Hindawi Publishing Corporation
ISRN Operations Research
Volume 2013, Article ID 203032, 10 pages
http://dx.doi.org/10.1155/2013/203032

*Research Article*

# A New Formulation of the Set Covering Problem for Metaheuristic Approaches

**Nehme Bilal, Philippe Galinier, and Francois Guibault**

*École Polytechnique de Montréal, C.P. 6079, Succ. Centre-Ville, Montréal, QC, Canada H3C 3A7*

Correspondence should be addressed to Nehme Bilal; nehmebilal@gmail.com

Received 18 February 2013; Accepted 4 April 2013

Academic Editors: L. Buzna, P. Ekel, C. Mohan, and M. Wang

Two difficulties arise when solving the set covering problem (SCP) with metaheuristic approaches: solution infeasibility and set redundancy. In this paper, we first present a review and analysis of the heuristic approaches that have been used in the literature to address these difficulties. We then present a new formulation that can be used to solve the SCP as an unconstrained optimization problem and that eliminates the need to address the infeasibility and set redundancy issues. We show that all local optimums with respect to the new formulation and a 1-flip neighbourhood structure are feasible and free of redundant sets. In addition, we adapt an existing greedy heuristic for the SCP to the new formulation and compare the adapted heuristic to the original heuristic using 88 known test problems for the SCP. Computational results show that the adapted heuristic finds better results than the original heuristic on most of the test problems in shorter computation times.

## 1. Introduction

The set covering problem (SCP) is a popular optimization problem that has been applied to a wide range of industrial applications, including scheduling, manufacturing, service planning, and location problems [1–4]. The SCP is *NP hard* in the strong sense [5]. The mathematical formulation of the SCP is as follows. Let $E = \{e_1, \dots, e_m\}$ be a universe of elements, and let $S = \{s_1, \dots, s_n\}$ be a collection of subsets $s_j \subseteq E$, where $\bigcup s_j = E$. Each set $s_j$ covers at least one element of $E$ and has an associated cost $c_j > 0$. The objective is to find a subcollection of sets $X \subseteq S$ that covers all of the elements in $E$ at a minimal cost. The mathematical programming model of the SCP is usually formulated as follows.

(i) Let $A^{m\times n}$ be a zero-one matrix where $a_{ij} = 1$ if element $i$ is covered by set $j$ and $a_{ij} = 0$ otherwise.

(ii) Let $X = \{x_1, x_2, \dots, x_n\}$ where $x_j = 1$ if set $j$ (with cost $c_j > 0$) is part of the solution and $x_j = 0$ otherwise.

Minimize

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

subject to

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

$$x_j \in \{0, 1\}. \tag{3}$$

The objective function (1) drives the search toward solutions at minimal cost. Constraint (2) (full coverage constraint) imposes the requirement that all the elements of the universe $E$ must be covered. If constraint (2) is not satisfied, the solution is infeasible. If constraint (2) is satisfied and the objective function is minimized, the solution will cover all of the elements at the minimal cost (optimal solution). If constraint (2) is relaxed, the objective function will drive the search toward an empty solution because the empty solution has the lowest cost (0). These observations show that the objective function and the full coverage constraint of the SCP guide the search in two opposite directions.

When solving the model with metaheuristic algorithms, two issues arise: solution infeasibility and set redundancy. A solution to the SCP is considered to be infeasible if one or more of the elements of the universe $E$ are uncovered. A set is considered to be redundant if all the elements covered by the set are also covered by other sets in the solution.

In this paper, we first review and analyze the literature to highlight the difficulties in dealing with solution infeasibility and set redundancy when solving the SCP with metaheuristic algorithms (Section 2). We then present a new formulation that can be used to solve the SCP as an unconstrained optimization problem and that eliminates the need for addressing the infeasibility and redundancy issues (Section 3). The new formulation uses a maximization objective that can replace both the cost minimization objective and the full coverage constraint of the classical formulation. The new formulation can also be seen as a new penalty approach that has many advantages over the existing penalty approaches (Section 3.2) for the SCP. Third, we present a simple descent heuristic that is based on the new formulation and that uses a simple 1-flip neighbourhood structure. The proposed descent heuristic is an adaptation of an existing greedy heuristic for the SCP. We show that all local optimums with respect to the new formulation and the 1-flip neighbourhood structure are feasible and free of redundant sets. Finally, the proposed descent heuristic is compared to the original greedy heuristic using 88 known set covering problems (Section 5).

## 2. Literature Review

In general, metaheuristic algorithms can be divided into three categories.

(i) *Constructive metaheuristics*: in each iteration, a new local optimum is found by constructing a new solution from scratch. A level of randomness is added to the construction step in order to avoid constructing the same solution over and over.

(ii) *Evolutionary algorithms*: in each iteration, two or more solutions are combined to create a new solution.

(iii) *Local search*: in each iteration, the current solution is replaced by one of its immediate neighbors (the solution is usually modified slightly).

In the following sections, we review the literature of solving the SCP with metaheuristic approaches and analyze how each category of metaheuristics addresses *solution infeasibility* and *set redundancy*.

*2.1. Constructive Metaheuristics.* When the SCP is solved with constructive metaheuristics, the local optimums found at the end of each constructive iteration are usually feasible. In fact, the constructive iteration ends when all of the elements are covered. For this reason, these metaheuristics do not have to deal with the infeasibility issue. However, the local optimums are not necessarily free of redundant sets, and a redundancy removal heuristic is needed. Constructive metaheuristics for the SCP includes ant colony optimization [6–10], Meta-RaPS [11], and GRASP [12]. All of these metaheuristics use a dedicated redundancy removal operator that removes redundant sets at the end of each iteration.

*2.2. Evolutionary Algorithms.* Evolutionary algorithms for the SCP need to address both infeasibility and set redundancy issues. Most evolutionary algorithms that are used to solve the SCP are based on the genetic algorithm (GA). Most of the GAs use a binary string solution representation where $x_j = 1$ if the set $s_j$ is part of the solution and $x_j = 0$ otherwise. The infeasibility issue arises when the crossover or mutation operator of the GA produces a child (solution) that does not cover all of the elements. In fact, a simple bit flip from 1 to 0 during crossover or mutation can produce an infeasible solution. If a cost minimization objective function is used, infeasible solutions will be preferred over feasible ones because infeasible solutions are usually cheaper. Two main approaches have been used in the literature to address the infeasibility issue.

The first approach uses a repair heuristic to transform infeasible solutions to feasible solutions before the evaluation step of the GA. A greedy-like repair heuristic is usually used [13–15]. In each iteration, the repair operator covers an uncovered element by selecting a new set that covers the element and adding it to the solution. In [15], all of the solutions are repaired for evaluation, but only 5% of them are replaced with the corresponding repaired versions. The aim is to allow the search to explore infeasible regions of the search space, which tend to be more effective than limiting the search to only feasible regions. A simpler repair heuristic is used in [16]. During the evaluation of a solution, a set is added to the solution if it covers an uncovered element(s) and is not already part of the solution. By adding new sets, repair heuristics may introduce redundant sets into the solutions. For this reason, genetic algorithms that use a repair operator also use a redundancy removal procedure that is applied after the repair and just before evaluation.

The second approach involves penalizing the objective value of infeasible solutions to drive the search toward the feasible region. A penalty term that makes infeasible solutions less attractive than feasible ones is added to the objective function. In [17], the same penalty $M$ is added to the objective value of all infeasible solutions. $M$ is high enough to guarantee that all feasible solutions have lower objective values than all infeasible solutions ($M = \sum c_j$). A drawback of using such an objective function is that infeasible solutions cannot be compared to each other because the objective function does not reflect the degree of infeasibility. Objective functions that penalize infeasible solutions while reflecting the degree of infeasibility are proposed in [16, 18]. In [18], the penalty attributed to an infeasible solution is proportional to the number of elements that are not covered in the solution. In [16], the penalty is proportional to the minimum cost it would take to cover all of the uncovered elements. In all discussed penalty approaches, the penalties are high enough to ensure that all infeasible solutions have higher objective values than all feasible ones. An immediate disadvantage of using such high penalties is that feasible solutions will always be preferred over infeasible ones. As a result, infeasible solutions will have low chances of surviving in the population, and the infeasible region of the search space will not be effectively explored.

*2.3. Local Search.* The feasibility constraint makes designing an effective local search metaheuristic for the SCP a difficult task. For this reason, few-local-search only heuristics have been developed for the SCP [8, 19]. Instead, most of the local search algorithms have combined local search with other techniques such as *Lagrangian relaxation*, *subgradient optimization*, *group theory,* and *linear programming* [1, 20–24]. In [25], after noting the difficulty of defining a good neighbourhood to solve the unicost set covering problem with local search, the authors proposed that the problem could be transformed to an equivalent satisfiability problem (SAT) that can be solved more adequately with local search.

Most local search algorithms for the SCP use a simple 1-flip neighbourhood structure defined by moves that only add (remove) one set at a time to (from) the solution. When a local optimum is reached, which is usually a feasible solution, it is difficult to decide in which direction to continue the search. Two cases arise.

(i) If the search space is restricted to the feasible region, only redundant sets are allowed to be removed. If no redundant sets exist in the solution, at least one redundant set must be added before a remove move is allowed to be performed. As a result, the infeasible region of the search space will not be explored and the search will tend to fall into local optimums and cycles. A more complex neighbourhood called 3-flip is used in [8] to make the search in the feasible region more effective. The 3-flip neighbourhood of a given solution consists of all of the solutions that can be obtained by adding (removing) at most 3 sets to (from) the solution. Eventhough the proposed heuristic is more effective than a simple 1-flip heuristic, it is not sufficient to avoid local optimums and cycles and it is significantly slower than the 1-flip heuristics.

(ii) If the search space is not restricted to the feasible region, the cost minimization objective drives the search toward the infeasible region, by removing sets from the current configuration (to minimize the cost), and it is unclear when to restore feasibility. In such situations, penalty approaches are usually used to penalize infeasible solutions.

If the penalty weights are too high, neighbors in the feasible region will be preferred over neighbors in the infeasible region, making the infeasible region unreachable. Lower or dynamic penalty weights are usually used to make the search more effective by allowing it to reach infeasible regions.

If the penalty weights are too low, the final solution found is not guaranteed to be feasible. A tabu search heuristic that uses such low penalties is proposed in [19] for the unicost set covering problem. A simple 1-flip neighbourhood structure is used. The objective is to minimize $(C + E)$ where $C$ is the number of sets used in the solution and $E$ is the number of uncovered elements. If a set covers only one uncovered element, adding (removing) it to (from) the solution will not have any effect on the objective function. As a result, this set might be left out of the solution, making it infeasible. To overcome the fact that this objective function does not guarantee feasibility, the neighbourhood is restricted such that if a set is removed during one iteration, one or more sets must be added in the next iteration to restore feasibility. Eventhough such a low penalty approach allows the search to reach the infeasible region, additional neighbourhood restrictions are used to restore feasibility, and the infeasible region is only scratched.

Dynamic penalty approaches, in which the penalty weights are repeatedly adjusted, are used to balance the search between the feasible and infeasible regions without using a repair operator or neighbourhood restrictions [1, 20–22, 26]. The most frequent dynamic penalty approaches that have been used in the literature are based on Lagrangian relaxation [27] and subgradient optimization [28]. Dynamic penalty approaches can be very effective but are difficult to be designed and implemented.

## 3. Proposed Formulation

In this work, we propose a new formulation of the SCP with a maximization objective. The aim of the proposed formulation is to express the real objective of the SCP in the objective function which is to *cover all* elements at a *minimal cost*. We view covering an element as collecting a gain at a given cost. In this perspective, we attribute a gain to each element. Because all elements must be covered, the gain attributed to each element must be higher than the cost of at least one of the sets that covers the element; otherwise, there is no benefit of covering that element. Let $c_{\min}(e_i)$ be the cost of the cheapest set among the sets that cover the element $e_i$. A gain $g_i = c_{\min}(e_i) + \epsilon$ is attributed to each element $e_i$ where $\epsilon$ is a small positive constant.

(i) Let $A^{m\times n}$ be a zero-one matrix where $a_{ij} = 1$ if element $e_i$ is covered by set $j$ and $a_{ij} = 0$ otherwise.

(ii) Let $X = \{x_1, x_2, \ldots, x_n\}$ where $x_j = 1$ if set $s_j$ (with cost $c_j > 0$) is part of the solution and $x_j = 0$ otherwise.

(iii) Let $Y = \{y_1, y_2, \ldots, y_m\}$ where $y_i = 1$ if element $e_i$ (with gain $g_i > 0$) is covered in the solution and $y_i = 0$ otherwise.

Maximize

$$\sum_{i=1}^{m} g_i y_i - \sum_{j=1}^{n} c_j x_j \tag{4}$$

subject to

$$y_i \leq \sum_{j=1}^{n} a_{ij} x_j, \quad i = 1, \ldots, m \tag{5}$$

$$x_j, y_i \in \{0, 1\}. \tag{6}$$

Constraint (5) is a relaxation of constraint (2) because it does not impose coverage of all the elements; its only purpose is to keep track of which elements of $E$ are part of the cover. Constraint (6) is the integrity constraint in mathematical programming. Constraints (5) and (6) do not need to be addressed as constraints in heuristic approaches but are presented for completeness of the mathematical programming formulation.

*Claim 1.* The optimal solution of the proposed formulation is a feasible solution (covers all elements).

*Proof.* Suppose that the optimal solution does not cover all of the elements and has an objective value $P$. Let $e_i$ be an uncovered element. By the definition of the gain $g_i$, we know that there is at least a set $s_j$ that covers element $e_i$ and has a cost $c_j = c_{\min}(e_i) = g_i - \epsilon$. If the set $s_j$ is added to the cover, the new objective value is $P' = P + (g_i - c_j) = P + (g_i - (g_i + \epsilon)) = P + \epsilon > P$. Thus, $P$ is not optimal. By contradiction, we conclude that the optimal solution covers all of the elements. □

*Claim 2.* The optimal solution of the proposed formulation covers all elements at a minimal cost.

*Proof.* We proved in Claim 1 that the optimal solution covers all of the elements. Hence, the first term of the objective function (4) is a constant in the optimal solution ($\sum_{i=1}^{m} g_i y_i = K$). The objective function becomes

$$\text{Maximize}\left(K - \sum_{j=1}^{n} c_j x_j\right) \Longrightarrow \text{Maximize}\left(-\sum_{j=1}^{n} c_j x_j\right) \Longrightarrow \text{Minimize} \sum_{j=1}^{n} c_j x_j. \tag{7}$$

Thus, the optimal solution of the proposed formulation is the cheapest feasible solution, which is the objective of the SCP. □

From heuristic algorithms perspective, we replaced a constrained optimization problem with an unconstrained optimization problem that has the same optimal solutions. Unconstrained optimization problems are known to be much easier to solve with heuristic algorithms than constrained optimization problems.

*3.1. Comparison to Penalty Approaches.* Eventhough the proposed formulation is a full mathematical programming formulation for the SCP, it is similar to the existing penalty approaches but with some important differences. The objective function presented in (4) can be rewritten as

$$\text{Minimize} \quad \sum_{j=1}^{n} c_j x_j + \sum_{i=1}^{m} g_i \overline{y_i}, \tag{8}$$

where $\overline{y_i} = 1$ if element $e_i$ is uncovered and $\overline{y_i} = 0$ otherwise. The value of the gain $g_i$ can be seen as the penalty associated with not covering the element $e_i$.

The proposed approach is different from high-penalty approaches because some infeasible solutions might have a better objective value than some feasible ones. For instance, let $U = \{s_1, s_2, s_3\}$, $s_1 = \{e_1, e_2, e_3\}$, $s_2 = \{e_1, e_2\}$, and $s_3 = \{e_3\}$. The costs of the sets are $c_1 = 10$, $c_2 = 2$, and $c_3 = 1$. The cheapest set that covers the element $e_1$ is $s_2$ with a cost $c_2 = 2$. Thus, by definition of the gain, $g_1$ is equal to $c_2 + \epsilon = 2 + \epsilon$. Similarly, we find that $g_2 = 2 + \epsilon$ and $g_3 = 1 + \epsilon$. Let $X_1 = \{s_1\}$ be a feasible solution, and let $X_2 = \{s_2\}$ be an infeasible one. Using the objective function (8), the objective value of $X_1$ is 10 and the objective value of $X_2$ is $(c_2 + e_3) = (3 + \epsilon)$. Thus, the infeasible solution $X_2$ has a lower (better) objective value than the feasible solution $X_1$, which does not occur with high-penalty approaches.

The proposed approach is different from low-penalty approaches because the penalties are high enough to drive the search toward the feasible region. We showed that the optimal solutions with respect to the new formulation are guaranteed to be feasible. The proof of feasibility of the optimal solution also shows that any infeasible solution can be transformed to a feasible one with a better objective value. For instance, in the previous example, the infeasible solution $X_2$ can be transformed to a feasible solution $X_3 = \{s_2, s_3\}$ (by adding the set $s_3$ to $X_2$) with an objective value of 3, which is lower (better) than the objective value of $X_2$ $(3 + \epsilon)$.

The proposed penalty approach is different from dynamic penalty approaches because the penalty weights are static and no adjustment is needed.

When high-penalty approaches are used, the search process of a heuristic algorithm is disturbed by the high penalties and driven immediately to the feasible region. On the other hand, low penalties do not disturb the search but cannot ensure feasibility. The aim of our approach is to choose the lowest possible penalties that avoid disturbing the search process while ensuring feasibility. Ensuring feasibility means that any infeasible solution can be transformed to a feasible one with a better objective value.

*3.2. Benefits of the New Formulation with respect to Metaheuristics.* The new formulation eliminates all issues related to solution infeasibility and set redundancy that were discussed in the literature review (Section 2). Because the objective function naturally penalizes redundant sets, the use of a redundancy removal operator is not needed. The objective function also penalizes infeasible solutions. As a result, the use of a repair or penalty approaches in evolutionary algorithms and the use of neighbourhood restrictions in local search algorithms are not needed. Finally, because no constraints are involved and the only driver of the search is the objective function proposed with the new formulation, designing a good neighbourhood and local search algorithm is quite simple. Such a simple neighbourhood is presented in Section 4.

## 4. Proposed Descent Heuristic (DH)

In this section, we present a simple descent heuristic that is based on the new formulation and that uses a 1-flip neighbourhood structure. We also show that all local optimums with respect to the new formulation and the 1-flip neighbourhood are feasible and free of redundant sets.

The proposed descent heuristic (DH) is an adaptation of the classical greedy heuristic that has been used in the literature for the SCP [29]. In this greedy heuristic, the set $s_j$ with the minimum ratio $\eta_j = c_j/\text{card}_j(X)$ is added to the solution in each iteration. The term $\text{card}_j(X)$ is the number

```
sol ← empty solution;
loop
    find the set $s_j$ with the maximum ratio $R_j$;
    if ($R_j > 0$) then
        flip bit $x_j$;
    else
        stop;
    end if
end loop
```

Algorithm 1: DH().

of elements that are covered by $s_j$ and are not covered by the current configuration $X$. Once all of the elements are covered, redundant sets are removed in decreasing order of cost. In DH, the term $\text{card}_j(X)$ of the classical greedy heuristic is replaced with $\delta_j$, where $\delta_j$ is the variation in the objective function associated with adding (removing) the set $s_j$.

DH starts from a given configuration and performs a sequence of moves on it until the solution is locally optimal. It uses a simple 1-flip neighbourhood structure with two types of moves: add and remove moves. add($j$) adds the set $s_j$ to the configuration (flips $x_j$ from 0 to 1), while remove($j$) removes the set $s_j$ from the configuration. In each iteration, the set $s_j$ with the maximum ratio $R_j = \delta_j/c_j$ is added (removed) to (from) the solution. The algorithm stops when the current configuration is better than all of its neighbors ($R_j \leq 0$ for all $j$). The outline of DH is presented in Algorithm 1.

*4.1. Redundancy Removal.* In contrast to the classical greedy heuristic, DH automatically removes the redundant sets from the solution. Let $X$ be a configuration where the set $s_j$ is redundant. The ratio $R_j$ associated with removing $s_j$ from $X$ is equal to $(c_j - 0)/c_j = 1$. Because $R_j > 0$, the move remove($j$) will be performed and the redundant sets will be removed. As a result, any solution that is improved with DH is necessarily free of redundant sets. The redundant sets are removed at any time during the progress of DH and not only at the end.

*4.2. Feasibility.* Consider $X$ to be a configuration where $e_i$ is not covered. Let $s^i_{\min}$ be the cheapest set that covers $e_i$, and let $c^i_{\min}$ be its associated cost. The gain $g_i$ associated with $e_i$ is equal to $c^i_{\min} + \epsilon$. If $e_i$ is the only uncovered element covered by $s^i_{\min}$ (worst case scenario), the ratio $R^i_{\min}$ associated with adding the set $s^i_{\min}$ to $X$ is equal to $(c^i_{\min} + \epsilon - c^i_{\min})/c^i_{\min} = \epsilon/c^i_{\min}$. Because $R^i_{\min} > 0$ (for all $\epsilon > 0$), the move add($s^i_{\min}$) will be performed and the solution will be feasible. As a result, any solution that is improved with DH is feasible.

*4.3. Discussion.* We showed that all of the solutions that are found with DH are feasible and free of redundant sets. With respect to the new formulation and the 1-flip neighbourhood structure, these solutions are local optimums. This is also true for all solutions obtained with any descent heuristic that is based on the new formulation and that uses the same neighbourhood structure. As a result, all local optimums with respect to the new formulation and the 1-flip neighbourhood structure are feasible and free of redundant sets.

## 5. Experimental Analysis

In this section, we present computational experiments with the proposed descent heuristic that is based on the new formulation. Although we showed in the previous sections that the new formulation provides many advantages over the classical formulation, the final performance of any metaheuristic algorithm depends on the implementation, the tuning of the parameters, and the sophistication of the approach. We do not assume that any metaheuristic approach that is based on the new formulation will outperform all metaheuristic approaches that are based on the classical formulation. In addition, experimenting with all classes of metaheuristics will not prove (or disprove) the superiority of the proposed formulation. Instead, we compare our descent heuristic to the original greedy heuristic that is based on the classical formulation. The aim is to compare the two formulations using similar algorithms. Since greedy heuristics are used for intensification in most of the metaheuristic approaches for the SCP, evaluating the effectiveness of a new descent heuristic that can replace these greedy heuristics provides a good indication of how suitable is the new formulation to metaheuristic approaches.

We compare DH to the classical greedy heuristic (GH) [29] on three classes of the known set covering problems.

(i) *OR-Library benchmarks*: this class includes 65 small and medium size randomly generated problems that were frequently used in the literature. Most metaheuristic approaches for the SCP have been tested on these problems. They are available in OR-Library [30] and are described in Table 1.

(ii) *Airline and bus scheduling problems*: this class includes fourteen real-world airline scheduling problems (AA instances) and two bus driver scheduling problems (bus instances). These problems were obtained from [31] and are described in Table 2.

(iii) *Railway scheduling problems*: this class includes seven large-scale railway crew scheduling problems from Italian railways and are available in OR-Library [30]. These problems are described in Table 3.

Most metaheuristic approaches for the SCP have been exclusively tested on OR-Library benchmarks. Because these benchmarks are relatively small, we experimented with larger problems that have been less frequently used in the literature.

In all presented tables, the name of each instance is given in the first column, the size of each instance is given in the second column (number of elements × number of sets), and the density of each instance is given in the third column. The density is the percentage of ones in the $A^{m\times n}$ matrix described in Section 1). The optimal or best-known solution of each instance is given in the fourth column. The solutions obtained with each heuristic are presented in columns 5 and 6. The last two columns contain the number of iterations performed by

Table 1: OR-Library benchmarks.

| | Characteristics | | Cost | | | Number of moves | |
|---|---|---|---|---|---|---|---|
| Instance | Size | Density | Best known | DH | GH | DH | GH |
| 4.1 | 200 × 1000 | 2% | 429 | 433 | 434 | 77 | 93 |
| 4.2 | 200 × 1000 | 2% | 512 | 523 | 552 | 75 | 94 |
| 4.3 | 200 × 1000 | 2% | 516 | 531 | 546 | 79 | 96 |
| 4.4 | 200 × 1000 | 2% | 494 | 503 | 507 | 70 | 93 |
| 4.5 | 200 × 1000 | 2% | 512 | 515 | 518 | 72 | 95 |
| 4.6 | 200 × 1000 | 2% | 560 | 575 | 597 | 78 | 83 |
| 4.7 | 200 × 1000 | 2% | 430 | 444 | 449 | 74 | 77 |
| 4.8 | 200 × 1000 | 2% | 492 | 493 | 525 | 70 | 77 |
| 4.9 | 200 × 1000 | 2% | 641 | 672 | 672 | 82 | 99 |
| 4.10 | 200 × 1000 | 2% | 514 | 519 | 528 | 71 | 86 |
| 5.1 | 200 × 2000 | 2% | 253 | 265 | 273 | 76 | 88 |
| 5.2 | 200 × 2000 | 2% | 302 | 314 | 335 | 71 | 82 |
| 5.3 | 200 × 2000 | 2% | 226 | 230 | 230 | 66 | 82 |
| 5.4 | 200 × 2000 | 2% | 242 | 246 | 254 | 69 | 86 |
| 5.5 | 200 × 2000 | 2% | 211 | 214 | 215 | 73 | 87 |
| 5.6 | 200 × 2000 | 2% | 213 | 216 | 227 | 69 | 85 |
| 5.7 | 200 × 2000 | 2% | 293 | 297 | 305 | 76 | 84 |
| 5.8 | 200 × 2000 | 2% | 288 | 297 | 304 | 77 | 85 |
| 5.9 | 200 × 2000 | 2% | 279 | 281 | 290 | 68 | 84 |
| 5.10 | 200 × 2000 | 2% | 265 | 271 | 274 | 74 | 81 |
| 6.1 | 200 × 1000 | 5% | 138 | 149 | 143 | 39 | 56 |
| 6.2 | 200 × 1000 | 5% | 146 | 156 | 154 | 44 | 53 |
| 6.3 | 200 × 1000 | 5% | 145 | 149 | 157 | 43 | 46 |
| 6.4 | 200 × 1000 | 5% | 131 | 134 | 140 | 46 | 51 |
| 6.5 | 200 × 1000 | 5% | 161 | 180 | 182 | 47 | 50 |
| A.1 | 300 × 3000 | 2% | 253 | 258 | 269 | 82 | 97 |
| A.2 | 300 × 3000 | 2% | 252 | 262 | 268 | 78 | 93 |
| A.3 | 300 × 3000 | 2% | 232 | 243 | 248 | 80 | 105 |
| A.4 | 300 × 3000 | 2% | 234 | 240 | 243 | 84 | 107 |
| A.5 | 300 × 3000 | 2% | 236 | 240 | 246 | 79 | 107 |
| B.1 | 300 × 3000 | 5% | 69 | 72 | 71 | 41 | 45 |
| B.2 | 300 × 3000 | 5% | 76 | 79 | 78 | 44 | 50 |
| B.3 | 300 × 3000 | 5% | 80 | 84 | 84 | 47 | 46 |
| B.4 | 300 × 3000 | 5% | 79 | 84 | 88 | 44 | 50 |
| B.5 | 300 × 3000 | 5% | 72 | 72 | 75 | 46 | 48 |
| C.1 | 400 × 4000 | 2% | 227 | 237 | 252 | 102 | 110 |
| C.2 | 400 × 4000 | 2% | 219 | 230 | 225 | 93 | 128 |
| C.3 | 400 × 4000 | 2% | 243 | 249 | 258 | 89 | 102 |
| C.4 | 400 × 4000 | 2% | 219 | 229 | 239 | 94 | 115 |
| C.5 | 400 × 4000 | 2% | 215 | 222 | 222 | 93 | 106 |
| D.1 | 400 × 4000 | 5% | 60 | 64 | 66 | 49 | 54 |
| D.2 | 400 × 4000 | 5% | 66 | 68 | 69 | 52 | 50 |
| D.3 | 400 × 4000 | 5% | 72 | 77 | 80 | 54 | 59 |
| D.4 | 400 × 4000 | 5% | 62 | 62 | 66 | 52 | 54 |
| D.5 | 400 × 4000 | 5% | 61 | 65 | 67 | 49 | 61 |

Table 1: Continued.

| | Characteristics | | | Cost | | Number of moves | |
|---|---|---|---|---|---|---|---|
| Instance | Size | Density | Best known | DH | GH | DH | GH |
| E.1 | 500 × 5000 | 10% | 29 | 30 | 30 | 30 | 35 |
| E.2 | 500 × 5000 | 10% | 30 | 33 | 35 | 31 | 37 |
| E.3 | 500 × 5000 | 10% | 27 | 29 | 31 | 29 | 31 |
| E.4 | 500 × 5000 | 10% | 28 | 32 | 31 | 32 | 33 |
| E.5 | 500 × 5000 | 10% | 28 | 30 | 30 | 30 | 32 |
| F.1 | 500 × 5000 | 20% | 14 | 16 | 17 | 16 | 17 |
| F.2 | 500 × 5000 | 20% | 15 | 16 | 16 | 15 | 16 |
| F.3 | 500 × 5000 | 20% | 14 | 17 | 15 | 17 | 17 |
| F.4 | 500 × 5000 | 20% | 14 | 17 | 15 | 17 | 14 |
| F.5 | 500 × 5000 | 20% | 13 | 15 | 15 | 17 | 15 |
| G.1 | 1000 × 10000 | 2% | 176 | 186 | 191 | 132 | 146 |
| G.2 | 1000 × 10000 | 2% | 154 | 166 | 176 | 115 | 139 |
| G.3 | 1000 × 10000 | 2% | 166 | 178 | 182 | 126 | 147 |
| G.4 | 1000 × 10000 | 2% | 168 | 178 | 179 | 128 | 138 |
| G.5 | 1000 × 10000 | 2% | 168 | 179 | 182 | 127 | 131 |
| H.1 | 1000 × 10000 | 5% | 63 | 69 | 69 | 68 | 65 |
| H.2 | 1000 × 10000 | 5% | 63 | 70 | 72 | 62 | 67 |
| H.3 | 1000 × 10000 | 5% | 59 | 63 | 66 | 62 | 62 |
| H.4 | 1000 × 10000 | 5% | 58 | 65 | 64 | 65 | 61 |
| H.5 | 1000 × 10000 | 5% | 55 | 60 | 61 | 61 | 60 |

Table 2: Airline and bus driver crew scheduling problems.

| | Characteristics | | | Cost | | Number of moves | |
|---|---|---|---|---|---|---|---|
| Instance | Size | Density | Best known | DH | GH | DH | GH |
| AA03 | 106 × 8661 | 4.05% | 33155 | 34637 | 35642 | 48 | 61 |
| AA04 | 106 × 8002 | 4.05% | 34573 | 36153 | 36749 | 45 | 62 |
| AA05 | 105 × 7435 | 4.05% | 31623 | 32249 | 32995 | 45 | 65 |
| AA06 | 105 × 6951 | 4.11% | 37464 | 38043 | 39422 | 43 | 70 |
| AA11 | 271 × 4413 | 2.53% | 35478 | 36965 | 39054 | 76 | 90 |
| AA12 | 272 × 4208 | 2.52% | 30815 | 33663 | 34044 | 77 | 85 |
| AA13 | 265 × 4025 | 2.60% | 33211 | 36337 | 37345 | 77 | 91 |
| AA14 | 266 × 3868 | 2.50% | 33219 | 36048 | 36530 | 77 | 95 |
| AA15 | 267 × 3701 | 2.58% | 34409 | 36269 | 37996 | 73 | 94 |
| AA16 | 265 × 3558 | 2.63% | 32752 | 36185 | 37160 | 79 | 85 |
| AA17 | 264 × 3425 | 2.61% | 31612 | 34326 | 36484 | 69 | 91 |
| AA18 | 271 × 3314 | 2.55% | 36782 | 39594 | 40603 | 84 | 101 |
| AA19 | 263 × 3202 | 2.63% | 32317 | 34749 | 36093 | 71 | 92 |
| AA20 | 269 × 3095 | 2.58% | 34912 | 37047 | 37744 | 82 | 86 |
| BUS1 | 454 × 2241 | 1.89% | 27947 | 28871 | 29673 | 88 | 100 |
| BUS2 | 681 × 9524 | 0.51% | 67760 | 69685 | 70606 | 282 | 280 |

each heuristic for each instance. The percentage deviations from the best-known solutions are presented in Figures 1, 2, 3 and 4.

In both DH and GH, each iteration involves finding the best set to be added (removed) to (from) the solution and updating the underlying data structure after a move is performed. Thus, the algorithmic complexity of each iteration is similar in both heuristics. In practice, the computation times are highly dependent on the implementation and the characteristics of the problem solved (size and density). For instance, finding the best move to be performed in each iteration can be implemented using a loop that iterates over all sets

Table 3: Railway crew scheduling problems.

| | Characteristics | | | Cost | | Number of moves | |
|---|---|---|---|---|---|---|---|
| Instance | Size | Density | Best known | DH | GH | DH | GH |
| RAIL507 | 507 × 63009 | 1.2% | 174 | 205 | 212 | 150 | 169 |
| RAIL516 | 516 × 47311 | 1.3% | 182 | 202 | 202 | 181 | 186 |
| RAIL582 | 582 × 55515 | 1.2% | 211 | 243 | 251 | 191 | 212 |
| RAIL2586 | 2586 × 920683 | 0.4% | 948 | 1102 | 1185 | 770 | 917 |
| RAIL2536 | 2536 × 1081841 | 0.4% | 691 | 828 | 891 | 581 | 660 |
| RAIL4284 | 4284 × 1092610 | 0.2% | 1065 | 1303 | 1385 | 997 | 1091 |
| RAIL4872 | 4872 × 968672 | 0.2% | 1534 | 1802 | 1900 | 1339 | 1521 |

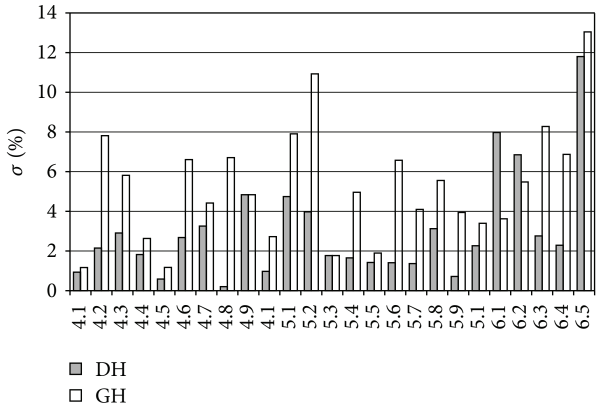


Figure 1: Percentage deviation from the best-known solution: OR-Library benchmarks 4.1 to 6.5.

or using a priority-queue-based data structure. Preliminary testing showed that choosing one way or another greatly affects the speed comparison of the discussed heuristics. To avoid an implementation-dependent comparison, and because these aspects of the implementation are out of the scope of this work, we recorded the number of iterations instead.

Both heuristics are deterministic, and only one run is required. The value of $\epsilon$ used in all DH runs is equal to $1 \times e^{-5}$. Smaller values of epsilon have caused numerical problems for some instances.

Our descent heuristic performed better than GH by finding better solutions for most of the test problems. For OR-Library benchmarks, DH found better solutions than GH for 47 instances, equal solutions for 10 instances, and worse solutions for 9 instances. For the airline, bus, and railway scheduling problems, DH found better solutions than GH for all problems except one (equal solutions for RAIL516). The percentage deviations presented in Figures 1, 2, 3 and 4 and the average percentage deviation presented in Table 4 show that the solutions found by DH are also significantly better in quality than those found by GH (up to 7.41% better for OR-Library, up to 6.83% better for airline and bus problems, and up to 9.12% better for railway problems).

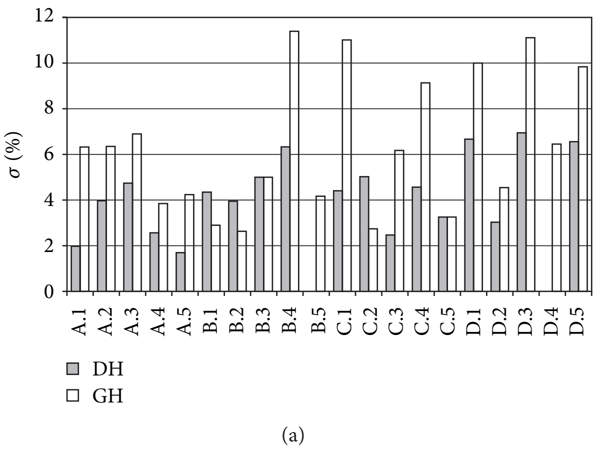
DH also performed fewer iterations than GH for most of the test problems. For OR-Library benchmarks, DH performed fewer iterations than GH for 56 instances, equal number of iterations for seven instances, and more iterations for only two instances. For the airline, bus, and railway scheduling problems, DH performed fewer iterations than GH for all problems except one (more iterations for BUS2). The average number of iterations performed by DH and GH is presented in Table 4. The average number of iterations shows that the number of iterations performed by DH is significantly smaller than the number of iterations performed by GH. Thus, DH is theoretically faster than GH.

As a result, the proposed descent heuristic that is based on the new formulation performs better than the corresponding greedy heuristic that is based on the classical formulation by finding better results for most of the test problems using fewer iterations, which can lead to shorter computation times.

## 6. Conclusions and Future Work

In this paper, we identified two issues that arise when solving the SCP with metaheuristic approaches: solution infeasibility and set redundancy. We highlighted the difficulties of addressing these issues when solving the SCP with the different classes of metaheuristics and proposed a new formulation that overcomes these difficulties. We showed that this formulation is, in fact, a new penalty approach that uses static penalty weights that are low enough to avoid disturbing the search but high enough to ensure the feasibility of the final solution. We also showed that all local optimums with respect to the new formulation and the 1-flip neighbourhood structure are feasible and free of redundant sets. As a result, building metaheuristic approaches for the SCP using the new formulation is straightforward.

To provide a first computational experience using the new formulation, we adapted a known greedy heuristic for the SCP to the new formulation and compared the adapted version to the original version using 88 set covering problems. The adapted version that is based on the new formulation found better solutions than the original version that is based on the classical formulation for 69 tests problems, equal solutions for ten problems, and worse solutions for nine problems. In addition, the adapted version performed fewer iterations than the original version for 78 test problems, equal number of iterations for two problems, and more iterations for eight problems. Thus the adapted version finds better



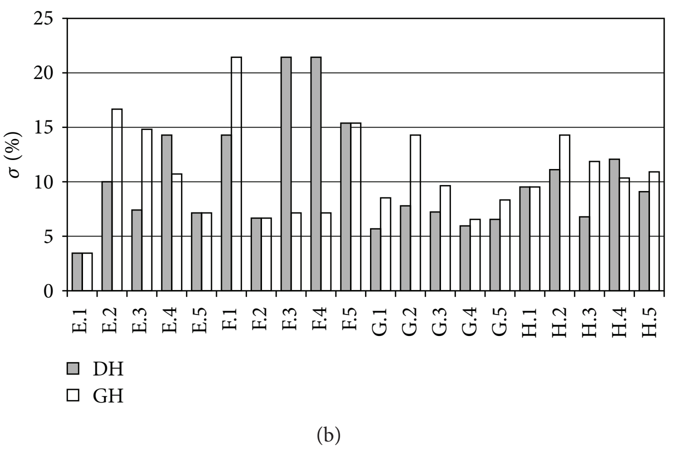


FIGURE 2: Percentage deviation from the best-known solution: OR-Library benchmarks A.1 to H.5.

TABLE 4: Average number of iterations and percentage deviations.

| Problems | Average number of iterations | | Average percentage deviation | |
|---|---|---|---|---|
| | DH | GH | DH | GH |
| OR-Library benchmarks | 64.89 | 74.51 | 5.46 | 7.31 |
| Airline and bus problems | 82.25 | 96.75 | 5.99 | 9.14 |
| Railway problems | 601.29 | 679.43 | 17.12 | 22.81 |

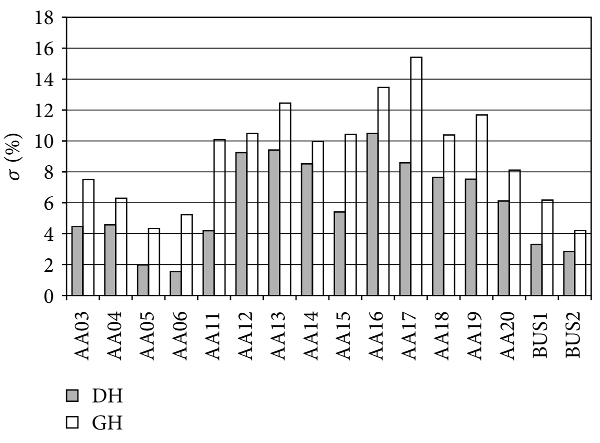


FIGURE 3: Percentage deviation from the best-known solution: airline and bus scheduling problems.

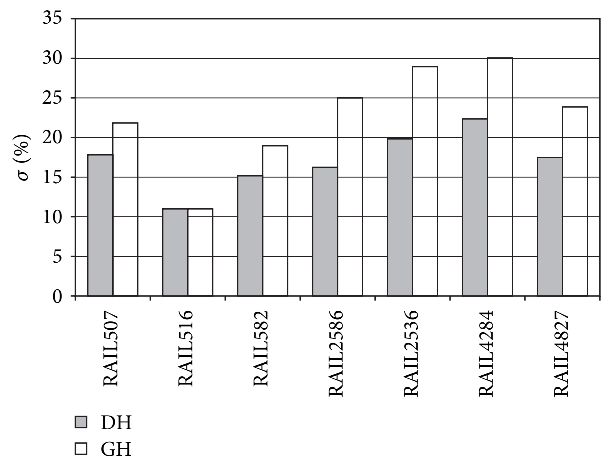


FIGURE 4: Percentage deviation from the best-known solution: railway scheduling problems.

solutions than the original version in potentially shorter computation times. Moreover, the adapted version was easier to implement because we did not need to handle feasibility and set redundancy.

Most current metaheuristic approaches for the SCP incorporate a descent or greedy heuristic that is responsible for the intensification part of the search. Thus, having a more effective descent heuristic can lead to better metaheuristic approaches.

## References

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

[2] G. Lan, G. W. DePuy, and G. E. Whitehouse, "An effective and simple heuristic for the set covering problem," *European Journal of Operational Research*, vol. 176, no. 3, pp. 1387–1403, 2007.

[3] E. Balas, *Class of Location, Distribution and Scheduling Problems: Modeling and Solution Methods*, Carnegie Mellon University. Design Research Center, New York, NY, USA, 1983.

[4] S. Ceria, "Set covering problem," in *Annotated Bibliographies in Combinatorial Optimization*, 1997.

[5] M. Garey and D. Johnson, *Computers and Intractability. A Guide To the Theory of NP-Completeness*, W. H. Freeman, Oxford, UK, 1979.

[6] L. Lessing, I. Dumitrescu, and T. Stutzle, "A comparison between acoalgorithms for the set covering problem," *Ant Colony Optimization and Swarm Intelligence*, pp. 105–122, 2004.

[7] M. Rahoual, R. Hadji, and V. Bachelet, "Parallel ant system for the set covering problem," in *Ant Algorithmspages*, pp. 249–297, 2002.

[8] Z. Ren, Z. Feng, L. Ke, and H. Chang, "A fast and efficient ant colony optimization approach for the set covering problem," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '08), (IEEE World Congress on Computational Intelligence)*, pp. 1839–1844, IEEE, June 2008.

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

[10] B. Crawford and C. Castro, "Integrating lookahead and post processing procedures with aco for solving set partitioning and covering problems," in *proceedings of the 8th International Conference on Artificial Intelligence and Soft Computing (ICAISC '06)*, pp. 1082–1090, 2006.

[11] G. DePuy, G. Whitehouse, and R. Moraga, "Using the meta-raps approach to solve combinatorial problems," in *Proceedings of the 2002 Industrial Engineering Research Conference*, vol. 19, p. 21, Citeseer, May 2002.

[12] T. A. Feo and M. G. C. Resende, "Greedy randomized adaptive search procedures," *Journal of Global Optimization*, vol. 6, no. 2, pp. 109–133, 1995.

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

[14] M. Solar, V. Parada, and R. Urrutia, "A parallel genetic algorithm to solve the set-covering problem," *Computers and Operations Research*, vol. 29, no. 9, pp. 1221–1235, 2002.

[15] D. Orvosh and L. Davis, "Using a genetic algorithm to optimize problems with feasibility constraints," in *Proceedings of the 1st IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, pp. 548–553, IEEE, June 1994.

[16] T. Bäck, M. Schütz, and S. Khuri, "A comparative study of a penalty function, a repair heuristic, and stochastic operators with the set-covering problem," in *Artificial Evolution*, pp. 320–332, Springer, New York, NY, USA, 1996.

[17] K. S. Al-Sultan, M. F. Hussain, and J. S. Nizami, "A genetic algorithm for the set covering problem," *Journal of the Operational Research Society*, vol. 47, no. 5, pp. 702–709, 1996.

[18] R. L. Wang and K. Okazaki, "An improved genetic algorithm with conditional genetic operators and its application to set-covering problem," *Soft Computing*, vol. 11, no. 7, pp. 687–694, 2007.

[19] N. Musliu, "Local search algorithm for unicost set covering problem," *Advances in Applied Artificial Intelligence*, vol. 4031, pp. 302–311, 2006.

[20] M. Yagiura, M. Kishida, and T. Ibaraki, "A 3-flip neighborhood local search for the set covering problem," *European Journal of Operational Research*, vol. 172, no. 2, pp. 472–499, 2006.

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

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

[23] G. Kinney and J. Barnes, "A group theoretic tabu search algorithm for set covering problems," Working Paper, 2004, http://www.researchgate.net/publication/228734774_A_group_theoretic_tabu_search_algorithm_for_set_covering_problems.

[24] G. W. Kinney, J. W. Barnes, and B. W. Colletti, "A reactive tabu search algorithm with variable clustering for the Unicost set covering problem," *International Journal of Operational Research*, vol. 2, no. 2, pp. 156–172, 2007.

[25] J. Bautista and J. Pereira, "A GRASP algorithm to solve the unicost set covering problem," *Computers and Operations Research*, vol. 34, no. 10, pp. 3162–3173, 2007.

[26] M. Caserta, "Tabu search-based metaheuristic algorithm for large-scale set covering problems," *Metaheuristics*, vol. 39, pp. 43–63, 2007.

[27] M. L. Fisher, "The lagrangian relaxation method for solving integer programming problems," *Management Science*, vol. 27, no. 1, pp. 1–18, 1981.

[28] M. Held, P. Wolfe, and H. P. Crowder, "Validation of subgradient optimization," *Math Program*, vol. 6, no. 1, pp. 62–88, 1974.

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

[30] J. Beasley, "OR-library: distributing test problems by electronic mail," *Journal of the Operational Research Society*, vol. 41, no. 11, pp. 1069–1072, 1990.

[31] B. Yelbay, S. Birbil, and K. Bulbul, "The set covering problem revisited: an empirical study of the value of dual information," *Optimization Online*, 2012.