Computers & Industrial Engineering 58 (2010) 774–784

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

New ideas for applying ant colony optimization to the set covering problem ☆

Zhi-Gang Ren *, Zu-Ren Feng, Liang-Jun Ke, Zhao-Jun Zhang

State Key Laboratory for Manufacturing Systems Engineering, Systems Engineering Institute, Xi'an Jiaotong University, Xi'an, Shaanxi 710049, China

ARTICLE INFO

Article history:
Received 7 October 2008
Received in revised form 2 February 2010
Accepted 12 February 2010
Available online 18 February 2010

Keywords:
Set covering problem
Ant colony optimization
Local search
Metaheuristic

ABSTRACT

The set covering problem (SCP) is a well known NP-hard problem with many practical applications. In this research, a new approach based on ant colony optimization (ACO) is proposed to solve the SCP. The main differences between it and the existing ACO-based approaches lie in three aspects. First, it adopts a novel method, called single-row-oriented method, to construct solutions. When choosing a new column, it first randomly selects an uncovered row and only considers the columns covering this row, rather than all the unselected columns as candidate solution components. Second, a kind of dynamic heuristic information is used in this approach. It takes into account Lagrangian dual information associated with currently uncovered rows. Finally, a simple local search procedure is developed to improve solutions constructed by ants while keeping their feasibility. The proposed algorithm has been tested on a number of benchmark instances. Computational results show that it is able to produce competitive solutions in comparison with other metaheuristics.

1. Introduction

The set covering problem (SCP) is a class of representative combinatorial optimization problems (COPs). It consists in finding a subset of columns in a zero–one matrix such that they cover all the rows of the matrix at minimum cost. Many real world problems can be modeled by the SCP, such as crew scheduling in airlines (e.g. Housos & Elmoth, 1997), facility location problem (Vasko & Wilson, 1984), and production planning in industry (Vasko & Wolf, 1987).

It is well known that the SCP is NP-hard in the strong sense (Garey & Johnson, 1979). Many algorithms have been developed to solve it. Exact algorithms are mostly based on brand-and-bound and branch-and-cut (Balas & Carrera, 1996; Fisher & Kedia, 1990). However, these algorithms are rather time consuming and can only solve instances of very limited size. For this reason, many research efforts have been focused on the development of heuristics to find good or near-optimal solutions within a reasonable period of time. Classical greedy algorithms are very simple, fast, and easy to code in practice, but they rarely produce high quality solutions for their myopic and deterministic nature (Chvatal, 1979). Lan and DePuy (2006) improved a greedy algorithm by incorporating randomness and memory into it and obtained promising results. Compared with classical greedy algorithms, heuristics based on Lagrangian relaxation with subgradient optimization are much more effective. The most efficient ones are those proposed by Ceria, Nobili, and Sassano (1998) and Caprara, Fischetti, and Toth (1999). As top-level general search strategies, metaheuristics were also applied to the SCP. An incomplete list of this kind of heuristics for the SCP includes genetic algorithm (Beasley & Chu, 1996), simulated annealing algorithm (Brusco, Jacobs, & Thompson, 1999), and tabu search algorithm (Caserta, 2007). For a deeper comprehension of most of the effective algorithms for the SCP in the literature, we refer the interested reader to the survey by Caprara, Toth, and Fischetti (2000).

In this paper, a new approach based on ant colony optimization (ACO) for the SCP is presented. ACO is a recently developed, population-based metaheuristic (Dorigo, Birattari, & Stützle, 2006; Dorigo & Di Caro, 1999; Dorigo & Stützle, 2004). So far, it has been successfully applied to a great variety of hard COPs. Researches on ACO have shown that it performs well on problems which have high fitness-distance correlation (FDC) (Stützle & Hoos, 2000). FDC provides a good way to describe the solution space characteristics of a COP by measuring the correlation between the solution fitness and the distance to the closest optimal solution. Finger, Stützle, and Ramalhinho (2002) tested and analyzed the FDC values for different types of SCP instances. The high correlation coefficient values for most of the instances tested indicate that it is appropriate to solve the SCP with ACO.

It is worth noting that some ACO-based approaches for the SCP have been proposed. Hadji, Rahoual, Talbi, and Bachelet (2000) applied ant system to the SCP and Lessing, Dumitrescu, and Stützle (2004) studied the behavior of several ACO variants in solving it. More recently, Crawford and Castro (2006) merged a post processing procedure with ACO for the SCP. All these work verified the feasibility of applying ACO to the SCP. However, their

☆ This manuscript was processed by Area Editor Ibrahim H. Osman.
* Corresponding author. Tel./fax: +86 29 82665487.
E-mail address: whrzg5258@gmail.com (Z.-G. Ren).

computational performance is relatively poor compared with other metaheuristics, such as genetic algorithm (Beasley & Chu, 1996). In this study, we propose a novel ACO-based approach, called Ant-Cover, with the aim of enhancing the ability of ACO in solving the SCP. Different from existing ACO-based algorithms which consider all the unselected columns as candidate solution components while choosing a new column, Ant-Cover first randomly selects an uncovered row and only takes into account the columns covering it. By this means, only the columns that can cover new rows may be selected at each construction step, and it needs less time to select a new column since the number of candidate solution components becomes less. On the other hand, Ant-Cover uses a kind of dynamic heuristic information to guide ants searching in the solution space, and the heuristic information is only associated with the Lagrangian dual information of currently uncovered rows. In addition, a local search procedure is developed to improve solutions constructed by ants. To verify the efficiency of the proposed algorithm, computational experiments are conducted on the benchmark instances from Beasley's OR Library (Beasley, 1990).

The remainder of this paper is organized as follows: In Section 2, we formally introduce the SCP, together with a brief description of a preprocessing method for it. In Section 3, we present the proposed algorithm in detail. The computational experiments and results are given in Section 4. Finally, conclusions are drawn in Section 5.

2. Problem formulation

The set covering problem can be formally defined as follows. Let $A = (a_{ij})$ be an m-row, n-column, zero–one matrix. We say that a column j covers a row i if $a_{ij} = 1$. Each column j is associated with a nonnegative real cost c_j. Let $I = \{1, \ldots, m\}$ and $J = \{1, \ldots, n\}$ be the row set and column set, respectively. The SCP calls for a minimum cost subset $S \subseteq J$, such that each row $i \in I$ is covered by at least one column $j \in S$. A mathematical model for the SCP is

$$v(\mathrm{SCP}) = \min \sum_{j \in J} c_j x_j \tag{1}$$

subject to

$$\sum_{j \in J} a_{ij} x_j \geq 1, \quad \forall\ i \in I, \tag{2}$$

$$x_j \in \{0, 1\}, \quad \forall\ j \in J \tag{3}$$

where $x_j = 1$ if $j \in S$, $x_j = 0$ otherwise. The following notations are often used to describe the set covering problem:

$J_i = \{j \in J \mid a_{ij} = 1\}$: the subset of columns covering row i.
$I_j = \{i \in I \mid a_{ij} = 1\}$: the subset of rows covered by column j.
$d = \Sigma_{i \in I} \Sigma_{j \in J}\, a_{ij}/(mn)$: the density of SCP, i.e., the ratio of non-zero entries in matrix A.

Preprocessing is a popular method to speed up algorithms by reducing the size of an instance. A number of preprocessing methods for the SCP have been proposed in the literature (Fisher & Kedia, 1990). In this study, two of those found to be most effective are used.

Column domination: If a column j whose rows I_j can be covered by other columns with a total cost lower than c_j, we say that column j is dominated and can be removed. For a column j, a reduced set covering problem can be derived with the members of I_j being its rows and the columns covering at least one row in I_j being its columns. Then domination judgment of column j implies checking whether there exists a solution to the reduced problem with a cost lower than c_j. It is an NP-C problem (Garey & Johnson, 1979). Hence checking all possible solutions is impractical. We therefore use this rule in a limited way. First, all the columns are sequenced in increasing order of cost and columns of equal cost are sequenced in decreasing order of the number of rows that they cover. This sequence is denoted as J'. Then, for each row $i \in I_j$, the column with minimal cost in J_i, denoted as low_i, is found out. It is the column of J_i first appearing in J'. If the total cost of all the columns low_i, $i \in I_j$, is lower than the cost of column j, column j is deleted from the instance.

Column inclusion: If a row is covered by only one column after the above domination, this column is definitely included in an optimal solution.

For the convenience of description, we explain the approach hereinafter assuming that it is applied to the original instances of SCP, though it is actually applied to the preprocessed ones. Besides, some abbreviations are introduced in the remaining part of this paper. All of them are listed in Table 1.

3. Description of the proposed approach

This section presents the details of the strategies used in Ant-Cover, including the solution construction method, the heuristic scheme, the pheromone update rule, and the local search procedure. As a basis, the principles of ACO are introduced first.

3.1. Principles of ant colony optimization

Ant colony optimization is inspired by the observation on the foraging behavior of real ant colonies (Dorigo & Di Caro, 1999). Its main idea is to model the problem as the search for a minimum cost path in a graph by a colony of artificial ants that work cooperatively and communicate through artificial pheromone trails. The pheromone trail is a kind of distributed information which is modified by ants to reflect their experience accumulated during the search process. The search process consists of a series of iterations. In each iteration, each ant constructs a solution step by step guided by pheromone trails and problem dependent heuristic information. Once each ant has generated a solution, it deposits some pheromone on "good" solution components. So the ants will search these promising areas in the next iteration with the aim of finding the optimal solution finally. Generally, the algorithm stops when a certain number of iterations have been performed or a fixed amount of CPU time has elapsed.

The first ACO algorithm is Ant System (Dorigo, Maniezzo, & Colorni, 1996). Since its debut, many efforts have been done to improve its performance, and consequently various ACO variants were proposed. Some of them are Ant Colony System (ACS) (Dorigo & Gambardella, 1997), Max–Min Ant System (MMAS) (Stützle &

Table 1
List of abbreviations.

Abbreviation	Full name
ACO	Ant Colony Optimization
ACS	Ant Colony System
ARPD	Average RPD value over the test instances
ARPI	Average RPI value over the test instances
COP	Combinatorial Optimization Problem
FDC	Fitness-Distance Correlation
GA	Genetic Algorithm
MIC	Marginal relative Improvement per unit of CPU time
MMAS	Max–Min Ant System
RPD	Relative Percentage Deviation of an objective value Z from Z_{ob}
RPI	Relative Percentage Improvement of Z from Z_{I}
SA	Simulated Annealing
SCP	Set Covering Problem
SROM	Single-Row-Oriented solution construction Method
TSP	Traveling Salesman Problem

Hoos, 2000), and the Hyper-Cube Framework for ACO (Blum & Dorigo, 2004). MMAS is one of the most outstanding ACO variants. Its main characteristic is that it tries to avoid search stagnation by imposing upper and lower limits on pheromone trails. Ant-Cover, the approach presented in this paper for the SCP, follows the standard algorithmic scheme of MMAS for static COPs. However, it has some new features with respect to the characteristics of the SCP. The framework of Ant-Cover will be given in Section 3.6.

3.2. Solution construction method

During the solution construction process, ants communicate with each other indirectly through pheromone trails, so we should define the pheromone scheme first. From the mathematical model of SCP (Eqs. (1)–(3)), it is easy to see that SCP is a kind of subset selection problem. For this kind of problems, Gutjahr (2006) analyzed three types of solution construction graphs, including chains, disks, and drums, while investigating the finite-time dynamics of ACO. Also, Solnon and Bridge (2006) studied another two construction graphs and provided corresponding pheromone schemes. One scheme is to lay pheromone on objects and the other is to lay pheromone on the pairs of objects. The comparison between the two schemes indicated that the former needs less storage space and can find satisfactory solutions for the problems tested within a shorter period of time. Due to these advantages, it is adopted by Ant-Cover. More precisely, in Ant-Cover, the columns in SCP are chosen as solution components. Each column j is associated with a pheromone trail τ_j and a heuristic factor η_j which, respectively, indicate the learned desirability and the prior desirability of including column j into an ant's solution. Heuristic factor is a function of problem dependent information. Its detailed definition will be shown in Section 3.3.

While constructing a solution, each ant starts with an empty solution and adds columns iteratively until all the rows are covered. It does so by probabilistically preferring solution components with high pheromone and heuristic values. For most of the existing ACO-based algorithms (Crawford & Castro, 2006; Lessing et al., 2004), at construction step t, an ant chooses column j from all the unselected columns according to the following state transition rule:

$$P(s_t = j|S_{t-1}) = \begin{cases} \frac{\tau_j \eta_j^\beta}{\sum_{q \in J \setminus S_{t-1}} \tau_q \eta_q^\beta}, & \text{if } j \in J \setminus S_{t-1} \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

where S_{t-1} denotes the partial solution constructed before step t, $J \setminus S_{t-1}$ denotes the subset of unselected columns, s_t denotes the column that will be chosen at step t, and the parameter β ($\beta \geqslant 0$) determines the relative importance of heuristic factor with respect to pheromone. Without considering the computation of heuristic factor,[1] the time complexity of step t is $O(n - |S_{t-1}|)$, where $|S_{t-1}|$ represents the number of columns in S_{t-1}. Let S be a complete solution with k columns. The time complexity of constructing S is $O(kn - k^2/2 + k/2)$ which can be calculated conveniently by summing the time complexity of each step. In most cases, $k \ll n$ holds, so this time complexity is approximately equal to $O(kn)$. That is, it is linear with the number of columns in the instance to be solved. Considering that the solution construction process needs to be repeated for many times in the algorithm, the computation cost of constructing a solution in this way is expensive, especially for large scale instances.

[1] There are several possible ways to define heuristic factor, and it is a little difficult to analyze the time complexity for some of them. Here we mainly pay attention to the computation complexity of solution construction method.

To reduce the computation burden, a possible way is to use candidate set strategy. It has been adopted by ACS (Dorigo & Gambardella, 1997) and MMAS (Stützle & Hoos, 2000) in solving large scale traveling salesman problem (TSP) instances where a certain number of nearest cities are contained in the candidate set of each city. While constructing a solution, an ant probabilistically chooses the next city among those in the candidate set, if possible. Only if all the cities in the candidate set have already been visited, one of the remaining cities is chosen. The candidate set strategy has shown great effectiveness in solving the TSP. However, while applying it to the SCP, two difficulties appear. First, there exists no strong relationship among columns in SCP, like the distance between every two cities in TSP, to determine which columns should be put into the candidate set. Second, it is difficult to determine the number of columns in the candidate set. Limited experimental results showed that a great value cannot reduce the computation burden remarkably and a small value may deteriorate the solution quality.

In fact, we can solve this problem in another way. For the SCP, each row must be covered by at least one column in order to get a feasible solution. It means that, for a row $i \in I$, at least one column must be selected from the set J_i. Therefore, at construction step t, it is feasible to choose an uncovered row i first, and then select a column j from the set J_i with probability

$$P(s_t = j|r_t = i, i \in R_{t-1}) = \begin{cases} \frac{\tau_j \eta_j^\beta}{\sum_{q \in J_i} \tau_q \eta_q^\beta}, & \text{if } j \in J_i \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

where R_{t-1} denotes the set of uncovered rows before step t, and r_t denotes the row chosen at step t. According to Eq. (5), the time complexity of selecting a column from the set J_i is $O(|J_i|)$. Considering the density of SCP d, we can get the average number of columns covering a row. It is $\bar{n} = dn$. Then a complete solution S with k columns can be obtained in $O(k\bar{n})$ time. That is, this time complexity is linear with the value of $\bar{n}$. Generally, $\bar{n} \ll n$ holds. Although the number of columns in each solution may be different, on the average, the time complexity of generating a solution based on Eq. (5) is much lower than that based on Eq. (4). Since an uncovered row must be chosen before applying Eq. (5), we name this method single-row-oriented solution construction method (SROM).

Another problem is to determine the processing order of those uncovered rows. It can be solved in three possible ways. They are the natural order of row index in matrix A, the decreasing order according to some heuristic information such as the number of columns covering the row, and the random order. Preliminary experimental results demonstrated that the third method performs best. The reason may be that, compared with the first two methods, its randomness allows ants to explore broader solution areas and can prevent search stagnation to some extent. Synthetically, a column j can be selected at construction step t with probability

$$P(s_t = j|R_{t-1}) = \begin{cases} \frac{1}{|R_{t-1}|} \sum\limits_{i \in I_j \cap R_{t-1}} \frac{\tau_j \eta_j^\beta}{\sum_{q \in J_i} \tau_q \eta_q^\beta}, & \text{if } I_j \cap R_{t-1} \neq \emptyset \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

where the expression $I_j \cap R_{t-1}$ indicates currently uncovered rows that can be covered by column j, and $1/|R_{t-1}|$ is the probability of each row in R_{t-1} being selected. Eq. (6) is obtained according to the total probability formula. It implies that, besides the influence of pheromone trails and heuristic information, the more new rows a column can cover, the more likely it is to be selected. Fig. 1 describes the main operations of the SROM.

To provide an insight into the performance of the new proposed solution construction method and to study its influence on the capability of ants to explore the solution space, we now try to

```
1. Initialize $S_t$ and $R_t$ by setting $t \leftarrow 0$, $S_t \leftarrow \emptyset$ , and $R_t \leftarrow I$.
2. while $R_t \neq \emptyset$ do
3.     $t \leftarrow t + 1$;
4.     Randomly select a row $i$ from the row set $R_{t-1}$;
5.     Select a column $j$ from the column set $J_i$ according to Eq. (5);
6.     Update solution $S_t$ and set $R_t$ by setting $S_t \leftarrow S_{t-1} \cup \{j\}$, $R_t \leftarrow R_{t-1} \setminus I_j$;
7. end while
8. Return solution $S_t$.
```

Fig. 1. Single-row-oriented solution construction method.

measure the diversity of the generated solutions at run time. Indeed, for a population-based search algorithm, maintaining population diversity is a key point to prevent premature convergence and stagnation. Many diversity measures have been introduced in the literature Burke, Gustafson, and Kendall (2002). In this research, we use the similarity ratio given by Solnon and Fenet (2006) while solving the maximum clique problem with ACO. For SCP, the similarity ratio for a set of solutions P is defined by the average number of columns that are shared by any pair of solutions in P, divided by the average size of solutions in P. Formally, it is given as follows:

$$r_{sim}(P) = \frac{\sum_{j \in J}(\text{freq}[j](\text{freq}[j] - 1))}{(|P| - 1)\sum_{S \in P}|S|} \tag{7}$$

where freq[j] denotes the number of solutions in P which have selected column j. This ratio is equal to one if all the solutions in P are identical, whereas it is equal to zero if the intersection of each pair of solutions in P is empty.

Fig. 2 plots the evolution of the similarity ratio of solutions generated at each iteration while applying two different solution construction methods. (Note that local search is not used here and parameters are set to the same values as described in Section 4.1.) Instance 5.1 is taken as an example here. As shown in Fig. 2, from an overall perspective, the similarity ratios for both methods increase with respect to the number of iterations which means that ants gradually focus on a sub-area of the solution space. On the other hand, the similarity ratio of SROM is always smaller than that of the traditional solution construction method based on Eq. (4). This suggests that, compared with the traditional method, SROM enables ants to explore broader solution areas. Note that there is

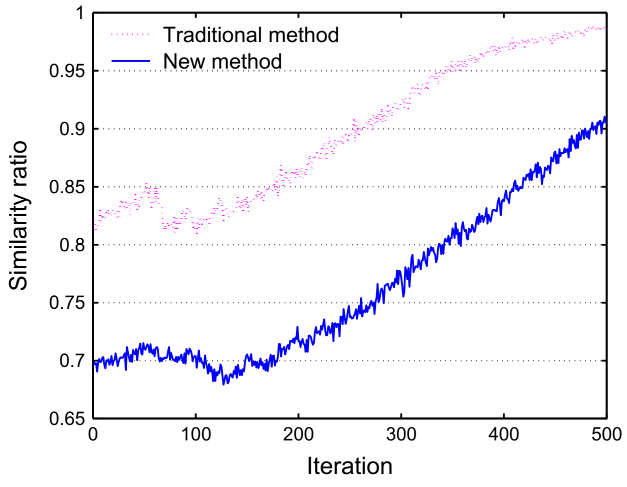

Fig. 2. Evolution of similarity ratio for two solution constrction methods.

a valley in each curve. The reason is that the Lagrangian multiplier vector is recalculated there. (This part is introduced with more details at the end of Section 3.3.) It changes the heuristic value and directs ants to explore new solution areas.

In summary, the single-row-oriented solution construction method has the following characteristics:

- It generates a solution much faster than the traditional method based on Eq. (4).
- No matter what kind of heuristic information (as long as the heuristic value is greater than zero. This condition is always met in ACO) is used in the algorithm, once a column is selected, a new row will be covered at least. So the maximum number of steps to construct a complete solution in this way is $m - r + 1$ where $r = \min\{|I_j| \mid j \in J\}$ is the minimum number of rows covered by a column. Generally, $r > 1$ holds.
- The traditional method considers all the unselected columns as candidate solution components. Instead, SROM just considers the columns that can cover new rows.
- The search areas of SROM are a little more diversified than those of the traditional method.

3.3. The definition of heuristic information

It is widely acknowledged that heuristic information can play a significant role in the performance of the ACO algorithms. Several kinds of heuristic information have been developed for the SCP, including static ones and dynamic ones. The most commonly used heuristic information for a column j is defined as follows (Crawford & Castro, 2006; Lessing et al., 2004):

$$\varphi_j = |I_j \cap R|, \tag{8}$$
$$\eta_j = \varphi_j / c_j \tag{9}$$

where R is the set of still uncovered rows and φ_j indicates the number of new rows that can be covered by column j. Obviously, the heuristic information defined by Eq. (9) depends on the current partial solution and is a kind of dynamic information. According to it, a column which can cover more new rows and is of lower cost has a greater heuristic value. In this section, we define another type of dynamic heuristic information based on the Lagrangian relaxation information of the SCP.

The Lagrangian relaxation (Wolsey, 1998) of SCP for a given Lagrangian multiplier vector $\boldsymbol{u} = (u_1, \ldots, u_m) \in R_+^m$ (R_+ is the set of nonnegative real numbers) can be defined as follows:

$$L(\boldsymbol{u}) = \min_{x \in \{0,1\}^n} \sum_{j \in J} c_j(\boldsymbol{u}) x_j + \sum_{i \in I} u_i \tag{10}$$

where $c_j(\boldsymbol{u}) = c_j - \Sigma_{i \in I_j} u_i$ is the Lagrangian cost associated with column j. $L(\boldsymbol{u})$ provides a lower bound on the optimal value of the SCP, i.e., $L(\boldsymbol{u}) \leqslant v(\text{SCP})$ holds for any $\boldsymbol{u} \in R_+^m$. The best such lower bound on the optimal solution of the SCP can be obtained by solving the Lagrangian dual problem, $\max\{L(\boldsymbol{u}) | \boldsymbol{u} \in R_+^m\}$, with subgradient method (Fisher, 1981). For a near-optimal Lagrangian multiplier vector $\boldsymbol{u}$, the corresponding Lagrangian cost $c_j(\boldsymbol{u})$ gives reliable information on the attractiveness of selecting column j into a SCP solution, because each column j in an optimal solution of SCP tends to have a small $c_j(\boldsymbol{u})$ value. The subgradient method generates a sequence of Lagrangian multiplier vectors with a given initial vector. Various implementations of this method are possible. In this study, we use the sophisticated implementation proposed by Caprara et al. (1999), as it can quickly produce a near-optimal Lagrangian multiplier vector. In Ant-Cover, the subgradient method is performed at the beginning of the algorithm, and is stopped after at most $10m$ iterations.

For a best Lagrangian multiplier vector $\boldsymbol{u}$ found by the subgradient method, we first define the following expression for each column j:

$$\psi_j(\boldsymbol{u}) = c_j - \sum_{i \in I_j \cap R} u_i. \tag{11}$$

The heuristic information used in Ant-Cover is defined as a function of $\psi_j(\boldsymbol{u})$ and φ_j. A function in which the heuristic value η_j increases with the argument φ_j and decreases with the argument $\psi_j(\boldsymbol{u})$ is desirable. The following two forms have been considered:

$$\eta_j = \varphi_j - \psi_j(\boldsymbol{u}), \tag{12}$$

$$\eta_j = \varphi_j / (\psi_j(\boldsymbol{u}) + \sigma). \tag{13}$$

It is found that the absolute value of $\psi_j(\boldsymbol{u})$ is usually much smaller than 1, whereas the value of φ_j is always greater than or equal to 1 since the candidate columns considered in Ant-Cover can cover at least a new row. Thus the influence of $\psi_j(\boldsymbol{u})$ on η_j may be submerged by φ_j according to the function depicted by Eq. (12). So we will focus on the function given in Eq. (13). Note that the probability of selecting a solution component cannot be negative in ACO and the values of $\psi_j(\boldsymbol{u}), j \in J$, may be negative sometimes. Therefore, they are normalized by adding a small positive number σ (σ is typically set to $2|\min\{c_j(\boldsymbol{u})|j \in J\}|$). Compared with Lagrangian cost $c_j(\boldsymbol{u})$, $\psi_j(\boldsymbol{u})$ only considers the dual information associated with the still uncovered rows, and results in higher accuracy of the computed heuristic values accordingly. The value of each η_j varies during the solution construction process. However, there is no need to compute η_j from scratch at each construction step, because φ_j and $\psi_j(\boldsymbol{u})$ can be computed in an incremental way. Once a column j is chosen, the values of φ_j and $\psi_j(\boldsymbol{u})$ are updated by the following steps:

```
for each $i \in I_j \cap R$ do
  for each $j \in J_i$ do
    $\psi_j(\boldsymbol{u}) \leftarrow \psi_j(\boldsymbol{u}) + u_i$ and $\varphi_j \leftarrow \varphi_j - 1$
  end for
end for
```

The heuristic information defined above provides a good guideline for ants' search in the solution space. However, no strict correlation exists between the heuristic information and the quality of the SCP solutions. Sometimes, a column with low heuristic value need be selected in order to get an overall good solution. Hence it is useful to supply a series of near-optimal Lagrangian multiplier vectors and to alter the influence of $\psi_j(\boldsymbol{u})$ on the heuristic information. Concretely, during the run process of Ant-Cover, if no solution quality improvement occurs for p (p is typically set to 50) consecutive iterations, we re-operate the subgradient method and get a new Lagrangian multiplier vector. The initial vector $\boldsymbol{u}$ of the subgradient method is obtained by putting a random perturbation on the best Lagrangian multiplier vector $\boldsymbol{u}^*$ found before, i.e., $u_i = (1 + \delta)u_i^*$ for each $i \in I$, where δ is a uniformly random number in the range $[-0.2, 0.2]$.

3.4. Pheromone update rule

While updating pheromone trails, Ant-Cover follows the basic rules used in MMAS (Stützle & Hoos, 2000). In every iteration, after each ant has constructed a complete solution, pheromone trails are updated. First, all the pheromone trails are decreased uniformly in order to simulate evaporation and allow ants to forget part of the history experience. Then the ants deposit an amount of pheromone on the columns contained in the best solution. It may be the best solution found in the current iteration (S_{ib}) or the best from the beginning of the algorithm (S_{gb}). The standard MMAS favors S_{ib} for pheromone reinforcement. By contrast, Ant-Cover focuses on the use of S_{gb}. The reason is that when using S_{gb}, the search can concentrate fast around this solution. The risk of getting trapped in poor solution areas may be reduced by the solution construction method used in Ant-Cover since it is of stronger randomness and allows ants to explore broader solution areas. The range of pheromone values is limited to an interval $[\tau_{\min}, \tau_{\max}]$ with the aim of avoiding search stagnation. More formally, pheromone trails are updated as follows:

$$\tau_j \leftarrow \rho\tau_j + \Delta\tau_j, \quad \forall\, j \in J \tag{14}$$

$$\Delta\tau_j = \begin{cases} 1 \Big/ \sum_{q \in S_{\text{gb}}} c_q, & \text{if } j \in S_{\text{gb}} \\ 0, & \text{otherwise} \end{cases} \tag{15}$$

$$\text{if } \tau_j < \tau_{\min}, \quad \text{then } \tau_j = \tau_{\min} \tag{16}$$

$$\text{if } \tau_j > \tau_{\max}, \quad \text{then } \tau_j = \tau_{\max} \tag{17}$$

where the parameter ρ (with $0 \leqslant \rho < 1$) is the pheromone persistence and $\Delta\tau_j$ is the amount of pheromone put on column j. If column j is contained in the best-so-far solution S_{gb}, then the value of $\Delta\tau_j$ is inversely proportional to the cost of S_{gb}. As suggested by MMAS, $\tau_{\max}$ is set to an estimate of the asymptotically maximum value of pheromone trails, i.e., $\tau_{\max} = 1/((1-\rho)(\Sigma_{j \in S\text{gb}} c_j))$. Each time a new best solution is found, $\tau_{\max}$ is updated. $\tau_{\min}$ is set to $\varepsilon\tau_{\max}$, where ε (with $0 < \varepsilon < 1$) is a ratio coefficient. Moreover, pheromone trails are initialized to an arbitrary high value for the purpose of achieving a higher exploration of solution space at the beginning of the algorithm.

3.5. Local search procedure

It is known that the coupling of ACO and local search is effective to improve the performance of the pure ACO (Dorigo & Stützle, 2004). In fact, all the best performing ACO algorithms for different COPs improve solutions generated by ants with local search algorithms. In this section, a new local search procedure is proposed for the SCP.

Although each solution generated in Ant-Cover is feasible, it may contain redundant columns since the rows covered by early selected columns may be covered by later selected ones too, then those early selected columns are redundant. The local search procedure proposed in this research tries to improve solution quality by removing redundant columns and replacing high cost columns with low cost ones while keeping the feasibility of solutions. The algorithm is implemented in the following way.

For each solution S constructed by ants, let o_i be the number of columns covering a row $i, i \in I$. It satisfies that $o_i \geqslant 1$ since the solution S is feasible. For each $j \in S$, let $W_j = \{i \in I_j | o_i = 1\}$ be the set of rows only covered by column j. If $|W_j| = 0$, it means that the column j is redundant and can be directly removed. On the other hand, if $|W_j| > 0$, we compare the cost of column j with the total cost of all the columns $low_i, i \in W_j$ (low_i is the same as the one defined in Section 2, i.e., the column with minimal cost among all the columns covering row i). If the latter is not higher than the former, then we replace column j with all the columns $low_i, i \in W_j$. Once a column is removed from or added to the solution S, the value of o_i is updated. The pseudocode for the local search procedure is given in Fig. 3.

In the local search procedure presented above, we try to remove columns with high cost first whenever possible. The operations from step 1 to 7 are easy to understand. They mainly deal with the case that a column j is redundant or can be replaced by another column with lower cost. However, the operations from step 8 to 11 are somewhat complex. They deal with the case of $|W_j| = 2$ which means that there are two rows covered by column j exclusively. If the column low_{q1} is the same as low_{q2} ($\{q_1, q_2\} = W_j$), but different

```
1.  Initialize $o_i$ by setting $o_i \leftarrow |S \cap J_i|$, $\forall\, i \in I$.
    // $o_i$ indicates the number of columns covering the row $i$.
2.  for each column $j$ in $S$ (in reverse order in which they locate in $J'$ ) do
    // Consider the columns with high cost first.
3.      Compute $W_j = \{i \in I_j \mid o_i = 1\}$; // $W_j$ contains the rows only covered by column $j$.
4.      If $|W_j| = 0$ then // Column $j$ is redundant
5.          Set $S \leftarrow S \setminus \{j\}$ and $o_i \leftarrow o_i - 1$, $\forall\, i \in I_j$; // Remove column $j$ and update $o_i$.
6.      else if $|W_j| = 1$ and $j \neq low_q$ ($\{q\} = W_j$) then
        // Column $j$ covers row $q$ exclusively and has greater cost than column $low_q$.
7.          Set $S \leftarrow S \setminus \{j\} \cup low_q$, $o_i \leftarrow o_i - 1$, $\forall\, i \in I_j$, and $o_i \leftarrow o_i + 1$, $\forall\, i \in I_{lowq}$;
            // Replace $j$ with $low_q$ and update $o_i$.
8.      else if $|W_j| = 2$ and $j \neq low_{q1} = low_{q2}$ ($\{q_1, q_2\} = W_j$) then
        // The column with minimal cost covering the row $q_1$ is the same as the one
        // covering $q_2$, but different from column $j$.
9.          Set $S \leftarrow S \setminus \{j\} \cup low_{q1}$, $o_i \leftarrow o_i - 1$, $\forall\, i \in I_j$, and $o_i \leftarrow o_i + 1$, $\forall\, i \in I_{lowq1}$;
10.     else if $|W_j| = 2$, $low_{q1} \neq low_{q2}$, and $c_{lowq1} + c_{lowq2} \leq c_j$ then
        // The total cost of $low_{q1}$ and $low_{q2}$ is not greater than $c_j$.
11.         Set $S \leftarrow S \setminus \{j\} \cup low_{q1} \cup low_{q2}$, $o_i \leftarrow o_i - 1$, $\forall\, i \in I_j$, $o_i \leftarrow o_i + 1$, $\forall\, i \in I_{lowq1}$,
            and $o_i \leftarrow o_i + 1$, $\forall\, i \in I_{lowq2}$;
12.     else go to step 2 directly; // Do no operation if $|W_j| \geq 3$.
13.     end if
14. end for
15. Return solution $S$.
```

Fig. 3. Pseudocode for the local search procedure.

from column j, then it replaces column j directly. On the other hand, if the two columns, low_{q1} and low_{q2}, are different and the total cost of them is not higher than the cost of column j, then column j is replaced by these two columns. No operation is done when $|W_j| \geqslant 3$, because limited experimental results indicated that the solution quality can rarely be improved in this case, and the computation cost is relatively high to handle it.

3.6. Framework of the proposed algorithm

As a summary, Fig. 4 describes the whole algorithm presented above. It mainly includes five elements: preprocessing, solving Lagrangian multiplier vector with subgradient method, solution construction, pheromone update, and local search procedure.

4. Experimental study

The performance of Ant-Cover was evaluated experimentally using 65 SCP test instances from Beasley's OR Library (Beasley, 1990). These instances are divided into 11 groups and each group contains 5 or 10 instances. Table 2 shows their detailed information where Density is the percentage of non-zero entries in the SCP matrix. The algorithm was coded in C++ and run on a PC with a 2.0 GHz Pentium IV CPU and 1.0 GB RAM, under Windows XP system. The quality of a solution is evaluated in terms of the relative percentage deviation (RPD) of the objective value Z from Z_{ob} which can be either the optimal or the best known objective value. The relative percentage improvement (RPI) of Z from an initial solution of Ant-Cover Z_I is also reported. These measures are computed as follows:

$$\text{RPD} = (Z - Z_{\text{ob}})/Z_{\text{ob}} \times 100, \tag{18}$$

$$\text{RPI} = (Z_I - Z)/(Z_I - Z_{\text{ob}}) \times 100. \tag{19}$$

Most of the results are expressed in terms of the average RPD values (ARPD), the average RPI values (ARPI), and the average CPU time (ACPU) over the instances tested in each experiment. Furthermore, a composite measure called marginal relative improvement per unit of CPU time (MIC) is used to evaluate differ-

```
1. Preprocess the given SCP instance.
2. Get a Lagrangian multiplier vector with subgradient method.
3. Initialize pheromone trails and related parameters.
4. while termination condition is not met do
5.     for ant $i$ = 1 to $n_a$ (number of ants) do
6.         Construct a solution $S$ based on SROM;
7.         Apply local search to solution $S$ optionally;
8.     end for
9.     Update pheromone trails according to Eq. (14-17).
10.    If the best solution is not improved for $p$ consecutive iterations then
11.        Perform subgradient method and get a new Lagrangian multiplier vector;
12.    end if
13. end while
14. Return the best solution found.
```

Fig. 4. Framework for Ant-Cover.

Table 2
Details of the test instances.

Instance set	No. of instances	m	n	Cost range	Density (%)	Optimal solution
4	10	200	1000	[1, 100]	2	Known
5	10	200	2000	[1, 100]	2	Known
6	5	200	1000	[1, 100]	5	Known
A	5	300	3000	[1, 100]	2	Known
B	5	300	3000	[1, 100]	5	Known
C	5	400	4000	[1, 100]	2	Known
D	5	400	4000	[1, 100]	5	Known
NRE	5	500	5000	[1, 100]	10	Unknown
NRF	5	500	5000	[1, 100]	20	Unknown
NRG	5	1000	10,000	[1, 100]	2	Unknown
NRH	5	1000	10,000	[1, 100]	5	Unknown

ent algorithms. This measure was first introduced by Osman (2004) and has been applied for evaluating many other algorithms (e.g. Osman, 2006). For a given heuristic, MIC is computed as the ratio of the ARPI value over ACPU value. It provides a good way for ranking algorithms by taking into account both solution quality and computational effort.

4.1. Parameter settings

Parameters have a profound influence on the performance of ACO algorithms. However, there exists no analytic method to guide the setting of these parameters so far. Usually, their values are determined in an experimental way which was also used in this research. For each parameter, a set of candidate values were considered. The parameters tested included $\beta \in \{1.0, 2.0, 5.0, 8.0\}$, $\rho \in \{0.90, 0.95, 0.98, 0.99\}$, $\varepsilon \in \{0.001, 0.002, 0.005, 0.010\}$, and $n_a \in \{10, 20, 50, 100\}$. We modified the value of one parameter while keeping the others' fixed. This totally resulted in $4 \times 4 \times 4 \times 4 = 256$ different groups of parameter settings. Finally, we found that when $\beta = 5.0$, $\rho = 0.99$, $\varepsilon = 0.005$, and $n_a = 20$, Ant-Cover performed best. So they were chosen as default settings in the following experiments. It is worth noting that although the parameter setting method described above is simple and easy to implement, it is somewhat tedious since it considers all possible parameter combinations and treats them equally. A good alternative is F-Race (Birattari, Stützle, Paquete, & Varrentrapp, 2002) which tries to systematically configure the components and set parameters for a metaheuristic. While setting parameters, it also sequentially evaluates all the candidates, but discards bad ones as soon as statistically sufficient evidence is gathered against them, so that more force can be focused on the most promising ones.

4.2. Effect of new strategies used in Ant-Cover

The performance of any metaheuristic depends on the employed strategies. This subsection aims to investigate the improvement made by the new developed solution construction method, the heuristic information based on Lagrangian dual information, as well as the local search procedure. For this purpose, the following variants were designed. Unless otherwise mentioned, all these variants have the same settings with Ant-Cover.

V1: An ACO variant using the traditional solution construction method based on Eq. (4) and the traditional heuristic information given in Eq. (9). This version serves as the benchmark for other variants.
V2: Same as V1, but using the single-row-oriented solution construction method (SROM).
V3: Same as V1, but using the heuristic information based on Lagrangian dual information.

The other two variants were Ant-Cover and Ant-Cover with local search (denoted as Ant-Cover + *ls*). All these variants were tested on all 65 SCP instances and given approximately identical runtime for a certain instance type. The maximum CPU time for each instance type and the corresponding results are reported in Table 3. In addition, Fig. 5 plots the ARPD values of each ACO variant and provides an easy way to compare these variants. A number of observations can be made from them.

- The effect of SROM can be illustrated by comparing V1 and V3 with V2 and Ant-Cover, respectively, since each pair have same settings except for the solution construction method. The solution quality of V2 (respectively, Ant-Cover) significantly improves that of V1 (respectively, V3) by decreasing the ARPD values to the amount between 2.83% and 8.36% (respectively, 0.25% and 5.41%).
- Similarly, the effect of the heuristic information based on Lagrangian dual information can be seen from the solutions of

Table 3
Time settings for different instance sets and the ARPD values of each ACO variant.

Instance set	Maximum time (s)	ARPD (%)				
		V1	V2	V3	Ant-Cover	Ant-Cover + *ls*
4	2	5.68	0.64	0.57	0.06	0.00
5	2	5.83	0.59	0.60	0.00	0.00
6	2	6.72	1.35	2.34	0.29	0.00
A	5	6.04	0.97	0.82	0.57	0.09
B	5	4.72	0.77	0.76	0.00	0.00
C	5	9.63	2.82	3.01	0.26	0.00
D	5	8.71	2.47	3.67	0.30	0.00
NRE	20	5.62	2.79	1.33	0.74	0.00
NRF	20	8.79	4.40	5.82	1.54	1.54
NRG	50	10.80	2.79	5.73	0.85	0.37
NRH	50	13.37	5.01	9.68	4.27	1.66
Average		7.81	2.24	3.12	0.81	0.33

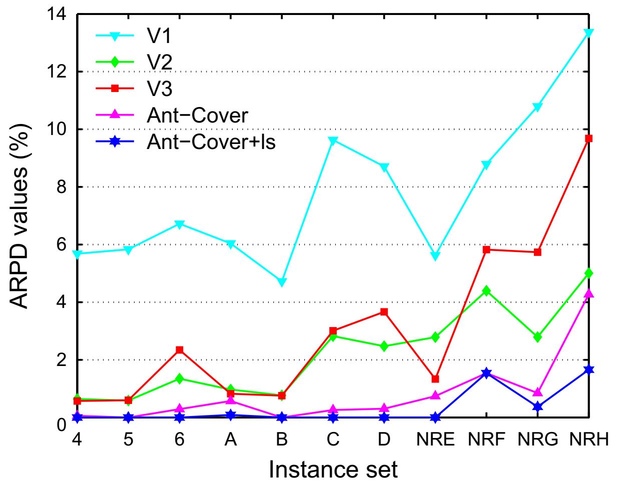

Fig. 5. ARPD values of five ACO variants.

V3 and Ant-Cover. On the average, V3 (respectively, Ant-Cover) decreases the ARPD values of V1 (respectively, V2) by 4.69% (respectively, 1.43%). The main reason for such a performance is that the Lagrangian dual information provides more accurate heuristic values for ants and accordingly guides them to the right solution areas with higher possibility.

- The result that Ant-Cover gets smaller ARPD values than both V2 and V3 indicates the solution construction method and heuristic information adopted in Ant-Cover can complement each other in generating high quality solutions.
- It can also be seen that while local search is used in Ant-Cover, the solution quality can be further improved. This reveals that the local search procedure is effective.

4.3. Comparison with existing ACO-based algorithms

We then compared Ant-Cover with an ACO-based algorithm proposed by Crawford and Castro (2006). It integrates ant colony system with a post processing procedure (denoted as ACS + *pp*) and performs best among all the ACO variants introduced by Crawford and Castro. To the best of our knowledge, it is the newest ACO-based algorithm for SCP in the literature. For comparison, the ACO variant V3 presented in the previous subsection was also used in this experiment. It is actually a MMAS algorithm. Its version with local search is accordingly denoted as MMAS + *ls*. The same test instances used in Crawford and Castro (2006) were adopted here. Each instance was solved 10 times by four algorithms we implemented and a maximum number of 500 iterations were allowed in each run with 10,000 solutions being generated. In contrast to them, ACS + *pp* was run once on each instance and 19,200 solutions were generated in each run. Computational results are given in Table 4. The first part report the optimal solutions to the instances tested and the best solutions found by each algorithm. Note that the results given in the last four columns are average values over 10 runs and the results in bold face indicate that they are optimal. The last three rows report the ARPD values, the average CPU time for first finding the final best solution (ACPU$_S$), and the average total CPU time for completing each run (ACPU$_T$) over 7 instances tested.

From Table 4, it can be observed that MMAS obtained better results than ACS + *pp*, since it got optimal solutions for 4 out of 7 instances in all the 10 runs and ACS + *pp* did not find an optimal solution. When local search is used in MMAS, its solution quality can be further improved. In all the 10 runs, MMAS + *ls* found optimal solutions for all the instances except for 6.1. Ant-Cover showed comparable performance with MMAS + *ls*, but it failed to find the optimal solution for instance 5.1 in some runs. Ant-Cover + *ls* was the only algorithm which got optimal solutions for all the instances in all the 10 runs. As for the ARPD, Ant-Cover + *ls* got the smallest value which confirms its competitiveness in solving the SCP.

Since the detailed computation time of ACS + *pp* is not reported in Crawford and Castro (2006), we just compare the computation time of other four algorithms. From Table 4, it can be seen that the Ant-Cover got smaller ACPU$_T$ value than MMAS while generating same number of solutions. This means that Ant-Cover runs faster than MMAS. On the other hand, the ACPU$_S$ value of Ant-Cover + *ls* (respectively, MMAS + *ls*) is smaller than that of Ant-Cover (respectively, MMAS) which implies that local search can speed up the convergence of Ant-Cover and MMAS. In addition, the comparison between the ACPU$_T$ values of Ant-Cover and Ant-Cover + *ls* (respectively, MMAS and MMAS + *ls*) shows that the computation time consumed by local search procedure contributes little to the total time consumed by Ant-Cover + *ls* (respectively, MMAS + *ls*). Finally, the ACPU$_S$ values of Ant-Cover + *ls* and MMAS + *ls* are much smaller than their ACPU$_T$ values. This suggests that, while using Ant-Cover + *ls* and MMAS + *ls* to solve those relatively small scale instances, the termination condition of iterating 500 times could be reduced without sacrificing solution quality. Due to the superior performance shown by Ant-Cover + *ls* in ARPD and ACPU$_S$ values, it is selected for further investigation.

It is notable that the difference between Ant-Cover and the MMAS algorithm considered in this experiment only lies in the state transition rule. This makes it easier to evaluate the performance of Ant-Cover. However, the performance of MMAS may be slightly improved by further tuning its parameters since the MMAS tested in the experiment above used the same parameters with Ant-Cover.

Table 4
Computational results and computation time on seven benchmark instances.

Instance	Z_{ob}	ACS + *pp*	MMAS	MMAS + *ls*	Ant-Cover	Ant-Cover + *ls*
4.1	429	435	429.6	**429.0**	**429.0**	**429.0**
4.2	512	530	**512.0**	**512.0**	**512.0**	**512.0**
4.8	492	499	**492.0**	**492.0**	**492.0**	**492.0**
5.1	253	265	**253.0**	**253.0**	254.2	**253.0**
6.1	138	143	139.2	138.6	**138.0**	**138.0**
6.2	146	152	146.4	**146.0**	**146.0**	**146.0**
6.3	145	149	**145.0**	**145.0**	**145.0**	**145.0**
Overall average						
ARPD (%)	0.00	3.08	0.18	0.06	0.07	0.00
ACPU$_S$ (s)		–	2.66	1.04	0.78	0.16
ACPU$_T$ (s)		–	8.03	8.18	2.42	2.47

Table 5
Detailed computational results on 65 instances.

Instance	Z_{ob}	GA			SA			Ant-Cover + *ls*				
		Z_{min}	Z_{max}	Z_{avg}	Z_{min}	Z_{max}	Z_{avg}	Z_{min}	Z_{max}	Z_{avg}	T_{avg_S} (s)	T_{avg_T} (s)
4.1	429	**429**	432	429.7	–	–	–	**429**	**429**	**429.0**	0.04	1.74
4.2	512	**512**	**512**	**512.0**	–	–	–	**512**	**512**	**512.0**	0.07	2.26
4.3	516	**516**	**516**	**516.0**	–	–	–	**516**	**516**	**516.0**	0.15	2.31
4.4	494	**494**	502	494.8	–	–	–	**494**	**494**	**494.0**	0.63	2.12
4.5	512	**512**	**512**	**512.0**	–	–	–	**512**	**512**	**512.0**	0.07	2.17
4.6	560	**560**	**560**	**560.0**	–	–	–	**560**	**560**	**560.0**	0.14	2.41
4.7	430	**430**	432	430.2	–	–	–	**430**	**430**	**430.0**	0.04	1.87
4.8	492	**492**	493	492.1	–	–	–	**492**	**492**	**492.0**	0.05	2.17
4.9	641	**641**	650	643.1	–	–	–	**641**	**641**	**641.0**	0.38	2.38
4.10	514	**514**	**514**	**514.0**	–	–	–	**514**	**514**	**514.0**	0.06	1.90
5.1	253	**253**	**253**	**253.0**	–	–	–	**253**	**253**	**253.0**	0.31	2.29
5.2	302	**302**	305	303.5	–	–	–	**302**	**302**	**302.0**	0.89	2.42
5.3	226	228	228	228.0	–	–	–	**226**	**226**	**226.0**	0.08	2.17
5.4	242	**242**	243	242.1	–	–	–	**242**	**242**	**242.0**	0.06	2.28
5.5	211	**211**	**211**	**211.0**	–	–	–	**211**	**211**	**211.0**	0.05	1.70
5.6	213	**213**	**213**	**213.0**	–	–	–	**213**	**213**	**213.0**	0.04	1.97
5.7	293	**293**	**293**	**293.0**	–	–	–	**293**	**293**	**293.0**	0.06	2.22
5.8	288	**288**	289	288.8	–	–	–	**288**	**288**	**288.0**	0.05	2.43
5.9	279	**279**	**279**	**279.0**	–	–	–	**279**	**279**	**279.0**	0.10	2.30
5.10	265	**265**	**265**	**265.0**	–	–	–	**265**	**265**	**265.0**	0.05	2.14
6.1	138	**138**	**138**	**138.0**	–	–	–	**138**	**138**	**138.0**	0.37	2.75
6.2	146	**146**	147	146.2	–	–	–	**146**	**146**	**146.0**	0.08	3.06
6.3	145	**145**	**145**	**145.0**	–	–	–	**145**	**145**	**145.0**	0.22	3.00
6.4	131	**131**	**131**	**131.0**	–	–	–	**131**	**131**	**131.0**	0.04	2.70
6.5	161	**161**	164	161.3	–	–	–	**161**	**161**	**161.0**	0.29	3.02
A.1	253	**253**	254	253.2	–	–	–	**253**	**253**	**253.0**	1.35	4.04
A.2	252	**252**	**252**	**252.0**	–	–	–	**252**	**252**	**252.0**	1.27	4.23
A.3	232	**232**	233	232.5	–	–	–	**232**	233	232.8	1.03	4.14
A.4	234	**234**	**234**	**234.0**	–	–	–	**234**	**234**	**234.0**	0.22	4.13
A.5	236	**236**	**236**	**236.0**	–	–	–	**236**	**236**	**236.0**	0.79	4.13
B.1	69	**69**	**69**	**69.0**	–	–	–	**69**	**69**	**69.0**	0.19	6.11
B.2	76	**76**	**76**	**76.0**	–	–	–	**76**	**76**	**76.0**	0.20	6.42
B.3	80	**80**	**80**	**80.0**	–	–	–	**80**	**80**	**80.0**	0.12	6.72
B.4	79	**79**	**79**	**79.0**	–	–	–	**79**	**79**	**79.0**	0.22	6.63
B.5	72	**72**	**72**	**72.0**	–	–	–	**72**	**72**	**72.0**	0.09	6.20
C.1	227	**227**	229	227.2	–	–	–	**227**	**227**	**227.0**	0.90	6.10
C.2	219	**219**	221	220.0	–	–	–	**219**	**219**	**219.0**	0.65	6.35
C.3	243	**243**	251	246.4	–	–	–	**243**	**243**	**243.0**	3.26	6.36
C.4	219	**219**	220	219.1	–	–	–	**219**	**219**	**219.0**	0.67	6.11
C.5	215	**215**	216	215.1	–	–	–	**215**	**215**	**215.0**	0.59	6.23
D.1	60	**60**	**60**	**60.0**	–	–	–	**60**	**60**	**60.0**	0.73	9.85
D.2	66	**66**	**66**	**66.0**	–	–	–	**66**	**66**	**66.0**	0.54	10.32
D.3	72	**72**	73	72.2	–	–	–	**72**	**72**	**72.0**	1.49	10.47
D.4	62	**62**	**62**	**62.0**	–	–	–	**62**	**62**	**62.0**	1.58	10.35
D.5	61	**61**	**61**	**61.0**	–	–	–	**61**	**61**	**61.0**	0.71	9.72
NRE.1	29	**29**	**29**	**29.0**	**29**	**29**	**29.0**	**29**	**29**	**29.0**	0.38	21.12
NRE.2	30	**30**	31	30.6	**30**	**30**	**30.0**	**30**	**30**	**30.0**	3.14	23.49
NRE.3	27	**27**	28	27.7	**27**	**27**	**27.0**	**27**	**27**	**27.0**	3.39	21.50
NRE.4	28	**28**	**28**	**28.0**	**28**	**28**	**28.0**	**28**	**28**	**28.0**	2.53	23.27
NRE.5	28	**28**	**28**	**28.0**	**28**	**28**	**28.0**	**28**	**28**	**28.0**	0.47	23.72
NRF.1	14	**14**	**14**	**14.0**	**14**	**14**	**14.0**	**14**	**14**	**14.0**	0.95	30.13
NRF.2	15	**15**	**15**	**15.0**	**15**	**15**	**15.0**	**15**	**15**	**15.0**	0.69	28.23
NRF.3	14	**14**	**14**	**14.0**	**14**	**14**	**14.0**	**14**	**14**	**14.0**	1.37	30.70
NRF.4	14	**14**	**14**	**14.0**	**14**	**14**	**14.0**	**14**	**14**	**14.0**	1.52	28.94
NRF.5	13	**13**	14	13.7	**13**	14	13.7	**13**	14	13.5	6.03	27.13
NRG.1	176	**176**	179	177.7	**176**	178	176.6	**176**	**176**	**176.0**	13.06	31.16
NRG.2	154	155	158	156.3	155	156	155.3	**154**	156	155.1	17.99	29.03
NRG.3	166	**166**	169	167.9	**166**	170	167.6	**166**	169	167.3	21.07	30.24
NRG.4	168	**168**	172	170.3	**168**	174	170.7	**168**	170	168.9	18.09	29.73
NRG.5	168	**168**	174	169.4	**168**	170	168.4	**168**	169	168.1	14.62	30.85
NRH.1	63	64	64	64.0	64	64	64.0	64	64	64.0	38.65	71.47
NRH.2	63	64	64	64.0	**63**	64	63.7	**63**	64	63.9	30.15	71.04
NRH.3	59	**59**	60	59.1	**59**	61	59.4	**59**	61	59.4	35.30	69.66
NRH.4	58	**58**	60	58.9	**58**	59	58.9	**58**	59	58.7	29.89	70.38
NRH.5	55	**55**	56	55.1	**55**	**55**	**55.0**	**55**	**55**	**55.0**	17.40	68.23

Table 6
Summarized results on solution quality and computation time.

	GA	SA	Ant-Cover + *ls*
No. of Z_{ob} found in at least one of 10 runs	61/65	18/20	64/65
No. of Z_{ob} found in all the 10 runs	33/65	10/20	55/65
ARPD (%)	0.40	0.72	0.17
ARPI (%)	97.57	96.07	99.06
$ACPU_T$ (s)	1469.90[a]	150.00[b]	14.65[c]
Adj-$ACPU_T$ (s)	1469.90	122.00	424.85
MIC	0.07	0.79	0.23

Adj-$ACPU_T$: Adjusted CPU time to an equivalent average on Silicon Graphics Indigo R4000, 100 MHZ.
[a] CPU time on a Silicon Graphics Indigo R4000, 100 MHZ (15 Mflop/s).
[b] CPU time on a PC with Pentium 100 MHZ (12.2 Mflop/s).
[c] CPU time on a PC with Pentium IV 2.0 GHZ (435 Mflop/s).

4.4. Comparison with other metaheuristics

We also compared Ant-Cover + *ls* with other two metaheuristics, i.e., genetic algorithm (GA) proposed by Beasley and Chu (1996), and simulated annealing (SA) proposed by Brusco et al. (1999).[2] It is worth noting that these two algorithms also use local search to improve their solution quality. Ant-Cover + *ls* was terminated after 500 iterations in this experiment. Table 5 shows the detailed results obtained by three algorithms. Column 2 reports the optimal or the best known solution value (Z_{ob}) of each instance. Z_{min}, Z_{max}, and Z_{avg} indicate the minimum, maximum, and average values of the final best solutions obtained in the 10 runs, respectively. For Ant-Cover + *ls*, we also report the average time for first finding the final best solution of each instance (T_{avg_S}), and the average total time (T_{avg_T}) over 10 runs. The mark '–' means that the corresponding results are not reported. Some summarized results on the solution quality, including the number of Z_{ob}s found in at least one of the 10 runs and in all the 10 runs, the ARPD and ARPI values of Z_{avg} over the instances tested are given in the first part of Table 6.

From the results, it can be observed that Ant-Cover + *ls* got the Z_{ob}s in at least one of the 10 runs for all the instances except for NRH.1. Also, it found the Z_{ob}s for 55 out of total 65 instances tested in all the 10 runs. In addition, it obtained the smallest ARPD value and the greatest ARPI value among three algorithms. All these results show that, compared with GA and SA, Ant-Cover + *ls* is competitive in terms of average solution quality.

As for the computation time, it can be seen from Table 5 that, on the average, Ant-Cover + *ls* can find the final best solutions in less than 3.3 (respectively, 39) seconds for the instances in sets 4–6 and A-D (respectively, sets NRE-NRH). The longest runtime of Ant-Cover + *ls* for completing all the operations on an instance, including preprocessing and 500 iterations is less than 72 s. However, it is difficult to compare the computation time for GA, SA, and Ant-Cover + *ls* since the computational efficiency is affected by many factors such as the implementation language, the compiler, the executing environment including CPU, RAM, and operating system, and it is even affected by the programmer's coding skills. For approximate comparison, a possible way is to transfer the actual computation time from different computers to a benchmark computer according to the performance report by Dongarra (2007). In this report, the performance of different computers is measured in terms of Mflop/s (millions of floating-point operations per second). The $ACPU_T$ row in Table 6 reports the average value of T_{avg_T} for the compared algorithms on their corresponding computers. The adjusted values are given in the Adj-$ACPU_T$ row. The MIC value is also provided in Table 6. It can be seen that Adj-$ACPU_T$ value of Ant-Cover + *ls* is smaller than that of GA, but greater than SA. Although Ant-Cover + *ls* performed best in terms of ARPD and ARPI values, it got smaller MIC value than SA due to its time requirement. It means that if one is looking for a quick solution, SA can be used; on the other hand, if one is looking for a good solution for a strategic design problem, then Ant-Cover would be recommended as it can provide better solution with consuming more CPU time.

[2] The results of two algorithms, called SAHNM and SAHWM, were reported in Brusco et al. (1999). We cite the results of SAHNM since they are a little better than those of SAHWM.

5. Conclusions

In this research, a new ACO-based approach, Ant-Cover, is proposed for the set covering problem. It uses a novel single-row-oriented solution construction method which can generate solutions in a faster way and allows ants to explore broader solution areas. Moreover, the heuristic information adopted in Ant-Cover considers the dual information associated with the still uncovered rows. It mines the specific information of the SCP and provides a good guideline for ants' search in the solution space. In addition, the proposed local search procedure can improve the solutions constructed by ants significantly at a relatively low computational cost.

Computational results show that the proposed algorithm is efficient in generating high quality solutions. Comparisons with some other metaheuristics reveal that it is competitive in solving the SCP. The work done in this study also demonstrates the versatility of ACO and its high potential in solving hard combinatorial optimization problems.

Acknowledgements

The authors would like to thank the editors and the anonymous reviewers for their helpful comments and suggestions. This work is supported by the National Natural Science Foundation of China under Grant No. 60875043, 60905044, and the Doctoral Foundation of Ministry of Education of China under Grant No. 20090201120042.

References

Balas, E., & Carrera, M. C. (1996). A dynamic subgradient-based branch-and-bound procedure for set covering. *Operations Research, 44*(6), 875–890.
Beasley, J. E. (1990). A Lagrangian heuristic for set covering problems. *Naval Research Logistics, 37*(1), 151–164.
Beasley, J. E., & Chu, P. C. (1996). A genetic algorithm for the set covering problem. *European Journal of Operational Research, 94*(2), 392–404.
Birattari, M., Stützle, T., Paquete, L., & Varrentrapp, K. (2002). A racing algorithm for configuring metaheuristics. In W. B. Langdon et al. (Eds.), *Proceedings of the genetic and evolutionary computation conference (GECCO 2002)* (pp. 11–18). San Fransisco: Morgan Kaufmann Publishers.
Blum, C., & Dorigo, M. (2004). The hyper-cube framework for ant colony optimization. *IEEE Transactions on Systems, Man, and Cybernetics – Part B, 34*(2), 1161–1172.
Brusco, M. J., Jacobs, L. W., & Thompson, G. M. (1999). A morphing procedure to supplement a simulated annealing heuristic for cost- and coverage-correlated set-covering problems. *Annals of Operations Research, 86*, 611–627.
Burke, R., Gustafson, S., & Kendall, G. (2002). A survey and analysis of diversity measures in genetic programming. In W. B. Langdon et al. (Eds.), *Proceedings of the genetic and evolutionary computation conference (GECCO 2002)* (pp. 716–723). San Fransisco: Morgan Kaufmann Publishers.
Caprara, A., Fischetti, M., & Toth, P. (1999). A heuristic method for the set covering problem. *Operations Research, 47*(5), 730–743.
Caprara, A., Toth, P., & Fischetti, M. (2000). Algorithms for the set covering problem. *Annals of Operations Research, 98*, 353–371.
Caserta, M. (2007). Tabu search-based metaheuristic algorithm for large-scale set covering problems. In K. F. Doerner et al. (Eds.), *Metaheuristics: Progress in complex systems optimization* (pp. 43–63). New York: Springer.
Ceria, S., Nobili, P., & Sassano, A. (1998). A Lagrangian-based heuristic for large-scale set covering problems. *Mathematical Programming, 81*, 215–228.

Chvatal, V. (1979). A greedy heuristic for the set-covering problem. *Mathematics of Operations Research, 4*(3), 233–235.

Crawford, B., & Castro, C. (2006). Integrating lookahead and post processing procedures with ACO for solving set partitioning and covering problems. In L. Rutkowski et al. (Eds.), *Proceedings of the 8th international conference on artificial intelligence and soft computing (ICAISC 2006)* (pp. 1082–1090). Berlin, Heidelberg: Springer-Verlag.

Dongarra, J. J. (2007). Performance of various computers using standard linear equations software. Technical report, CS-89-85. Computer Science Department, University of Tennessee.

Dorigo, M., Birattari, M., & Stützle, T. (2006). Ant colony optimization: Artificial ants as a computational intelligence technique. *IEEE Computational Intelligence Magazine, 1*(4), 28–39.

Dorigo, M., & Di Caro, G. (1999). The ant colony optimization meta-heuristic. In D. Corne, M. Dorigo, & F. Glover (Eds.), *New ideas in optimization* (pp. 11–32). Maidenhead, UK: McGraw-Hill.

Dorigo, M., & Gambardella, L. M. (1997). Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation, 1*(1), 53–66.

Dorigo, M., Maniezzo, V., & Colorni, A. (1996). The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics – Part B, 26*(1), 29–41.

Dorigo, M., & Stützle, T. (2004). *Ant colony optimization*. Cambridge: MIT Press.

Finger, M., Stützle, T., & Ramalhinho H. (2002). Exploiting fitness distance correlation of set covering problems. In S. Cagnoni et al. (Eds.), *Proceedings of EvoWorkshops 2002* (pp. 61–71). Berlin, Heidelberg: Springer-Verlag.

Fisher, M. L. (1981). The Lagrangian relaxation method for solving integer programming problems. *Management Science, 27*(1), 1–18.

Fisher, M. L., & Kedia, P. (1990). Optimal solution of set covering/partitioning problems using dual heuristics. *Management Science, 36*(6), 674–688.

Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of NP-completeness*. San Francisco, CA: Freeman.

Gutjahr, W. J. (2006). On the finite-time dynamics of ant colony optimization. *Methodology and Computing in Applied Probability, 8*, 105–133.

Hadji, R., Rahoual, M., Talbi, E., & Bachelet V. (2000). Ant colonies for the set covering problem. In M. Dorigo et al. (Eds.), *Proceedings of ANTS 2000* (pp. 63–66).

Housos, E., & Elmoth, T. (1997). Automatic optimization of subproblems in scheduling airlines crews. *Interfaces, 27*(5), 68–77.

Lan, G., & DePuy, G. W. (2006). On the effectiveness of incorporating randomness and memory into a multi-start metaheuristic with application to the set covering problem. *Computer & Industrial Engineering, 51*(3), 362–374.

Lessing, L., Dumitrescu, I., & Stützle, T. (2004). A comparison between aco algorithms for the set covering problem. In M. Dorigo et al. (Eds.), *Proceedings of ANTS 2004* (pp. 1–12). Berlin, Heidelberg: Springer-Verlag.

Osman, I. H. (2004). Metaheuristics: Models, design and analysis. In E. Kozan (Eds.), *Proceedings of the 5th Asia-Pacific industrial engineering and management systems conference* (pp. 1.2–1.16). Australia: Queensland University of Technology.

Osman, I. H. (2006). A tabu search procedure based on a random Roulette diversification for the weighted maximal planar graph problem. *Computers & Operations Research, 33*, 2526–2546.

Solnon, C., & Bridge, D. (2006). An ant colony optimization meta-heuristic for subset selection problems. In N. Nedjah & L. Mourelle (Eds.), *System engineering using particle swarm optimization* (pp. 7–29). New York: Nova Science Publisher.

Solnon, C., & Fenet, S. (2006). A study of ACO capabilities for solving the maximum clique problem. *Journal of Heuristics, 12*, 155–180.

Stützle, T., & Hoos, H. H. (2000). Max–min ant system. *Future Generation Computer Systems, 16*(9), 889–914.

Vasko, F. J., & Wilson, G. R. (1984). Using a facility location algorithm to solve large set covering problems. *Operations Research Letters, 3*(2), 85–90.

Vasko, F. J., & Wolf, F. E. (1987). Optimal selection of ingot sizes via set covering. *Operations Research, 35*(3), 346–353.

Wolsey, L. A. (1998). Lagrangian duality. In L. A. Wolsey (Ed.), *Integer programming* (pp. 167–181). Wiley-Interscience.