to use s^* is incremented by 1 and the number of iterations in the interval to use s' is decremented by 1, and this rule runs until the end of the execution.

7 Experiments and Results

In this section, we report the experiments carried with the Ant-Set, the methodology adopted and the results obtained. We compare the results of the Ant-Set with other ACO-based approaches and also present a discussion based on the results.

7.1 Methodology

The Ant-Set was implemented in C++11 language and the experiments were carried on a computer Intel Core i7-4770 of 3.4 GHz with 16 GB of main memory running Ubuntu 16.04 as the operational system.

The experiments were carried with 70 SCP instances from OR-Library[1] [Beasley, 1990b]. Information regarding the instances are presented in Table

[1] `http://people.brunel.ac.uk/~mastjjb/jeb/orlib/scpinfo.html`, accessed in August, 2018

Algorithm 2 Ant-Set

AntSet(*scp-instance*, *parameters*)

$\tau_j = 1 \quad \forall j \in C$
$f(s^*) = \infty$
$i = 0$
while $i < mi$ and not ReachMaxItsWithoutImprov($i, s^*, miwi$) and TimeElapsed() $< mt$
  $f(s') = \infty$
  for $k = 1$ **to** *nants*
    $s_k = \varnothing$
    while s_k is not feasible
      $e =$ Randomly choose a line which is not covered by any column in the s_k, as in Eq. (5)
      $j = \arg\max_{h \in N(e, s_k)}\{\tau_h^{\alpha}\eta_h^{\beta}\}$, where $N(e,\ s_k)$ contain all the columns that cover the line e, except the ones in s_k, as in Eqs. (6), (7) and (10)
      $s_k = s_k \cup \{j\}$
    ApplyJbLocalSearch(s_k, ρ_1^{JB}, ρ_2^{JB})
    EliminateRedundantColumns(s_k)
    if $f(s_k) < f(s')$
      $s' = s_k$
  if $f(s') < f(s^*)$
    $s^* = s'$
  $\tau_{ij} = (1-\rho)\tau_{ij} \quad \forall i, j \in C$ (as in Eq. (11))
  $s =$ ChooseByFrequency(i, *nich*, s', s^*)
  $\tau_{ij} = \tau_{ij} + \left(\frac{f(s^*)}{f(s)}\right)^y \quad \forall i, j \in s$ (as in Eq. (12))
  $i = i + 1$
return s^*

1. The first column (C) indicates the instance classes, where the SCP *4* and *5* classes contains 10 instances each and the remaining classes contains 5 instances. The second column (OBK of SCP Instances) indicates the optimal or best known solution (OBK) for each instance of the class, where values in bold indicates that a optimal solution is known, according to [Ren et al., 2010]. The third column shows the sizes of the instances ($m \times n$) and the last column presents the density of each instance in percentage (D%). We perform pre-processing to reduce a SCP instance in the same way as described in [de Oliveira, 1999].

For each instance, the algorithm was executed 10 times, each time with a different random seed. From these 10 executions, for each instance, we extract the mean time of execution, the mean quality of the solutions, the value of the worst found solution, the number of iterations taken to find the first best solution, the time taken to find the first best solution and the value of the best found solution.

Table 1: Classes of SCP instances

C	OBK of SCP Instances										$m \times n$	D
	1	2	3	4	5	6	7	8	9	10		%
4	**429**	**512**	**516**	**494**	**512**	**560**	**430**	**492**	**641**	**514**	200 × 1000	2
5	**253**	**302**	**226**	**242**	**211**	**213**	**293**	**288**	**279**	**265**	200 × 2000	2
6	**138**	**146**	**145**	**131**	**161**						200 × 1000	5
a	**253**	**252**	**232**	**234**	**236**						300 × 3000	2
b	**69**	**76**	**80**	**79**	**72**						300 × 3000	5
c	**227**	**219**	**243**	**219**	**215**						400 × 4000	2
d	**60**	**66**	**72**	**62**	**61**						400 × 4000	5
e	**5**	**5**	**5**	**5**	**5**						50 × 500	20
nre	29	30	27	28	28						500 × 5000	10
nrf	14	15	14	14	13						500 × 5000	20
nrg	176	154	166	168	168						1000 × 10000	2
nrh	63	63	59	58	55						1000 × 10000	5

In [Mulati and Constantino, 2011], the parameters $nants$, α, β, ρ, and y are fixed and based on the literature as well as on some preliminary experiments, while calibration is done over the parameters related to iterations and times. In the present paper, we calibrate and use the same set of parameters for every SCP instance. We set the maximum time of execution (mt) to 1800 seconds and the maximum number of iterations (mi) to 900. The remaining parameters were tuned through the irace package [López-Ibáñez et al., 2016] on a subset of SCP instances. The parameters number of ants ($nants$), importance of pheromone (α), importance of heuristic information (β), evaporation rate (ρ), pheromone deposit rate (y), number of iterations of an interval to change the updating solution ($nich$) and maximum number of iterations without improvement ($miwi$) were set to be tuned over the variations that follows:

- $nants \in \{4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48, 52, 56, 60, 64\}$;
- $\alpha \in [1, 16]$;
- $\beta \in [1, 16]$;

- $\rho \in [0.1, 0.9]$;
- $y \in [2, 4]$;
- $nich \in [5, 30]$;
- $miwi \in \{5, 7, 10, 13, 20, 29, 39, 58, 80, 115, 180\}$.

The parameters that achieved the best results are: $nants = 60$, $\alpha = 1$, $\beta = 14$, $\rho = 0.2$, $y = 2$, $nich = 16$ and $miwi = 115$. For the JB local search, we executed experiments with the four combinations of parameters presented in [Jacobs and Brusco, 1994] and found that the best results were obtained with $\rho_1^{JB} = 0.4$ and $\rho_2^{JB} = 1.1$.

7.2 Results

The results obtained in the experiments with the Ant-Set without and with local search, respectively, are presented in Table 2. Each row presents the results for one instance class, where the results are averaged by the number of instances from the given instance class. The first column of Table 2 contains the SCP instance class (C). The second column (#) presents the number of instances from the referred instance class. The columns ATT, WS and AS contains summarized data from the 10 trials executed over each instance from the class, where ATT is the average time in seconds, WS is the distance between the worst solutions and the OBK in percentage, and AS is the distance between the average solutions and the OBK in percentage.

The remaining columns, IFB, TFB and BS contains information from the best trial of each run. IFB is the number of iterations to find the first best solution, TFB is the time taken, in seconds, to find the first best solution and BS is the percentage distance between the best solutions and the OBK.

The results presented in Table 2 are grouped by instance class. The detailed results obtained with the Ant-Set for the 70 SCP instances are presented in Appendix A on Tables 5, 6 and 7.

7.2.1 Ant-Set

Observing the results presented in Table 2 it is possible to see that, without the application of the JB local search, the Ant-Set obtained results in average (AS%) 0.63% distant from the OBK solutions. Considering the worst obtained solutions in the runs (WS%), the results are 1.19% distant from the optimal solutions, while the best obtained solutions (BS%) are 0.24% distant from the optimal solutions. The Ant-Set algorithm obtained the optimal solutions for 8 of the 12 SCP instance classes, with an average execution time of 5.8 seconds.

Table 2: Results by Ant-Set without and with JB local search grouped by SCP class.

		Ant-Set (without JB)						*Ant-Set+JB*					
C	#	ATT s	WS %	AS %	IFB	TFB s	BS %	ATT s	WS %	AS %	IFB	TFB s	BS %
4	10	0.5	0.22	0.06	31	0.1	**0**	9.4	0.23	0.10	26	1.6	0.02
5	10	0.9	0.31	0.09	31.5	0.2	**0**	16.3	0.35	0.08	21.2	2.0	**0**
6	10	0.6	0.56	0.14	11.4	0.0	**0**	4.7	**0**	**0**	8.8	0.3	**0**
a	5	2.0	0.65	0.40	30.8	0.4	0.25	37.5	0.49	0.21	45.8	10.7	**0**
b	5	2.0	0.25	0.02	3.0	0.0	**0**	14.1	**0**	**0**	1.6	0.2	**0**
c	5	3.8	0.97	0.25	54.0	1.0	**0**	69.9	0.26	0.12	58.4	21.5	**0**
d	5	3.5	0.91	0.33	7.8	0.2	**0**	25.8	0.58	0.09	6.8	1.2	**0**
e	5	0.2	**0**	**0**	1.0	0.0	**0**	0.5	**0**	**0**	1.0	0.0	**0**
nre	5	6.9	2.12	0.93	3.2	0.2	**0**	22.5	**0**	**0**	2.2	0.3	**0**
nrf	5	11.5	4.40	2.40	5.0	0.4	1.54	21.3	1.54	0.15	7.0	1.1	**0**
nrg	5	17.7	1.21	0.80	53.0	5.1	0.49	396.6	1.22	0.52	98.8	163.0	0.25
nrh	5	20.6	2.64	2.13	44.0	6.4	0.63	131.6	2.95	1.77	34.6	27.3	1.31
		5.8	1.19	0.63	23.0	1.2	0.24	62.5	0.64	0.25	26.0	19.1	0.13

The application of the JB local search improves significantly the obtained solutions in terms of solution quality. Considering the average, worst and best solutions, the distances to the optimal solutions decreased to 0.25%, 0.64% and 0.13% respectively. The Ant-Set with the application of the JB local search was also able to obtain the optimal solutions for more instance classes, finding the optimal solutions for 9 of the 12 SCP instance classes, however, for the *SCP 4* class, the algorithm lost the optimality for the solution of the best trial. While the application of the JB procedure resulted in improvements to the solutions, there was a high increase in execution time, with the algorithm taking in average 62.5 seconds to run, although that can be mainly attributed to the *nrg* class, which took in average 396.6 seconds to execute.

It is worth noting that, for the Ant-Set with and without the application of the JB local search, the stopping criteria was always number of iterations without improvement (*miwi* parameter). In terms of iterations to find the first best solution, considering the best trial of the run, the Ant-Set with and without the JB procedure required in average almost the same number of iterations: 23 and 26, respectively.

7.2.2 Comparison with Ant-Line

In order to compare the Ant-Set with the Ant-Line, we executed experiments with the Ant-Line algorithm using the same methodology as described in Section 7.1, also setting the parameters mt to 1800 seconds and mi to 900, and using irace [López-Ibáñez et al., 2016] to select the remaining parameters. The parameters that obtained the best results for the Ant-Line are: $nants = 64$, $\alpha = 1$, $\beta = 15$, $\rho = 0.1$, $y = 2$, $nich = 11$ and $miwi = 115$. The results obtained by the Ant-Line without and with the application of the JB procedure, grouped by instance classes, are presented in Table 3.

Table 3: Results by Ant-Line without and with JB local search grouped by SCP class.

		Ant-Line (without JB)						*Ant-Line+JB*					
C	#	ATT s	WS %	AS %	IFB	TFB s	BS %	ATT s	WS %	AS %	IFB	TFB s	BS %
4	10	0.5	0.07	0.02	32.4	0.1	**0**	5.7	0.13	0.06	32.0	1.3	**0**
5	10	0.7	0.11	0.04	20.9	0.1	0.03	5.8	0.26	0.04	23.5	0.8	**0**
6	10	0.5	**0**	**0**	7.4	0.0	**0**	1.9	**0**	**0**	3.6	0.1	**0**
a	5	1.3	0.41	0.26	79.8	0.6	**0**	9.7	0.33	0.27	44.6	2.8	0.17
b	5	1.4	**0**	**0**	1.6	0.0	**0**	3.7	**0**	**0**	1.2	0.0	**0**
c	5	2.3	0.35	0.12	61.4	0.7	**0**	16.2	0.26	0.11	46.6	4.2	0.08
d	5	2.6	0.61	0.27	7.4	0.1	**0**	6.5	0.61	0.06	4.8	0.2	**0**
e	5	0.2	**0**	**0**	1.0	0.0	**0**	0.3	**0**	**0**	1.0	0.0	**0**
nre	5	5.4	2.12	0.77	1.4	0.1	**0**	10.1	0.74	0.07	2.0	0.2	**0**
nrf	5	9.2	2.97	2.40	2.8	0.2	1.54	15.9	2.97	0.45	2.6	0.3	**0**
nrg	5	9.1	1.22	0.76	64.0	3.1	0.49	61.9	1.11	0.56	86.0	22.7	0.13
nrh	5	13.4	2.64	1.77	26.4	2.6	0.63	32.4	2.29	1.64	23.0	4.9	1.31
		3.9	0.88	0.53	25.5	0.6	0.22	14.2	0.72	0.27	22.6	3.1	0.14

The Ant-Line algorithm without the JB procedure, obtained results in average 0.53% distant from the OBK solutions, with an average execution time of 3.9 seconds. Considering the worst obtained solutions, the results are 0.88% away from the OBK solutions, while the best solutions are 0.22% distant from the OBK solutions. The Ant-Line with the application of the JB local search improved the quality of the solutions, obtaining in terms of average, worst and best solutions 0.27%, 0.72% and 0.14% distant from the OBK solutions, respectively, with an average time of 14.2 seconds.

Comparing the results presented in Tables 2 and 3, without the JB local search, it can be seen that the Ant-Line outperforms the Ant-Set. The Ant-Line algorithm obtained better solutions in terms of quality of the worst found solution, and slightly better results in terms of average quality and quality of the best found solution. Considering the best found solutions, the Ant-Line obtained slightly better results, while requiring in average more iterations to found the best solutions (25.5). Both algorithms obtained the optimal solutions to 8 of the 12 instance classes. In terms of execution time, the Ant-Line outperforms the Ant-Set, with average times of execution of 3.9 and 5.8 seconds respectively, although it is important to note that the computational complexity of the pheromone operations is different for both algorithms.

The application of the JB local search improved the results of both algorithms, with the Ant-Set obtaining the best results in terms of averaged solution quality, quality of the worst found solution and quality of the best found solution. Considering the best found solutions, the Ant-Set obtained the optimal solutions to 9 of the 12 instance classes, while the Ant-Line obtained the optimal solutions for 8 of the 12 classes. The main issue with the Ant-Set algorithm is execution time: the Ant-Set is considerably slower, requiring in average 62.5 seconds to execute, but that can be partially attributed to the *nrg* and *nrh* classes, which require a high number of iterations to find the best solutions. The higher amount of iterations required are associated with the pheromone manipulation, which suggests that the pheromone manipulation based on the desirability of including subsets of columns together to the solution results in a more diversified exploration of the solution space.

7.2.3 Comparison with State-of-the-Art

Although there are other metaheuristics that present interesting results in terms of solution quality and execution time [Sundar and Singh, 2012], our major interest is comparing the Ant-Set with other ACO approaches. The current best ACO algorithm to the SCP is the LP-MMAS-LS, proposed by [Al-Shihabi et al., 2015]. The LP-MMAS-LS is a hybrid of the MMAS [Stützle and Hoos, 2000], with the construction graph and the local search used by [Ren et al., 2010], while also using information from Lagrangian relaxation to reduce the size of the problem.

Table 4 presents results by the Ant-Set with the JB local search and the LP-MMAS-LS, containing the average time of execution in seconds (ATT), the distance between the average solutions and the OBK (AS) and the distance between the best solutions and the OBK (BS), grouped by instance classes. The information regarding the Ant-Set was extracted from Table 2, excluding the SCP e instance class. The experiments for the LP-MMAS-LS were conducted on a computer Intel Core i5-3210M of 2.5 GHz with 8 GB of main memory.

As reported in Table 4, the LP-MMAS-LS outperforms the Ant-Set. LP-MMAS-LS is the first ACO algorithm to obtain the optimal solution for all 65 instances [Al-Shihabi et al., 2015]. In comparison, the Ant-Set with JB local search is able to obtain the optimal solution for 59 of the 65 instances, with the best solutions being 0.14% distant from the optimal solutions.

In terms of average solution, the results are competitive, with the LP-MMAS-LS obtaining results 0.22% distant from the OBK solutions and the Ant-Set obtaining results 0.28% distant from the OBK solutions. Considering the average time of execution, the LP-MMAS-LS significantly outperforms the Ant-Set. The high amount of time required by the Ant-Set can be attributed to the JB local search, while the low amount of time required by the LP-MMAS-LS is related to the reduction in the number of columns.

Table 4: Results by Ant-Set with JB local search and LP-MMAS-LS [Al-Shihabi et al., 2015] grouped by SCP class.

		Ant-Set+JB			LP-MMAS-LS		
C	#	ATT	AS	BS	ATT	AS	BS
		s	%	%	s	%	%
4	10	9.4	0.10	0.02	1.0	**0**	**0**
5	10	16.3	0.08	**0**	1.1	**0**	**0**
6	10	4.7	**0**	**0**	1.0	**0**	**0**
a	5	37.5	0.21	**0**	2.1	0.17	**0**
b	5	14.1	**0**	**0**	1.0	**0**	**0**
c	5	69.9	0.12	**0**	3.3	0.03	**0**
d	5	25.8	0.09	**0**	1.1	**0**	**0**
nre	5	22.5	**0**	**0**	1.3	**0**	**0**
nrf	5	21.3	0.15	**0**	1.3	0.46	**0**
nrg	5	396.6	0.52	0.25	11.1	0.50	**0**
nrh	5	131.6	1.77	1.31	6.8	1.25	**0**
		68.2	0.28	0.14	2.8	0.22	**0**

8 Conclusions and Future Work

In this paper, we proposed an ACO-based algorithm to solve the SCP problem called Ant-Set. The algorithm is based on line-orientation, that is, the construction of solutions is firstly based on the uncovered lines, while it also uses a novelty way to manipulate pheromone, where the pheromone trails are associated with the connections between components of a constructive graph, which indicates the desirability of including sets of components together to the solution.

We performed experiments with 70 instances from OR-Library, while also applying a local search in order to improve the solutions found by the artificial ants. The results suggests that the proposed pheromone manipulation is capable of exploring different combinations in the solution space, being able to obtain optimal solutions for more instances classes than the Ant-Line algorithm and also improving the quality of the obtained solutions.

Future works will consist in incorporate maximum and minimum limits to the pheromone values, improving the heuristic information, including SCP and linear programming specific techniques, the application of different local searches, adaptive adjustment of parameters during the execution of the algorithm, and parallel implementations of the Ant-Set algorithm in order to improve execution time.

Acknowledgments

The authors would like to thank Fundação Araucária–PR for the financial support.

References

[Aickelin, 2002] Aickelin, U. (2002). An indirect genetic algorithm for set covering problems. *Journal of the Operational Research Society*, pages 1118–1126.

[Al-Shihabi et al., 2015] Al-Shihabi, S., Arafeh, M., and Barghash, M. (2015). An improved hybrid algorithm for the set covering problem. *Computers & Industrial Engineering*, 85:328–334.

[Al-Sultan et al., 1996] Al-Sultan, K., Hussain, M., and Nizami, J. (1996). A genetic algorithm for the set covering problem. *Journal of the Operational Research Society*, pages 702–709.

[Balas and Ho, 1980] Balas, E. and Ho, A. (1980). Set covering algorithms using cutting planes, heuristics, and subgradient optimization: A computational study. *Math. Program. Study*, 12:37–60.

[Beasley, 1990a] Beasley, J. (1990a). A lagrangian heuristic for set-covering problems. *Naval Research Logistics (NRL)*, 37(1):151–164.

[Beasley, 1990b] Beasley, J. (1990b). OR-Library: distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41(11):1069–1072.

[Beasley and Chu, 1996] Beasley, J. and Chu, P. (1996). A genetic algorithm for the set covering problem. *European Journal of Operational Research*, 94(2):392–404.

[Caprara et al., 1999] Caprara, A., Fischetti, M., and Toth, P. (1999). A Heuristic Method for the Set Covering Problem. *Operations Research*, 47(5):730–743.

[Caprara et al., 2000] Caprara, A., Toth, P., and Fischetti, M. (2000). Algorithms for the set covering problem. *Annals of Operations Research*, 98(1):353–371.

[Chvatal, 1979] Chvatal, V. (1979). A greedy heuristic for the set-covering problem. *Mathematics of operations research*, pages 233–235.

[Colorni et al., 1991] Colorni, A., Dorigo, M., Maniezzo, V., et al. (1991). Distributed optimization by ant colonies. In *Proceedings of the First European Conference on Artificial Life*, pages 134–142. Paris, France: Elsevier Publishing.

[Cormen et al., 2009] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to Algorithms*. MIT Press, Cambridge, Massachusetts, 3rd edition.

[Cormen et al., 2009] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to Algorithms, Third Edition.* MIT Press, 3rd edition.

[Crawford and Castro, 2006] Crawford, B. and Castro, C. (2006). Integrating lookahead and post processing procedures with aco for solving set partitioning and covering problems. In *International Conference on Artificial Intelligence and Soft Computing*, pages 1082–1090. Springer.

[de Oliveira, 1999] de Oliveira, N. V. (1999). Problema de Cobertura de Conjuntos – Uma Comparação Numérica de Algoritmos Heurísticos. Master's thesis, Universidade Federal de Santa Catarina, Programa de Pós-Graduação em Engenharia de Produção.

[Desrochers and Soumis, 1989] Desrochers, M. and Soumis, F. (1989). A column generation approach to the urban transit crew scheduling problem. *Transportation science*, 23(1):1–13.

[Dorigo, 1992] Dorigo, M. (1992). *Optimization, Learning and Natural Algorithms.* PhD thesis, Ph. D. thesis, Politecnico di Milano IT, Dipartimento di Elettronica ed Informatica.

[Dorigo et al., 2006] Dorigo, M., Birattari, M., and Stützle, T. (2006). Ant Colony Optimization - Artificial Ants as a Computational Intelligence Technique. *IEEE Computational Intelligence Magazine*, 1(4):28–39.

[Dorigo and Gambardella, 1997] Dorigo, M. and Gambardella, L. M. (1997). Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Trans. Evolutionary Computation*, 1(1):53–66.

[Dorigo et al., 1996] Dorigo, M., Maniezzo, V., and Colorni, A. (1996). Ant system: optimization by a colony of cooperating agents. *Systems, Man, and Cybernetics, Part B, IEEE Transactions on.*

[Dorigo and Stützle, 2004] Dorigo, M. and Stützle, T. (2004). *Ant Colony Optimization.* Bradford Books. MIT Press, Cambridge, Massachusetts.

[Farahani et al., 2012] Farahani, R. Z., Asgari, N., Heidari, N., Hosseininia, M., and Goh, M. (2012). Covering problems in facility location: A review. *Computers & Industrial Engineering*, 62(1):368–407.

[Feo and Resende, 1989] Feo, T. and Resende, M. (1989). A probabilistic heuristic for a computationally difficult set covering problem. *Operations research letters*, 8(2):67–71.

[Fisher, 1981] Fisher, M. (1981). The Lagrangian Relaxation Method for Solving Integer Programming Problems. *Management Science*, 27(1):1–18.

[Flores et al., 1999] Flores, P., Neto, H., and Marques-Silva, J. (1999). On applying set covering models to test set compaction. In *VLSI, 1999. Proceedings. Ninth Great Lakes Symposium on*, pages 8–11. IEEE.

[Francis and White, 1977] Francis, R. and White, J. (1977). Facility layout and location: An analytical approach. 1974.

[Garey and Johnson, 1979] Garey, M. and Johnson, D. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness.* WH Freeman & Co. New York, NY, USA.

[Haddadi, 1997] Haddadi, S. (1997). Simple lagrangian heuristic for the set covering problem. *European journal of operational research*, 97(1):200–204.

[Hadji et al., 2000] Hadji, R., Rahoual, M., Talbi, E., and Bachelet, V. (2000). Ant colonies for the set covering problem. In *Abstract proceedings of ANTS2000–From Ant Colonies to Artificial Ants: A Series of International Workshops on Ant Algorithms*, pages 63–66.

[Housos and Elmroth, 1997] Housos, E. and Elmroth, T. (1997). Automatic optimization of subproblems in scheduling airline crews. *Interfaces*, pages 68–77.

[Jacobs and Brusco, 1994] Jacobs, L. and Brusco, M. (1994). A simulated annealing-based heuristic for the set-covering problem. In *Proceedings of Decision Sciences Institute*, volume 2, pages 1189–91.

[Jacobs and Brusco, 1995] Jacobs, L. and Brusco, M. (1995). A local-search heuristic for large set-covering problems. *Naval research logistics*, 42(7):1129–1140.

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

[Leguizamon and Michalewicz, 2000] Leguizamon, G. and Michalewicz, Z. (2000). Ant Systems for Subset Problems. *Unpublished manuscript.*

[Lessing et al., 2004] Lessing, L., Dumitrescu, I., and Stutzle, T. (2004). A Comparison Between ACO Algorithms for the Set Covering Problem. *Lecture Notes in Computer Science.*

[López-Ibáñez et al., 2016] López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L. P., Birattari, M., and Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58.

[Maniezzo, 1999] Maniezzo, V. (1999). Exact and Approximate Nondeterministic Tree-Search Procedures for the Quadratic Assignment Problem. *INFORMS Journal on Computing*, 11(4):358–369.

[Maniezzo and Colorni, 1999] Maniezzo, V. and Colorni, A. (1999). The ant system applied to the quadratic assignment problem. *Knowledge and Data Engineering, IEEE Transactions on*, 11(5):769 –778.

[Marchiori and Steenbeek, 2000] Marchiori, E. and Steenbeek, A. (2000). An Evolutionary Algorithm for Large Scale Set Covering Problems with Application to Airline Crew Scheduling. In *Real-World Applications of Evolutionary Computing: EvoWorkshops 2000: EvoIASP, EvoSCONDI, EvoTel, EvoSTIM, EvoRob, and EvoFlight, Edinburgh, Scotland, UK, April 17, 2000: Proceedings*, pages 367–381. Springer.

[Mulati, 2009] Mulati, M. H. (2009). Investigação da meta-heurística de otimização por colônia de formigas artificiais aplicada ao problema de cobertura de conjunto. Master's thesis, PCC, DIN, Universidade Estadual de Maringá, Maringá - PR - Brasil.

[Mulati and Constantino, 2011] Mulati, M. H. and Constantino, A. A. (2011). Antline: a line-oriented aco algorithm for the set covering problem. In *Chilean Computer Science Society (SCCC), 2011 30th International Conference of the*, pages 265–274. IEEE.

[Rahoual et al., 2002] Rahoual, M., Hadji, R., and Bachelet, V. (2002). Parallel ant system for the set covering problem. *Ant Algorithms*, pages 249–297.

[Ren et al., 2008] Ren, Z., Feng, Z., Ke, L., and Chang, H. (2008). A fast and efficient ant colony optimization approach for the set covering problem. In *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on*, pages 1839–1844. IEEE.

[Ren et al., 2010] Ren, Z., Feng, Z., Ke, L., and Zhang, Z. (2010). New ideas for applying ant colony optimization to the set covering problem. *Computers & Industrial Engineering*, 58(4):774–784.

[Solnon and Bridge, 2006] Solnon, C. and Bridge, D. (2006). An ant colony optimization meta-heuristic for subset selection problems. *System Engineering using Particle Swarm Optimization, Nova Science.*

[Stutzle and Hoos, 1998] Stutzle, T. and Hoos, H. (1998). Improvements on the ant-system: Introducing the max-min ant system. In *Artificial neural nets and genetic algorithms: proceedings of the International Conference in Norwich, UK, 1997*, volume 228, page 245. Springer Verlag Wien.

[Stützle and Hoos, 2000] Stützle, T. and Hoos, H. (2000). Max–min ant system. *Future Generation Computer Systems*, 16(8):889–914.

[Sundar and Singh, 2012] Sundar, S. and Singh, A. (2012). A hybrid heuristic for the set covering problem. *Operational Research*, 12(3):345–365.

[Toregas et al., 1971] Toregas, C., Swain, R., ReVelle, C., and Bergman, L. (1971). The location of emergency service facilities. *Operations Research*, pages 1363–1373.

[Vasko, 1984] Vasko, F. (1984). An efficient heuristic for large set covering problems. *Naval Research Logistics Quarterly*, 31(1):163–171.

[Yagiura et al., 2006] Yagiura, M., Kishida, M., and Ibaraki, T. (2006). A 3-flip neighborhood local search for the set covering problem. *European Journal of Operational Research.*

A Detailed Results by Ant-Set

This appendix presents the detailed results obtained in the experiments with the Ant-Set. The results obtained without and with local search are reported on Tables 5, 6 and 7, including the averaged data by SCP instance class.

The first column of those tables contains the SCP instance (C). The second column presents the OBK for the referred instance, according to [Ren et al., 2010]. The columns ATT, WS and AS contains summarized data from the 10 trials executed over an instance, where ATT is the average time from the trials of the run in seconds, WS is the distance between the worst solution reported by a trial of the run and the OBK of the instance in percentage, and AS is the distance between the average solution of the run and the OBK of the instance in percentage.

The remaining columns, IFB, TFB and BS contains information from the best trial of the run, where IFB is the number of iterations to find the first best solution of the trial, TFB is the time taken, in seconds, to find the first best solution and BS is the percentage distance between the best solution find by the trial and the OBK of the instance.

Table 5: Results by Ant-Set for the instances of SCP classes 4, 5 and 6.

		Ant-Set (without JB)						*Ant-Set+JB*					
C	OBK	ATT	WS	AS	IFB	TFB	BS	ATT	WS	AS	IFB	TFB	BS
		s	%	%		s	%	s	%	%		s	%
4.1	429	0.5	**0**	**0**	30	0.1	**0**	8.8	0.23	0.19	48	2.8	**0**
4.2	512	0.4	**0**	**0**	24	0.1	**0**	9.9	**0**	**0**	30	1.7	**0**
4.3	516	0.5	0.19	0.02	40	0.1	**0**	9.8	0.78	0.08	17	1.1	**0**
4.4	494	0.5	0.20	0.16	61	0.2	**0**	9.8	0.20	0.14	32	1.9	**0**
4.5	512	0.4	0.39	0.12	20	0.1	**0**	7.4	0.39	0.31	38	1.9	**0**
4.6	560	0.5	0.18	0.04	24	0.1	**0**	8.8	**0**	**0**	19	1.2	**0**
4.7	430	0.5	0.47	0.05	21	0.1	**0**	9.5	0.47	0.05	1	0.1	**0**
4.8	492	0.4	**0**	**0**	23	0.1	**0**	7.0	0.20	0.20	4	0.2	0.20
4.9	641	0.5	0.78	0.23	62	0.2	**0**	15.5	**0**	**0**	68	5.0	**0**
4.10	514	0.4	**0**	**0**	5	0.0	**0**	8.0	**0**	**0**	3	0.2	**0**
4		0.5	0.22	0.06	31	0.1	**0**	9.4	0.23	0.10	26	1.6	0.02
5.1	253	0.9	**0**	**0**	29	0.2	**0**	15.6	**0**	**0**	14	1.3	**0**
5.2	302	1.0	0.33	0.26	132	0.7	**0**	30.0	0.99	0.17	121	11.7	**0**
5.3	226	0.9	0.88	0.18	38	0.3	**0**	15.3	0.88	0.44	23	2.0	**0**
5.4	242	0.8	**0**	**0**	15	0.1	**0**	14.4	0.41	0.04	16	1.5	**0**
5.5	211	0.9	0.47	0.19	18	0.1	**0**	13.3	0.47	0.05	2	0.2	**0**
5.6	213	0.8	**0**	**0**	4	0.0	**0**	13.4	**0**	**0**	1	0.1	**0**
5.7	293	1.0	0.34	0.17	24	0.2	**0**	16.3	0.34	0.07	19	2.0	**0**
5.8	288	0.9	0.69	0.07	15	0.1	**0**	14.8	**0**	**0**	5	0.5	**0**
5.9	279	0.8	0.36	0.07	20	0.2	**0**	14.5	0.36	0.07	3	0.3	**0**
5.10	265	0.9	**0**	**0**	20	0.1	**0**	15.7	**0**	**0**	8	0.8	**0**
5		0.9	0.31	0.09	31.5	0.2	**0**	16.3	0.35	0.08	21.2	2.0	**0**
6.1	138	0.6	1.45	0.29	15	0.1	**0**	5.6	**0**	**0**	17	0.6	**0**
6.2	146	0.5	1.37	0.41	11	0.0	**0**	4.6	**0**	**0**	8	0.3	**0**
6.3	145	0.5	**0**	**0**	4	0.0	**0**	4.4	**0**	**0**	4	0.1	**0**
6.4	131	0.5	**0**	**0**	7	0.0	**0**	4.3	**0**	**0**	7	0.2	**0**
6.5	161	0.6	**0**	**0**	20	0.1	**0**	4.9	**0**	**0**	8	0.3	**0**
6		0.6	0.56	0.14	11.4	0.0	**0**	4.7	**0**	**0**	8.8	0.3	**0**

Table 6: Results by Ant-Set for the instances of SCP classes a, b, c, d and e.

		Ant-Set (without JB)						*Ant-Set+JB*					
C	OBK	ATT	WS	AS	IFB	TFB	BS	ATT	WS	AS	IFB	TFB	BS
		s	%	%		s	%	s	%	%		s	%
a.1	253	1.8	1.19	0.75	41	0.5	0.40	44.3	0.79	0.32	85	20.4	**0**
a.2	252	2.3	0.79	0.08	48	0.6	**0**	37.6	0.79	0.16	32	9.7	**0**
a.3	232	2.0	0.43	0.43	24	0.3	0.43	31.8	0.43	0.39	27	6.4	**0**
a.4	234	1.8	0.43	0.43	14	0.2	0.43	41.4	0.43	0.17	70	14.1	**0**
a.5	236	2.0	0.42	0.30	27	0.3	**0**	32.7	**0**	**0**	15	3.1	**0**
a		2.0	0.65	0.40	30.8	0.4	0.25	37.5	0.49	0.21	45.8	10.7	**0**
b.1	69	2.1	**0**	**0**	1	0.0	**0**	13.2	**0**	**0**	1	0.1	**0**
b.2	76	2.1	**0**	**0**	5	0.1	**0**	14.7	**0**	**0**	2	0.2	**0**
b.3	80	2.0	1.25	0.12	2	0.0	**0**	13.5	**0**	**0**	1	0.1	**0**
b.4	79	2.1	**0**	**0**	5	0.1	**0**	15.6	**0**	**0**	3	0.3	**0**
b.5	72	1.9	**0**	**0**	2	0.0	**0**	13.4	**0**	**0**	1	0.1	**0**
b		2.0	0.25	0.02	3.0	0.0	**0**	14.1	**0**	**0**	1.6	0.2	**0**
c.1	227	4.0	0.88	0.18	59	1.1	**0**	88.9	**0**	**0**	90	34.2	**0**
c.2	219	4.0	1.37	0.50	79	1.5	**0**	76.1	0.91	0.27	68	23.4	**0**
c.3	243	4.1	1.65	0.41	64	1.2	**0**	71.8	0.41	0.33	97	35.4	**0**
c.4	219	2.7	**0**	**0**	24	0.4	**0**	53.1	**0**	**0**	11	3.6	**0**
c.5	215	3.9	0.93	0.14	44	0.9	**0**	59.4	**0**	**0**	26	10.6	**0**
c		3.8	0.97	0.25	54.0	1.0	**0**	69.9	0.26	0.12	58.4	21.5	**0**
d.1	60	3.8	1.67	0.67	17	0.4	**0**	28.3	**0**	**0**	9	1.6	**0**
d.2	66	3.6	1.52	0.15	7	0.2	**0**	25.5	1.52	0.15	8	1.3	**0**
d.3	72	3.6	1.39	0.83	7	0.3	**0**	29.5	1.39	0.28	15	2.7	**0**
d.4	62	3.2	**0**	**0**	2	0.0	**0**	22.3	**0**	**0**	1	0.3	**0**
d.5	61	3.3	**0**	**0**	6	0.1	**0**	23.7	**0**	**0**	1	0.1	**0**
d		3.5	0.91	0.33	7.8	0.2	**0**	25.8	0.58	0.09	6.8	1.2	**0**
e.1	5	0.2	**0**	**0**	1	0.0	**0**	0.5	**0**	**0**	1	0.0	**0**
e.2	5	0.2	**0**	**0**	1	0.0	**0**	0.5	**0**	**0**	1	0.0	**0**
e.3	5	0.2	**0**	**0**	1	0.0	**0**	0.5	**0**	**0**	1	0.0	**0**
e.4	5	0.2	**0**	**0**	1	0.0	**0**	0.5	**0**	**0**	1	0.0	**0**
e.5	5	0.2	**0**	**0**	1	0.0	**0**	0.6	**0**	**0**	1	0.0	**0**
e		0.2	**0**	**0**	1.0	0.0	**0**	0.5	**0**	**0**	1.0	0.0	**0**

Table 7: Results by Ant-Set for the instances of SCP classes nre, nrf, nrg, and nrh.

		Ant-Set (without JB)						*Ant-Set+JB*					
C	OBK	ATT s	WS %	AS %	IFB	TFB s	BS %	ATT s	WS %	AS %	IFB	TFB s	BS %
nre.1	29	6.7	**0**	**0**	1	0.0	**0**	19.8	**0**	**0**	1	0.2	**0**
nre.2	30	7.5	3.33	0.67	6	0.4	**0**	24.3	**0**	**0**	3	0.5	**0**
nre.3	27	6.7	3.70	1.48	2	0.1	**0**	24.5	**0**	**0**	2	0.3	**0**
nre.4	28	6.8	3.57	2.50	6	0.3	**0**	23.8	**0**	**0**	4	0.6	**0**
nre.5	28	6.8	**0**	**0**	1	0.0	**0**	20.3	**0**	**0**	1	0.1	**0**
nre		6.9	2.12	0.93	3.2	0.2	**0**	22.5	**0**	**0**	2.2	0.3	**0**
nrf.1	14	11.4	**0**	**0**	1	0.1	**0**	20.1	**0**	**0**	1	0.1	**0**
nrf.2	15	11.5	**0**	**0**	1	0.1	**0**	20.0	**0**	**0**	1	0.2	**0**
nrf.3	14	11.8	7.14	3.57	19	1.6	**0**	20.7	**0**	**0**	3	0.5	**0**
nrf.4	14	11.5	7.14	0.71	3	0.3	**0**	19.8	**0**	**0**	1	0.1	**0**
nrf.5	13	11.1	7.69	7.69	1	0.1	7.69	26.0	7.69	0.77	29	4.7	**0**
nrf		11.5	4.40	2.40	5.0	0.4	1.54	21.3	1.54	0.15	7.0	1.1	**0**
nrg.1	176	15.0	0.57	0.06	33	3.2	**0**	293.8	**0**	**0**	32	52.0	**0**
nrg.2	154	17.7	1.30	0.91	41	4.1	0.65	414.2	1.30	0.91	87	142.0	0.65
nrg.3	166	22.0	3.01	1.93	94	8.8	1.20	486.4	3.01	1.20	209	338.2	0.60
nrg.4	168	18.5	1.19	1.13	59	5.6	0.60	442.6	1.79	0.48	115	197.4	**0**
nrg.5	168	15.1	**0**	**0**	38	3.7	**0**	346.2	**0**	**0**	51	85.3	**0**
nrg		17.7	1.21	0.80	53.0	5.1	0.49	396.6	1.22	0.52	98.8	163.0	0.25
nrh.1	63	23.4	3.17	3.02	57	7.5	1.59	137.1	4.76	2.38	37	29.7	1.59
nrh.2	63	19.3	3.17	2.86	56	7.5	1.59	143.4	3.17	2.06	38	30.9	1.59
nrh.3	59	21.3	3.39	2.88	45	9.0	**0**	108.5	3.39	3.39	11	7.6	3.39
nrh.4	58	21.4	3.45	1.90	52	6.9	**0**	160.6	3.45	1.03	79	61.6	**0**
nrh.5	55	17.9	**0**	**0**	10	1.3	**0**	108.4	**0**	**0**	8	6.6	**0**
nrh		20.6	2.64	2.13	44.0	6.4	0.63	131.6	2.95	1.77	34.6	27.3	1.31