A Two-Phase Heuristic for Set Covering

Salim Haddadi

LabSTIC, 8 Mai 1945 University, PO Box 401, 24000 Guelma, Algeria
E-mail: salim.haddadi@yahoo.com

Fatima Guessoum

LabSTIC, 8 Mai 1945 University, PO Box 401, 24000 Guelma, Algeria
E-mail: fatima_guessoum@yahoo.fr

Meryem Cheraitia

LabSTIC, 8 Mai 1945 University, PO Box 401, 24000 Guelma, Algeria
E-mail: meryem.cheraitia@hotmail.fr

Abdellah Salhi

Department of Mathematical Sciences, University of Essex, Wivenhoe Park, Colchester CO4 3SQ, UK E-mail: as@essex.ac.uk

Abstract: The set covering problem (SCP) is a well-known computationally intractable problem. We suggest here a two-phase heuristic to solve it. The first phase reduces substantially the size of the given SCP by removing some variables; the second phase applies a simple Lagrangian heuristic applied to the reduced problem. Construction and improvement heuristics are embedded in the Lagrangian solution approach. The construction heuristic provides good covers by solving small SCPs. The improvement heuristic inserts these covers into larger ones from which better covers are extracted, again by solving different but also small SCPs. The novelty lies in the reduction of the problem size by an effective variable-fixing heuristic, which, in practice, eliminates up to 95% of the variables of the problem without sacrificing the solution quality. Extensive computational and comparative results are presented.

Keywords: Set Covering; Variable-fixing Heuristic; Greedy Heuristic with Regret; Lagrangian Heuristic.

Reference to this paper should be made as follows: Haddadi, S., Cheraitia, M., Guessoum, F. and Salhi, A. (201X) ‘A Two-Phase Heuristic for Set Covering’, *International Journal of Mathematics in Operational Research*, Vol. x, No. x, pp.xxx–xxx.

Biographical notes:
Salim Haddadi received his PhD in Computer Science from Blaise Pascal University, Clermont-Ferrand, France. He is professor of operations research at the Computer Science department, University of Guelma. His areas of interest include combinatorial optimisation, computational complexity, and scheduling.

He has published several research papers at international journals and a book edited by Hermes/Lavoisier.

Meryem Cheraitia and Fatima Guessoum are preparing their PhD in Computer Science at the University of Guelma. Their research interests include issues related to image recognition, data mining, and operational research. They coauthored, with S. Haddadi, a paper published at an international journal, and several conference papers.

Abdellah Salhi is Professor of Operational Research. He obtained his PhD from the University of Aston in Birmingham, UK. His research interests are in the design, analysis, implementation and application of mathematical programming algorithms. Application areas include optimisation, scheduling, decision making under partial information, data-mining and forecasting. Recently, he led a project on workforce scheduling in container terminals, in conjunction with the Port of Felixstowe, UK. He has introduced the Plant Propagation Algorithm for global optimisation, a heuristic inspired by the way strawberry plants propagate using runners. He has published over sixty papers in refereed journals and conference proceedings.

1 Introduction

Consider n positive numbers $c_1, \cdots, c_n$ and a binary $m \times n$-matrix A. Each cost c_j is attached to the j^{th} column of A. We say that column j covers row i if $a_{ij} = 1$. Let $N = \{1, \cdots, n\}$ and $M = \{1, \cdots, m\}$. A cover is any subset $C \subset N$ such that $\sum_{j \in C} a_{ij} \geq 1$ for all $i \in M$. Clearly, a cover exists when A does not have a null row. The cost of the cover C is $\sum_{j \in C} c_j$. SCP aims at finding a minimum cost cover. A well-known integer programming formulation of SCP is

$$\begin{array}{ll} \min \sum_{j \in N} c_j x_j & \\ \sum_{j \in N} a_{ij} x_j \geq 1 & i \in M \\ x_j \in \{0, 1\} & j \in N \end{array} \tag{1}$$

where x is the indicator vector of the cover, i.e.

$$x_j = \begin{cases} 1 \text{ if column } j \text{ is in the cover} \\ 0 \text{ otherwise} \end{cases}$$

We shall use the set C and the vector x to alternatively refer to the cover. A column $j \in C$ is redundant if $C \setminus \{j\}$ is still a cover. A cover that contains a redundant column is called redundant. Let $I_j = \{i \in M | a_{ij} = 1\}, j \in N$, be the set of all rows covered by column j, and $J_i = \{j \in N | a_{ij} = 1\}, i \in M$, be the set of all columns covering row i.

1.1 Well-known facts

SCP is strongly NP-hard, which means that we do not expect a pseudo-polynomial-time algorithm unless $P = NP$. In fact, SCP is hard even on very special instances (Haddadi, 2015). Similarly, we do not expect any polynomial-time approximation scheme unless $P = NP$. Furthermore, the best achievable performance guarantee is $\Theta(\log m)$ (Feige, 1998).

1.2 Applications

Despite these discouraging facts, SCP has been successfully applied in a number of domains such as crew scheduling (Azadeh et al., 2012), location of emergency facilities (Ablanedo-Rosas et al., 2009; Rajagopalan et al., 2008), vehicle routing (Cacchiani et al., 2014), selection of portfolio (Nepal et al., 2009), and conservation biology (Moore et al., 2003).

1.3 Our contributions

They are as follows:

- A new greedy heuristic based on the concept of regret is proposed. We do not study its theoretical performance (recall that we do not expect a better complexity bound than $\Theta(\log m)$). Nevertheless, an empirical analysis tells us that this heuristic is more effective than the well-known greedy heuristic (Chvátal, 1979).
- Applying a subgradient method to the Lagrangian relaxation of the constraints to discard a number of variables, thus reducing the problem size drastically; up to 95% of variables are removed. Solving the reduced problem gives good approximate solutions. This very attractive in practice.
- An effective Lagrangian heuristic is applied to the much smaller resulting SCP.

1.4 Outline of the paper

The paper is organized as follows. Section 2 is a review of the most recent literature on SCP and related topics. Section 3 is on the methodology followed to address the issues of concern. Section 4 concerns the computational experience and reports comparative results of the Lagrangian heuristic put forward here and existing approaches on benchmark problem instances. Section 5 is the conclusion.

2 Related work

This section reviews the most prominent literature related to SCP. Let us begin with the existing exact algorithms. Caprara et al. (2000) surveyed all of them. Surprisingly, they reported that the general-purpose commercial solver Cplex is superior to all of these tree-search algorithms. Since exact methods are applicable only on small to medium instances, to deal with the large scale instances arising in practice, heuristic algorithms are necessary for finding near-optimal covers in a reasonable amount of time.

Roughly speaking, there are two classes of approximate methods, Lagrangian heuristics and metaheuristics. Lagrangian heuristics try to repair the solution of the relaxed problem during the subgradient method. We consider three of them here. The effective algorithm of Caprara et al. (1999) has two main characteristics which are a dynamic pricing scheme for the variables and a systematic use of variable-fixing. Ceria et al. (1998) investigate a Lagrangian-based heuristic for solving large-scale SCPs arising from crew-scheduling at the Italian Railways Company. Haddadi (1997) presents a simple Lagrangian-based heuristic which repairs the solution of the relaxed problem, and then extracts the best cover from it.

Metaheuristics are high-level methods whose purpose is to drive simple low-level heuristics for finding good solutions to an optimization problem, without guaranteeing optimality, by sampling the solution space which is too large to be completely searched. Although effective in identifying optimal or near solutions, they often lack theoretical analysis, and suffer from the non-reproducibility of the results.

In the indirect genetic algorithm of Aickelin (2002), the actual solutions are found by an external decoder and post optimized by hill-climbing. Beasley and Chu (1996) suggest several modifications to a basic genetic algorithm (new fitness-based crossover, variable mutation rate, heuristic feasibility operator). Brusco et al. (1999) use a simulated annealing heuristic in which a morphing procedure is incorporated. The latter enables the replacement of columns in the cover by better ones (morphs). A dynamic primal-dual algorithm is proposed by Caserta (2007), where a tabu search primal method is hybridized with a Lagrangian based dual scheme. A cultural evolutionary architecture is used in Crawford et al. (2014) to maintain knowledge of diversity and fitness learned over each generation during the search. Lan et al. (2007) develop a Meta-RaPS heuristic where solutions are generated and improved. Random factors are introduced in the construction and improvement methods. In the metaheuristic of Naji-Azimi et al. (2010), an initial population is generated. A fixed number of local search and movement iterations based on the electromagnetism metaphor is allowed. Mutation is applied for escaping from local optima. The metaheuristic of Ohlsson et al. (2009) is based on a mean field feedback artificial neural network. In combination with annealing, a set of mean field equations is iterated for obtaining an approximate energy minimum. The approach of Ren et al. (2010) is based on ant colony optimization. Yagiura et al. (2006) use a 3-flip neighborhood where moves consist in cleverly exchanging at most three columns.

3 Solution methodology

We now present the solution approach advocated in this paper. It consists of three components: a new greedy heuristic to construct covers; a variable-fixing heuristic to reduce the size of the SCP problem; and a Lagrangian heuristic to generate a solution to the problem.

3.1 New greedy heuristic

Since we are willing to apply the subgradient method, our goal in this section is to construct a good cover to start with. For this purpose, we propose a new greedy heuristic.

Greedy heuristics are constructive algorithms that build feasible solutions, step by step, by making irreversible decisions. This is where lies their benefit (speed) as well as their drawback (good decisions made at first steps have deplorable consequences in last stages). Our idea here is to incorporate the concept of regret for choosing between alternatives. This idea is not new, and usually results in better quality solutions (see for example Hassin and Keinan (2008) where the regret is incorporated in greedy heuristics for the traveling salesman problem).

The standard greedy heuristic for SCP (Chvátal, 1979) considers the columns with the question: which of the columns is to be picked up first ? The question is solved in the following way. Beginning with an empty cover, 1) compute a score for each column, 2) pick the column with the least score up in the cover, 3) remove the covered rows, and repeat the three steps until all rows are covered.

In the regret version, instead of the columns, we propose to consider the rows with the question: which of the rows is to be covered first ? To deal with this question, we compute a value for each row, call it regret, based on the score of the columns. For computing the regret of row i we consider all the columns covering it. The regret is the difference between the second smallest and the smallest score of these columns. Now, the row with the largest regret value should be covered first by the column having the least score. Details are given in Figure 1.

```
Input: $m, n, c_j, I_j, j \in N$
Output: Cover $x$ and its cost

Step 0  $x_j \leftarrow 0, j \in N$
        $cost \leftarrow 0$
Step 1 // Compute the score $\sigma_j$ of each column $j$
        If $I_j = \emptyset$ then $\sigma_j \leftarrow \infty$ else $\sigma_j \leftarrow c_j / |I_j|, j \in N$
Step 2 // Compute the regret $\rho_i$ of each row $i$
        For $i \in M$, $i$ uncovered
            $a \leftarrow \sigma_{j^*} = \min_{j \in J_i} \sigma_j$
            $b \leftarrow \min_{j \in J_i, j \neq j^*} \sigma_j$
            $\rho_i \leftarrow b - a$
            $store_i \leftarrow j^*$ // Remember the column which covers row $i$
Step 3 // Let $i^*$ be the row with maximum regret
        $i^* \leftarrow \text{argmax}_{i \in M} \sigma_i)$
        $j^* \leftarrow store_{i^*}$
        $x_{j^*} \leftarrow 1$
        $cost \leftarrow cost + c_{j^*}$
        $I_j \leftarrow I_j \setminus I_{j^*}, j \in N$
Repeat steps 1, 2, 3 until $I_j = \emptyset, j \in N$
Return $x, cost$
```

Figure 1 New greedy heuristic.

To understand the rationale behind the concept of regret, suppose that row i^* has the maximum regret. This means, by letting J_{i^*} be the set of all the columns of the binary matrix A covering row i^*, by letting $j^*, \widetilde{j} \in J_{i^*}$ be the columns with respectively the smallest and second smallest score $c_j / |I_j|$, that the value

$$\sigma_{i^*} = \frac{c_{\widetilde{j}}}{\left|I_{\widetilde{j}}\right|} - \frac{c_{j^*}}{|I_{j^*}|}$$

is maximum. Normally we should cover row i^* with column j^* (*i.e.* column j^* enters the cover). Suppose this is not the case, and suppose another column is selected. This column may cover most of the rows covered by column j^*, so that the latter will no longer be selected. So, for covering row i^* we have to call upon a column which is at best as bad as $\widetilde{j}$. Since the "distance" σ_{i^*} between the columns j^* and $\widetilde{j}$ is maximum, we would regret not to choose column j^* first.

As we shall see later, when examining the computational effect of allowing the regret in the greedy approach, we will show the evident superiority of the greedy heuristic with regret.

3.2 Variable-fixing heuristic

Before describing the variable-fixing heuristic, let us recall some relevant results on Lagrangian relaxation and subgradient optimization. Subsequently,

3.2.1 Basic facts on Lagrangian relaxation and subgradient optimization

Given a vector of Lagrangian multipliers $\pi \in \mathbb{R}_+^m$, associated with the m constraints of (1), the Lagrangian relaxation problem is

$$\text{LR}(\pi)\begin{cases} \min z(\pi) = \sum_{j\in N}\left(c_j - \sum_{i\in I_j}\pi_i\right)x_j + \sum_{i\in M}\pi_i \\ x_j \in \{0,1\}, j \in N \end{cases}$$

A detailed description of Lagrangian relaxation and subgradient optimization can be found, for instance, in Umetani and Yagiura (2007). We recall here that, for fixed π, $z(\pi)$ constitutes a lower bound for the optimal value of SCP. Usually, instead of computing the best Lagrangian lower bound, which is the optimal value $z(\pi^*)$ of the Lagrangian dual $\max_{\pi\in\mathbb{R}_+^m} z(\pi)$, we compute a near optimal value z_{LB} by means of the popular iterative subgradient method. This method generates a sequence of vectors $\pi^{(0)}, \pi^{(1)}, \cdots$ and lets $z_{LB} = \max_{k\geq 0} z\left(\pi^{(k)}\right)$.

In every iteration r, for fixed $\pi^{(r)}$, the relaxation problem LR($\pi^{(r)}$) is easily solved by setting for $j \in N$

$$\begin{cases} x_j^{(r)} = 1 & \text{if } c_j - \sum_{i\in I_j}\pi_i^{(r)} < 0 \\ x_j^{(r)} = 0 & \text{if } c_j - \sum_{i\in I_j}\pi_i^{(r)} > 0 \\ x_j^{(r)} = 0 \text{ or } 1 & \text{if } c_j - \sum_{i\in I_j}\pi_i^{(r)} = 0 \end{cases} \tag{2}$$

Typically, a maximum number T of iterations is fixed, and the method takes $O(m \times n \times d)$ time, where d is the number of nonzero entries in A.

3.2.2 The variable-fixing heuristic

This procedure reduces heuristically the number of columns of A (or variables of SCP). Suppose that P is a subset of N with $P = \{j_1, \cdots, j_p\}$. Consider the problem

$$\text{(REDSCP)}\begin{cases} \min \sum_{k\in P} c_k x_k & \\ \sum_{k\in P} a_{ik}x_k \geq 1 & i \in M \\ x_k \in \{0,1\} & k \in P \end{cases}$$

REDSCP is a set covering problem. It may have no feasible solution unless the constraint matrix has no null row. Clearly, it is a restriction of SCP since it is obtained from it by adding the constraints $x_j = 0, j \notin P$. Therefore, any feasible solution of REDSCP can be extended to a feasible solution of SCP. We shall construct REDSCP by specifying the set P, has two significant properties:

(i) It has much less variables than the original SCP ($p \ll n$);

(ii) it potentially contains the optimal cover of the original SCP.

The set P is constructed by applying the subgradient method to the original instance of SCP. When the method finishes, we compute for each index $j \in N$ the value

$$f_j = \frac{1}{T} \sum_{r=0}^{T-1} x_j^{(r)}$$

where $x_j^{(r)}, j \in N$, is obtained in every iteration r by the formula (2). Clearly, $0 \leq f_j \leq 1, j \in N$. The value f_j is the "frequency" of the inclusion of the variable x_j in the relaxed solution. It is in some sense a score function of column j which suggests that the greater the value f_j, the greater our wish to put column j in a set P of promising candidates. Surprisingly, the value f_j is null for most of the columns, which means that the variable x_j is never selected by the subgradient algorithm because the reduced cost $c_j - \sum_{i \in I_j} \pi_i^{(r)}$ is never non-positive. As a consequence of this observation, our idea is to include in REDSCP only the columns j corresponding to nonzero f_j's. Let p be their number. We claim that these columns are the most likely to belong to an optimal cover of the original SCP. We do not have a formal proof of this claim. In fact, if we remove the term "most likely" the claim becomes false for there exist instances for which the constructed REDSCP does not contain the optimal cover. However, the computational experience confirms the claim by showing that REDSCP defined in this way always contains the optimal (or best-known) cover.

The idea of eliminating variables is not new (see Caprara et al. (1999), Ceria et al. (1998)). Usually, the rules of elimination are derived from the reduced costs. They are mathematically correct, and the smaller resulting problem is equivalent to the original problem. In this paper, the approach is different. It is purely heuristic. We discard every variable that is never selected by the subgradient algorithm. Clearly, the reduced problem and the original one are not equivalent. By dealing with the reduced problem we may miss the optimal solution. So, if our goal is to solve exactly SCP, our approach should be avoided. If the goal is to approximate SCP, our heuristic approach results in a very small reduced problem, the solution of which includes, almost always, the best known solution of the original SCP. This will be discussed later in the section devoted to the computational experience. The details of the variable-fixing heuristic are given in a pseudo-code in Figure 2. Since our goal in this paper is to approximate SCP, from now on we shall be concerned only with the much smaller REDSCP.

3.3 Lagrangian heuristic

A general scheme of a Lagrangian heuristic for a combinatorial optimization problem P may be summarized as follows:

1. choose a Lagrangian relaxation of problem P;
2. use the subgradient method to solve the Lagrangian relaxation of P;
3. in each iteration of the subgradient method, try to derive a feasible solution from the relaxed solution.

The third is the crucial step. However, it is not used in our method. Instead, we use the reduced costs to select promising columns. Because we use the dual solution instead of the primal, we continue referring to this kind of approach as Lagrangian heuristic.

```
**Input** $m, n, c_j, I_j, j \in N$
**Output** Set $P$ // For the definition of REDSCP
Initialization
    • Compute $J_i, i \in M$
    • Using the greedy heuristic with regret, compute an upper
      bound $z_{UB}$ on the value of the optimal cover
    • $\pi_i^{(0)} \leftarrow \min_{j \in J_i} c_j / |I_j|, i \in M$
    • $z_{LB} \leftarrow -\infty$
    • $\rho \leftarrow 2$ // $\rho$ is halved every 50 iterations in our implementation
    • $f_j \leftarrow 0, j \in N$
Loop
For iteration $r = 0, \cdots, T-1$
    • Solve problem LR($\pi^{(r)}$)
    • $f_j \leftarrow f_j + x_j^{(r)}, j \in N$
    • If $z_{LB} < z(\pi^{(r)})$ then $z_{LB} \leftarrow z(\pi^{(r)})$
    • // Compute the subgradient vector
      $\sigma_i^{(r)} \leftarrow 1 - \sum_{j \in J_i} x_j^{(r)}, i \in M$
    • // Compute the new Lagrangian multipliers
      $\pi_i^{(r+1)} \leftarrow \max\left\{0, \pi_i^{(r)} + \rho \frac{z_{UB} - z(\pi^{(r)})}{\|\sigma_i^{(r)}\|^2} \sigma_i^{(r)}\right\}, i \in M$
End for
$f_j \leftarrow f_j / T, j \in N$
Return $P = \{j \in N, f_j > 0\}$
```

Figure 2 Variable-fixing heuristic.

Recall that REDSCP is defined with m rows and p columns. Let us redefine for $i \in M$, $J_i = \{j \in P | a_{ij} = 1\}$ to be the set of the columns in P covering row i.

Since we are willing to perform the subgradient method, our first task is to apply the greedy heuristic with regret for computing an upper bound of the optimal value of SCP. Let $CoverSize$ be the size (number of columns) of the greedy cover found. Before describing the Lagrangian heuristic, we begin by presenting its two main components which are performed in every iteration of the subgradient method.

3.3.1 Construction procedure

The construction method begins with an empty cover in which a small subset S of the p columns is inserted. The parameter $\%CoverSizeCons$ controls the number $s = |S|$ of the columns to be inserted with $s = \%CoverSizeCons \times CoverSize$. Half of the columns ($s/2$) is greedily inserted with the regret principle where the costs are replaced with the reduced costs. The second half is randomly selected.

Define $C1 = P \setminus S$ as the set of columns not in the cover, and let $R1$ be the set of rows uncovered by the columns of S. Our task consists of solving the small SCP

$$\text{(scp1)} \begin{cases} \min \sum_{j \in C1} c_j x_j & \\ \sum_{j \in C1} a_{ij} x_j \geq 1 & i \in R1 \\ x_j \in \{0,1\} & j \in C1 \end{cases}$$

Let E be the optimal cover obtained. We have that $C = S \cup E$ is a feasible, possibly redundant, cover. Clearly, the smaller the value of the parameter $\%CoverSizeCons$, the larger the size of problem scp1. There is no need for removing the redundant columns from the cover obtained since this task will be implicitly performed by the improvement heuristic.

3.3.2 Improvement procedure

The improvement heuristic takes as input the cover C provided by the construction procedure and computes a larger cover C' by adding to C a given number of columns from $P \setminus C$. The parameter $\%CoverSizeImp$ controls the number $t = \%CoverSizeImp \times CoverSize$ of columns to be added. Half of the columns $(t/2)$ is inserted using the greedy with regret heuristic with the reduced costs. The second half is randomly selected. Obviously, the larger the number $\%CoverSizeImp$, the larger the size of problem scp2. Clearly, the resulting cover C' is redundant. Next, we extract from it the best cover by solving again a small SCP (see Haddadi (1997)).

Let $C2 \subset C'$ be the set of redundant columns and let $R2 \subset M$ be the set of rows not covered by the columns in $C' \setminus C2$. To extract the best cover from C' we solve the problem

$$\text{(scp2)} \begin{cases} \min \sum_{j \in C2} c_j x_j & \\ \sum_{j \in C2} a_{ij} x_j \geq 1 & i \in R2 \\ x_j \in \{0,1\} & j \in C2 \end{cases}$$

Let V be the optimal cover found. Then $C' \setminus C2 \cup V$ is the best cover extracted from C'. Clearly, it is at least as good as C. This improvement heuristic turns out to be effective, as we shall see later.

The Lagrangian heuristic consists of a basic subgradient method where the construction and the improvement procedures are embedded. Details are given in Figure 3.

4 Computational experience

The section reports our extensive computational experiments with benchmark instances, and proposes to compare our approach with the state-of-art. All implementation details will be provided to make our method reproducible.

4.1 Experimental setup

Our approach is coded in C and run on an Intel Pentium Dual Core, 2GHz. The web site http://people.brunel.ac.uk/˜mastjjb/jeb/ provides the benchmark instances, which are grouped in eight sets with five instances in each. These instances are well-known and widely used to evaluate and compare exact and heuristic methods. Their characteristics are presented in Table 1. The metaheuristics used for comparison are summarized in Table 2.

```
Input Data of problem REDSCP, $\%CoverSizeCons$, $\%CoverSizeImp$
Output Best cover $C^*$ and its cost $z_{UB}$
Initialization
    • Using the greedy heuristic with regret, compute a cover $C^*$
      and its cost $z_{UB}$
    • $s \leftarrow \%CoverSizeCons \times CoverSize$
    • $t \leftarrow \%CoverSizeImp \times CoverSize$
    • $\pi_i^{(0)} \leftarrow \min_{j \in J_i} c_j / |I_j|, i \in M$
    • $\rho \leftarrow 2$
Loop
For iteration $r = 0, \cdots, T-1$
    • Using the greedy heuristic with regret, insert $s/2$ columns in an empty set $S$
    • Randomly insert $s/2$ columns in $S$
    • Define and solve problem scp1, and let $C$ be the feasible cover found
    • Update $C^*$ and its cost $z_{UB}$
    • Construct a cover $C'$ by adding in a greedy with regret manner $t/2$ columns
      from $P \backslash C$ to $C$
    • Add $t/2$ randomly chosen columns from $P \backslash C$ to $C'$
    • Extract the best cover from $C'$ by solving problem scp2
    • Update $C^*$ and its cost $z_{UB}$
    • Solve the relaxed problem
    • Compute the subgradient vector
    • Compute the Lagrangian multipliers
Return $C^*$ and $z_{UB}$
```

Figure 3 Lagrangian heuristic.

4.2 Detailed results of the greedy heuristic with regret

The results and comparison with the greedy heuristic of Chvátal (1979) are shown in Table 3. The labels of the columns are self-explanatory. We show that the greedy heuristic with regret almost always provides much better covers with smaller size. The average deviation of the two greedy heuristics from best are respectively 14.46% and 7.98%. This tells that the greedy heuristic with regret is almost twice more effective.

4.3 Detailed results of the variable-fixing heuristic

During the variable-fixing heuristic, the number of iterations of the subgradient method is fixed to 200. The relaxation coefficient ρ starts with the value 2 and is halved every 50 iterations (Note that we are not interested in the value of the lower bound). The result of the variable-fixing heuristic is the set P used for defining REDSCP.

Table 4 shows the results of the variable-fixing heuristic, where the columns labeled p and % red. refer to the number of columns of REDSCP and the size reduction percentage

Name	Number of instances	Density (%)	Number of rows	Number of columns	Optimal solution ?
A	5	2	300	3000	yes
B	5	5	300	3000	yes
C	5	2	400	4000	yes
D	5	5	400	4000	yes
NRE	5	10	500	5000	no
NRF	5	20	500	5000	no
NRG	5	2	1000	10000	no
NRF	5	5	1000	10000	no

Table 1 Characteristics of the instances

Label	Methodology	Reference
AICK	Indirect genetic algorithm	Aickelin (2002)
LAN	Simple heuristic	Lan et al. (2007)
OHLS	Mean field approach	Ohlsson et al. (2009)
REN	Ant colony optimization	Ren et al. (2010)
YAGI	3-flip neighborhood local search	Yagiura et al. (2006)
HADD	Our approach	

Table 2 Heuristics in competition.

respectively. It can be seen that the computing times are small (less than one second) and that the resulting REDSCP has a much smaller number of columns than the original SCP. For instance, while the original number of variables of problem instance NRH1 is $10,000$, the corresponding REDSCP has only 639 variables.

To be convinced that REDSCP contains the best known cover, it is submitted to Cplex 12.6, a general-purpose commercial solver, within a time-limit of $18,000$ seconds. This step is intended only as a “certificate”, and is not considered as a method for approximating SCP, although the instances in the data sets A, B, C and D, are satisfactorily solved (see Table 4 where the symbol $-$ means that the cost of the cover found by Cplex is equal to the cost of the best known). The term ‘Limit’ in the column labeled ‘Time’ means that Cplex reaches the fixed time-limit without finishing computing. In the remaining of the paper, we shall deal only with the data sets NRE to NRH, which are the largest and the hardest, noting that our heuristic identifies all the best covers for the instances in the data sets A, B, C and D.

4.4 Parameter settings and results of the Lagrangian heuristic

The number of iterations of the subgradient method is fixed to 200. A good value of the parameter $\%CoverSizeCons$ is 0.2 or 20%. It is obtained as a compromise between the quality of the cover and its solution time. A smaller value (0.1 for example) gives better constructed covers, at the expense of larger problem scp1. In the same manner, we found that a good value of the parameter $\%CoverSizeImp$ is 1.2. A larger value (1.5 for example) gives better improved covers by solving a larger problem scp2. Typically, problem scp1 has a small number of constraints while problem scp2 has a small number of variables.

In our implementation, problems scp1 and scp2 are not solved exactly (this would take a large computing time) but just approximated by using Cplex during a small time-limit.

		Greedy		Greedy w. regret				Greedy		Greedy w. regret	
	Best	Cost	Size	Cost	Size		Best	Cost	Size	Cost	Size
A1	253	288	89	260	66	B1	69	77	45	78	40
A2	252	284	88	269	72	B2	76	86	47	84	40
A3	232	270	91	249	67	B3	80	89	47	83	40
A4	234	278	89	254	68	B4	79	89	50	83	43
A5	236	271	89	259	74	B5	72	78	45	77	42
		15.31%		7.03%				11.40%		7.86%	
C1	227	257	103	241	83	D1	60	74	52	69	43
C2	219	258	104	232	81	D2	66	74	51	71	44
C3	243	276	96	265	78	D3	72	83	52	78	47
C4	219	257	99	238	75	D4	62	71	55	63	43
C5	215	233	96	227	79	D5	61	69	51	64	42
		13.76%		7.08%				15.67%		7.48%	
NRE1	29	30	30	31	28	NRF1	14	16	16	15	15
NRE2	30	36	32	33	29	NRF2	15	16	15	16	16
NRE3	27	30	29	31	28	NRF3	14	17	17	15	14
NRE4	28	32	32	29	26	NRF4	14	17	17	16	15
NRE5	28	33	32	30	28	NRF5	13	16	16	15	15
		13.34%		8.48%				17.38%		10.12%	
NRG1	176	203	130	189	108	NRH1	63	76	67	69	56
NRG2	154	182	129	163	108	NRH2	63	74	65	68	58
NRG3	166	192	130	184	110	NRH3	59	65	60	64	58
NRG4	168	191	127	177	106	NRH4	58	69	62	62	55
NRG5	168	194	127	181	110	NRH5	55	63	60	60	58
		15.67%		7.37%				16.36%		8.38%	

Table 3 Comparison of the greedy heuristics

To make this time-limit depend on the instance size we define it to be $M/2000$. This way, the computing time of our heuristic can be anticipated. For example, for an instance with $M = 1000$, a simple count shows that our method requires an amount of computing time of about 200 seconds, since there are 200 iterations, and in every iteration two small SCPs are approximated within half a second each. The computational results of the Lagrangian heuristic are shown in Table 5 where the symbol $-$ has the same meaning.

4.5 Comparison with existing heuristics

As already shown in Table 2, five recent meta-heuristics are compared with our method, called HADD. We do not re-implement them, but just use the published results which are reported in Table 6 where the symbol $-$ has the same meaning. The computing times are reported in seconds and are related to the machine on which the method is run. Note that Lan et al. (2007) and Ohlsson et al. (2009) report only the solution time, but not the total computing time, so that their methods cannot be compared with ours from the execution speed point of view. The times reported in the methods AICK, OHLS, REN, and YAGI, are average times over ten runs.

	Variable-fixing				solving REDSCP			Variable-fixing				solving REDSCP	
	p	% red.	Time	Best	Cost	Time		p	% red.	Time	Best	Cost	Time
A1	259	91.37	0.08	253	–	0.23	B1	178	94.07	0.08	69	–	0.23
A2	268	91.07	0.07	252	–	0.28	B2	221	92.63	0.08	76	–	0.55
A3	284	90.53	0.07	232	–	0.22	B3	200	93.33	0.10	80	–	0.22
A4	287	90.43	0.05	234	–	0.17	B4	219	92.70	0.11	79	–	0.59
A5	268	91.07	0.08	236	–	0.17	B5	185	93.83	0.09	72	–	0.27
C1	321	91.98	0.11	227	–	0.25	D1	251	93.73	0.15	60	–	0.45
C2	346	91.35	0.11	219	–	0.31	D2	238	94.05	0.12	66	–	1.12
C3	354	91.15	0.08	243	–	0.66	D3	276	93.10	0.14	72	–	0.83
C4	348	91.30	0.09	219	–	0.30	D4	238	94.05	0.13	62	–	0.72
C5	302	92.45	0.10	215	–	0.36	D5	215	94.63	0.12	61	–	0.39
NRE1	242	95.16	0.40	29	–	19.27	NRF1	269	94.62	0.85	14	–	24.74
NRE2	309	93.82	0.41	30	–	277.28	NRF2	242	95.16	0.82	15	–	13.90
NRE3	267	94.66	0.43	27	–	24.35	NRF3	290	94.20	0.83	14	–	11.39
NRE4	264	94.72	0.40	28	–	30.50	NRF4	281	94.38	0.84	14	–	68.87
NRE5	277	94.46	0.39	28	–	13.81	NRF5	278	94.44	0.83	13	–	456.94
NRG1	663	93.37	0.34	176	–	4694.80	NRH1	639	93.61	0.86	63	–	Limit
NRG2	638	93.62	0.35	154	–	630.95	NRH2	600	94.00	0.87	63	–	Limit
NRG3	625	93.75	0.36	166	–	10815.64	NRH3	532	94.68	0.86	59	–	Limit
NRG4	613	93.87	0.36	168	–	Limit	NRH4	580	94.20	0.85	58	–	Limit
NRG5	636	93.64	0.34	168	–	Limit	NRH5	541	94.59	0.82	55	–	Limit

Table 4 Details of the variable-fixing heuristic

	Best	Cost	Time		Best	Cost	Time
NRE1	29	–	85.32	NRF1	14	–	69.31
NRE2	30	–	108.05	NRF2	15	–	42.22
NRE3	27	–	95.33	NRF3	14	–	78.07
NRE4	28	–	106.04	NRF4	14	–	105.20
NRE5	28	–	55.71	NRF5	13	–	109.92
NRG1	176	–	166.38	NRH1	63	–	208.33
NRG2	154	–	182.11	NRH2	63	–	205.74
NRG3	166	–	189.63	NRH3	59	–	205.51
NRG4	168	–	190.21	NRH4	58	–	203.44
NRG5	168	–	196.77	NRH5	55	–	200.13

Table 5 Computational results on NRE-NRH instances.

From the point of view of accuracy, it can be seen in Table 8 that the methods LAN, YAGI and HADD are effective since they are all able to find the optimal or best-known solution, although a single run of YKI may fail. The methods REN and AICK are less effective since they fail on one and three instances respectively, while the method OHLS can be considered as inefficient.

It remains to compare the methods from the execution speed point of view. As usual, these comparisons are hard to perform because different platforms, languages and operating systems are used (see Table 7). Computing times need to be scaled before we can compare them. The reference (Dongarra, 2011) evaluates the performance of various computers using linear algebra software. Although thousands of computers are tested, the machines used in the comparison do not exist in the database. Nevertheless, similar machines, when found, are taken into account. From the information provided by Dongarra (2011), if we consider the Pentium II (400 MHz) as reference, rough scaling factors are shown in Table 7, and the scaled times in Table 8.

From this comparison, our method turns out to be almost five times slower. This is due to the use of Cplex, instead of a faster heuristic, for approximating problems scp1 and scp2. On the other hand, average computing times seem not to be a fair way to count the computing times of a method (AICK, OHLS, REN, YAGI) which is run ten times.

		AICK		LAN		OHLS		REN		YAGI		HADD	
	Best	Cost	Time	Cost	Time	Cost	Time	Cost	Time	Cost	Time	Cost	Time
NRE1	29	–	17	–		–		–	21.12	–	180	–	85.32
NRE2	30	–	63	–		32		–	23.49	–	180	–	108.05
NRE3	27	–	60	–		28		–	21.50	–	180	–	95.33
NRE4	28	–	41	–		29		–	23.27	–	180	–	106.04
NRE5	28	–	99	–		29		–	23.72	–	180	–	55.71
NRF1	14	–	21	–		–		–	30.13	–	180	–	69.31
NRF2	15	–	44	–		–		–	28.23	–	180	–	42.22
NRF3	14	–	234	–		15		–	30.70	–	180	–	78.07
NRF4	14	–	174	–		15		–	28.94	–	180	–	105.20
NRF5	13	–	241	–		14		–	27.13	–	180	–	109.92
NRG1	176	–	144	–		180		–	31.16	–	180	–	166.38
NRG2	154	155	327	–		157		–	29.03	–	180	–	182.11
NRG3	166	–	408	–		173		–	30.24	–	180	–	189.63
NRG4	168	–	303	–		175		–	29.73	–	180	–	190.21
NRG5	168	–	532	–		175		–	30.85	–	180	–	196.77
NRH1	63	–	668	–		65		64	71.47	–	180	–	208.33
NRH2	63	66	443	–		66		–	71.04	–	180	–	205.74
NRH3	59	–	648	–		62		–	69.66	–	180	–	205.51
NRH4	58	59	235	–		60		–	70.38	–	180	–	203.44
NRH5	55	–	66	–		56		–	68.23	–	180	–	200.13

Table 6 Results of the competitive heuristics.

5 Conclusion

In this paper, we proposed:

	Machine used	Frequency	Similar machine found in the database	Theoretical peak (Mflops)	Scaling factor	Number of runs
AICK	Pentium II	450 MHz	Intel Pentium II Xeon 450 MHz	450	1.13	10
LAN	Pentium IV	1.7 GHz	Not found	Not known		1
OHLS	Pentium II	400 MHz	Not found	400 (expected)	1.00	10
REN	Pentium IV	2.0 GHz	Not found	Not known	10.00 ?	10
YAGI	SUN Ultra2 2300		Sun UltraSparc II 300 MHz	600	1.50	10
HADD	Pentium Dual Core	2.0 GHz	HP DL385 2.2 GHz (dual core) Opteron 275	4000 (expected)	10.00	1

Table 7 Machines used.

	AICK	LAN	OHLS	REN	YAGI	HADD
Number of best solutions found	17	20	3	19	20	20
Average computing time	238.40			38.00	180.00	140.17
Average scaled computing time	269.39			380.00	270.00	1401.70

Table 8 Comparison of the heuristics.

1. a greedy heuristic with regret which turns more effective than the greedy heuristic of Chvátal (1979);

2. an effective and promising variable-fixing procedure, eliminating up to 95% of the variables of the problem, without sacrificing solution quality;

3. a simple and easy to implement Lagrangian heuristic, competitive with the state-of-the-art.

The main two conclusions that can be drawn are the following:

- If we are interested in approximating SCP, we have just to deal with the reduced problem which represents a small piece of the entire original SCP. But as shown, it always contains the best-known cover. Furthermore, any successful heuristic would benefit from this size reduction and applies much faster. This may be appealing for practitioners, since instead of dealing with the whole data, one's attention can be restricted to the reduced problem.

- The idea underlying the definition of REDSCP, from the information provided by the subgradient method, is generic enough and can potentially be applied to any combinatorial optimization problem.

The following are some limitations of our approach. Solving problems scp1 and scp2 with Cplex was not a good idea. The Lagrangian solution approach can run faster if a good heuristic is used instead of Cplex. Second, our heuristic is tailored for non-unicost SCP instances (unicost set covering instances are those with identical costs). RAIL instances are well-known very large scale instances arising from crew scheduling in the Italian Railway company, with the costs being 1 or 2. Because precisely of the cost structure, we unfortunately found that our heuristic performs poorly on unicost and RAIL instances, since can remove only a small part of the columns during the variable-fixing phase. Without removing a large part of the latter, problem scp1 would be very large to be approximated by using Cplex within a reasonable amount of time.

As a direction for future research, we will be focusing on the design and implementation of a fast heuristic for approximating problems scp1 and scp2. It would also be interesting to study the theoretical performance of the greedy heuristic with regret.

Acknowledgement

The authors are grateful to the two anonymous referees whose comments and suggestions contribute to greatly improve the presentation and the readability of the manuscript.

References

Ablanedo-Rosas, J.H., Gao, H., Alidaee, B. and Teng W.Y.W. (2009) ‘Allocation of emergency and recovery centres in Hidalgo, Mexico’. *International Journal of Services Sciences*, Vol. 2, No. 2, pp.206–218.

Aickelin, U. (2001) ‘An indirect genetic algorithm for set covering problems’. *Journal of the Operational Research Society*, Vol. 53, No. 10, pp.1118–1126.

Azadeh, A., Asadipour, G., Eivazy, H. and Nazari-Shirkouhi, S. (2012) ‘A unique hybrid particle swarm optimisation algorithm for simulation and improvement of crew scheduling problem’. *International Journal of Operational Research*, Vol. 13, No. 4, pp.406–422.

Bautista, J. and Pereira, J. (2007) ‘A GRASP algorithm to solve the unicost set covering problem’. *Computers and Operations Research*, Vol. 34, No. 10, pp.3162–3173.

Beasley, J.E. and Chu, P.C. (1996) ‘A genetic algorithm for set covering problems’. *European Journal of Operational Research*, Vol. 94, No. 2, pp.392–404.

Brusco, E., Jacobs, L.W. and 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*, Vol. 86, No. 0, pp.611–627.

Cacchiani, V., Hemmelmayr, V.C. and Tricoire, F. (2014) ‘A set-covering based heuristic algorithm for the periodic vehicle routing problem’. *Discrete Applied Mathematics*, Vol. 163, pp.53–64.

Caprara, A., Fischetti, M. and Toth, P. (1999) ‘A heuristic method for the set covering problem’. *Operations Research*, Vol. 47, No. 5, pp.730–743.

Caprara, A., Toth, P. and Fischetti, M. (2000) ‘Algorithms for the set covering problem’. *Annals of Operations Research*, Vol. 98, No. 1–4, pp.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), Springer, New-York.

Ceria, S., Nobili, P. and Sassano, A. (1998) 'A Lagrangian-based heuristic for large scale set covering problems'. *Mathematical Programming*, Vol. 81, No. 2, pp.215–228.

Chvátal, V. (1979) 'A greedy heuristic for set-covering problem'. *Mathematics of Operations Research*, Vol. 4, No. 3, pp.233–235.

Crawford, B., Soto, R. and Monfroy, E. (2014) 'Cultural Algorithms for the Set Covering Problem'. *Lecture Notes in Computer Science*, Vol. 7929 , pp.27–34.

Dongarra J.J. (2011) Performance of various computers using standard linear equations software. [online] Technical report CS-89-85, University of Manchester. http://ash2.icl.utk.edu/sites/ash2.icl.utk.edu/files/publications/2006/icl-utk-282-2006.pdf (Accessed December 2015).

Feige U. (1998) 'A threshold of $log\ n$ for approximating set cover'. *Journal of the ACM*, Vol. 45, No. 4, pp.634–652.

Haddadi S. (1997) 'Simple Lagrangian heuristic for the set covering problem'. *European Journal of Operational Research*, Vol. 97, No. 1, pp.200–204.

Haddadi S. (2015) 'Benders decomposition for set covering problems almost satisfying the consecutive ones property'. *Journal of Combinatorial Optimization*, doi:10.1007/s10878.015-9935.

Hassin, R. and Keinan, A. (2008) 'Greedy heuristics with regret, with application to the cheapest insertion algorithm for the TSP'. *Operations Research Letters*, Vol. 36, No. 2, pp.243–246.

Lan, G., DePuy, G.W. and Whitehouse, G.E. (2007) 'An effective and simple heuristic for the set covering problem'. *European Journal of Operational Research*, Vol. 176, No. 3, pp.1327–1333.

Moore, J.L., Folkmann, M., Balmford, A., Brooks, T., Burgess, N., Rahbek, C., Williams, P.H. and Krarup, J. (2003) 'Heuristic and optimal solutions for set-covering problems in conservation biology'. *Ecography*, Vol. 26, No. 5, pp.595–601.

Naji-Azimi, Z., Toth, P. and Galli, P. (2010) 'An electromagnetism metaheuristic for the unicost set covering problem'. *European Journal of Operational research*, Vol. 205, No. 2, pp.290–300.

Nepal, B., Lassan, G., Drow, B. and Chelst, K. (2009) 'A set-covering model for optimizing selection of portfolio of microcontrollers in an automotive supplier company'. *European Journal of Operational Research*, Vol. 193, No. 1, pp.272–281.

Ohlsson, M., Peterson, C. and Södeberg, B. (2009) 'An efficient mean field approach to the set covering problem'. *European Journal of Operational Research*, Vol. 133, No. 3, pp.583–595.

Rajagopalan, H.K., Saydam, C. and Xiao, J. (2008) 'A multiperiod set covering location model for dynamic redeployment of ambulances'. *Computers and Operations Research*, Vol. 35, No. 3, pp.814–826.

Ren, Z.G., Feng, Z.R., Ke, L.J. and Zhang, Z.J. (2010) 'New ideas for applying ant colony optimization to the set covering problem'. *Computers and Industrial Engineering*, Vol. 58, No. 4, pp.774–784.

Umetani, S. and Yagiura, M. (2007) 'Relaxation heuristics for the set covering problem'. *Journal of the Operational Research Society of Japan*, Vol. 50, No. 4, pp.350–375.

Yagiura, M., Kishida, M. and Ibaraki, T. (2006) 'A 3-flip neighborhood local search for the set covering problem'. *European Journal of Operational Research*, Vol. 172, No. 2, pp.472–499.