Part II

Tabu Search

Chapter 3

TABU SEARCH-BASED METAHEURISTIC ALGORITHM FOR LARGE-SCALE SET COVERING PROBLEMS

Marco Caserta
Instituto Tecnológico de Monterrey
Calle del Puente, 222
Col. Ejidos de Huipulco
Del. Tlalpan, México DF, 14380
México
marco.caserta@itesm.mx

Abstract This paper presents an algorithm for the Set Covering Problem whose centerpiece is a new primal-to-dual scheme aimed at linking any primal solution to the dual feasible vector that best reflects the quality of the primal solution. This new mechanism is used to intertwine a tabu search based primal intensive scheme with a Lagrangian based dual intensive scheme to design a dynamic primal-dual algorithm that progressively reduces the gap between upper and lower bound. The algorithm has been tested on benchmark problems from the literature: the gap between upper and lower bound in 6 instances of problems whose optimal solution is not known has been further reduced, 4 of them via improvements in the lower bound, and 4 by producing a solution that is better than the best solution provided by other procedures.

Keywords: Set Covering; Tabu Search; Metaheuristic; Primal-to-Dual.

1. Introduction

Our interest toward the set covering problem (SCP) is motivated by its use in the minimization of the number of patterns required to discriminate observations from a given population. Having an effective SCP algorithm, designed to tackle very large instances of SCP, is vital in order

to define a pattern generation and pattern minimization scheme with high classification power. This paper is devoted to the development of a tabu search-based metaheuristic algorithm for very large scale set covering instances. We design a dynamic primal-and-dual scheme especially suited for large instances of SCP that are typical in the classification of data from massive data sets.

The set covering problem is a $0-1$ integer problem with m rows in $M = \{1, \ldots, m\}$, and n columns in $N = \{1, \ldots, n\}$. A mathematical formulation for SCP is

$$(\text{SCP}) : \min \{z = \mathbf{cx} : \mathbf{Ax} \geq \mathbf{1}, \mathbf{x} \in \mathbb{B}^n\},$$

where $\mathbf{c} \in \mathbf{Z}_+^n$ and $\mathbf{A}$ is a matrix of 0's and 1's. In the following, we call *cover* a binary vector $\mathbf{x} \in \mathbb{B}^n$ that is a feasible solution of SCP, while a *prime cover* is a cover with no redundant columns. Also, let $J_i = \{j \in N : a_{ij} = 1\}$ be the index set of columns covering row i, and $I_j = \{i \in M : a_{ij} = 1\}$ the index set of rows covered by column j.

Many real-world applications can be formulated as SCP, including traditional delivery and routing problems, as well as scheduling and location problems. More recent applications of SCP are found in probe selection in hybridization experiments for DNA sequencing (*e.g.*, Borneman et al., 2001) and feature selection and pattern construction in LAD, the logical analysis of numerical data (*e.g.*, Boros et al., 1996).

SCP is $\mathcal{NP}$-complete (Garey and Johnson, 1979), hence exact solution procedures are doomed to fail in solving practical SCP problems. Furthermore, it is parameterized intractable, which is, $W[2]-$complete with respect to the parameter "solution size" (Downey and Fellows, 1999; Niedermeier, 2006; Dom et al., 2006). Supported by its applicability, the need for solution procedures that can efficiently handle large-scale instances of SCP has attracted a vast amount of interest in the optimization community in the past four decades and a great deal of effort has been directed, especially in the past two decades, toward the development of approximate algorithms for SCP. As a result, some algorithms are capable of solving SCPs with thousands of rows and millions of columns (*e.g.*, Ceria et al., 1998, Caprara et al., 1999).

To summarize, most approximate solution procedures for SCP are dual heuristic procedures based upon the solution of the Lagrangian relaxations of SCP via subgradient optimization (*e.g.* Caprara et al., 1999, Ceria et al., 1998, Balas and Carrera, 1996, Fisher and Kedia, 1990, Vaasko and Wilson, 1984, and Balas and Ho, 1980). As the dual procedures require greedy-type primal heuristics in order to build a primal cover, they can also be viewed as primal-and-dual algorithms with "dual-to-primal" mechanisms. In addition, more "advanced" dual

procedures for SCP typically feature some forms of probing and variable fixing schemes that dynamically update primal and dual information of SCP and aid in finding more effective solutions of SCP (*e.g.*, Caprara et al., 1999, Ceria et al., 1998, Balas and Carrera, 1996, and Beasley, 1990). Most algorithms designed to tackle very large scale instances work on a subset of variables, called "core problem" or "kernel problem". An interesting approach aimed at identifying the kernel problem has been proposed by Weihe, 1998, whose paper presents an effective data reduction technique that has been tested on very large railway problems. The objective is to select the minimum set of stations needed to cover a given set of trains. Real-world instances from the German and European Railroad network have been successfully solved by the author. The proposed scheme can be divided into two phases: first, the irreducible core problem is identified via dominance and equivalence relations; next, the core instance is solved via brute-force, when possible, or via heuristic scheme, when the dimensions of the core make an exhaustive search still too expensive. The author proposes an interesting approach, since he suggests that, when dealing with large scale instances, one should first work on the preprocessing scheme and, afterwards, design the routine that will work on the core instances, since it is only then that the characteristics of the core instances are known.

A major contribution of the paper is the development of a "primal-to-dual" (p2d) mechanism that, for any given primal solution, constructs a *feasible* dual vector that minimizes the gap between the upper bound of SCP given by *the* cover and the lower bound given by a feasible dual solution with respect to the sufficient optimality conditions presented in Theorem 3.1. The benefit of the primal-to-dual mechanism is twofold: (i) if the current cover is optimal to SCP, it verifies the optimality and the search process can be terminated; (ii) otherwise, it constructs a dual vector $\mathbf{u}$ that serves as a new starting vector for subgradient optimization. If different prime covers are provided, the primal-to-dual scheme constructs different $\mathbf{u}$'s, allowing subgradient optimization to explore different regions of the dual solution space. This, in turn, allows greedy-type dual-to-primal heuristics to construct different prime covers for SCP.

In this paper we integrate effective dual-to-primal mechanisms from the literature and a specialization of the novel primal-to-dual mechanism provided in Caserta and Ryoo, 2001 for SCP. We develop a primal-intensive, "dynamic" primal-and-dual metaheuristic for large-scale SCP. Computational experiments with the proposed metaheuristic on 94 benchmarks from Caprara et al., 1999, Balas and Carrera, 1996, and Wedelin, 1995 indicate that the proposed algorithm advances the state-of-the-art

in SCP quite substantially. Out of 94 benchmark problems, 21 of them have not been solved to optimality. For 6 of these 21 problems, our algorithm reduces the gap between best lower and upper bounds: new best solutions to 4 problems are found and the lower bounds of 4 problems have been improved. For the 73 benchmarks solved to optimality, the proposed algorithm finds the optimal solutions.

The proposed algorithm is made up of metaheuristic components that contribute to the efficiency and efficacy of the proposed algorithm. We first present an overview of the overall algorithm in Section 2. Subsequently, we present the metaheuristic components of the proposed algorithm in Sections 3-7. Computational experiments with 94 SCP benchmark problems are summarized in Section 8 and concluding remarks are provided in Section 9.

2. Overall Algorithm

In this section we present the overall algorithm, while the remaining sections will clarify each step of the proposed scheme. The basic idea of the proposed scheme is related to the development of a mechanism that connects the search in the primal space with the exploration of the dual space. This scheme, called (p2d), is thoroughly presented in Section 6 and is what makes the algorithm quite effective. The reason why (p2d) sensibly improves the performance of the algorithm is that it allows to create "synergies" between the primal phase, based upon the Tabu Search paradigm, and the dual phase, based upon the Lagrangian Relaxation technique coupled with subgradient optimization.

The pseudocode of Algorithm `PD_SCA()` along with Figure 3.1 provide a first overview of the general algorithm.

3. Tabu Search Metaheuristic

The tabu search metaheuristic of the proposed algorithm is the result of a specialization of the meta-strategy provided in Caserta and Ryoo, 2001 for SCP. For reasons of space, we provide details for those components that are problem-specific in nature for SCP. The proposed scheme is aimed at thoroughly exploring the feasible space along with a portion of the infeasible space. Furthermore, by introducing random and memory-based mechanisms, it aims at striking the balance between diversification and intensification.

The overall tabu search metaheuristic procedure is summarized in Procedure `Tabu_Search_Metaheuristic()`, while the remainder of the section is devoted to explaining the different ingredients of such procedure.

Algorithm `PD_SCA()`;
initialize $\mathbf{u}$ via (3.3)
call `Define_Core_Problem()` {`Section 7`}
$K = \left\lceil \frac{|N|}{|N_C|} \right\rceil$ {`total number of core problems examined`}
$k = 0$ {`cycles counter`}
while $k < K$ **do**
call `Tabu_Search_Metaheuristic()` {`Section 3`}
solve (p2d) {`Section 6`}
call `Define_Core_Problem()` {`Section 7`}
call `Lagrangian_Optimization()` {`Section 5`}
call `Fixing_to_Zero()` {`Section 7`}
call `Fixing_to_One()` {`Section 7`}
$k \leftarrow k + 1$ {`increase cycles counter`}
end while

Procedure `Tabu_Search_Metaheuristic()`;
Input: $\mathbf{x}^*$, UB, $\mathbf{x}^0$ (initial cover), $\mathcal{TL}$, (core) problem instance
Output: $\mathbf{x}^*$, UB, $\mathcal{TL}$
for $phase \in regular, intensification, diversification$ **do**
$k \leftarrow 0$ {`# excursions into allowed infeasible region`}
$\bullet = -$ {`start with the releasing phase`}
$t \leftarrow 0$ {`tabu search counter`}
while $k < 2$ **do**
call `Composite_Move_Assignment()`
if $\mathbf{x}^{t+1} \in X$ **then**
if $\mathbf{cx}^{t+1} < UB$ **then**
$\mathbf{x}^* \leftarrow \mathbf{x}^{t+1}$; $UB \leftarrow \mathbf{cx}^{t+1}$ {`update primal information`}
end if
if $(\mathbf{x}^t \in X)$ and $(\mathbf{x}^{t+1} \in \overline{X})$ **then**
$k \leftarrow k + 1$ {`end tabu iteration`}
end if
if $(\mathbf{x}^{t+1} \in X)$ and $(\mathbf{x}^t \in \overline{X})$ **then**
solve (p2d) {`see Section 6`}
partial pricing {`see Section 7`}
call `Lagrangian_Optimization()` {`see Section 5`}
end if
end if
end while
end for

Let $\mathbf{x}^k$ denote the current prime cover. Let us denote by B the index set of columns that take value 1 in $\mathbf{x}^k$. Let $M^0 = \{i \in M : J_i \cap B = \emptyset\}$ denote the set of rows that are uncovered in $\mathbf{x}^k$. With this notation, the feasible space of SCP can be defined as $X := \left\{\mathbf{x} \in \{0,1\}^n : \left|M^0\right| = 0\right\}$. In contrast to X, let us define $\overline{X} := \left\{\mathbf{x} \in \{0,1\}^n : \left|M^0\right| \leq \alpha m\right\} \setminus X$ as the "allowed infeasible space" of SCP, where α is a predetermined parameter chosen in $[0,1)$. A key feature of the proposed tabu search metaheuristic is its ability to escape from a locally optimal solution via an excursion into the allowed infeasible space. Owing to the monotone decreasing property of the objective function in $\mathbf{x}$, solutions in $\overline{X}$ are, usually, more attractive than the feasible solutions. Thus, even if $\mathbf{x}^k$ is a locally dominant prime cover, the search path will be able to escape from it to a remote, different prime cover $\mathbf{x}^{k+1}$ through a sequence of 1-neighborhood moves in $X \cup \overline{X}$.

Each composite move, from $\mathbf{x}^k$ to $\mathbf{x}^{k+1}$, is comprised of a sequence of a finite number of 1-neighborhood moves, selected in such a way that a monotonic property in the search path is preserved with respect to $|M^0|$, a measurement of the amount of infeasibility associated with $\mathbf{x}^k$ (see Figure 3.1-(a).) Furthermore, let us indicate with $I_j^- = \{i \in I_j : |J_i \cap B| = 1\}$ the set of rows uniquely covered, in the current solution $\mathbf{x}^k$, by a column $j \in B$, and with $I_j^+ = I_j \cap M^0$ the set of rows currently uncovered that would be covered by adding a column $j \in N \setminus B$ to the partial cover $\mathbf{x}^k$. Finally, let us indicate with $\mathcal{TL}$ the tabu list. The primal phase is made up by two sub-phases, which allow to implement a strategic oscillation mechanism around the boundaries of the feasible region:

1. ascending sub-phase: columns are constantly 'released' (set to zero), in such a way that, on the one hand, the objective function value monotonically improves and, on the other hand, the infeasibility level monotonically increases. During this phase, at each iteration, a non-tabu move ($j \notin \mathcal{TL}$) is chosen as $j_1 \in \Gamma^-$, where:

$$\Gamma^- := \left\{j_l \in B, I_{j_l}^- \neq \emptyset, j_l \notin \mathcal{TL} : c_{j_l}|I_{j_l}^-| \geq c_{j_l+1}|I_{j_l+1}^-|\right\}$$

2. descending sub-phase: columns are constantly 'added' (set to one), such that the infeasibility level monotonically decreases, eventually reaching a prime cover. During this phase, at each iteration, a non-tabu move ($j \notin \mathcal{TL}$) is chosen as $j_1 \in \Gamma^+$, where:

$$\Gamma^+ := \left\{j_l \in N \setminus B, I_{j_l}^+ \neq \emptyset, j_l \notin \mathcal{TL} : \overline{r}_{j_l} \leq \overline{r}_{j_l+1}\right\}$$

with $\overline{r}_j := c_j - \sum_{i \in I_j^+} u_i$.

We switch from one sub-phase to the other when the corresponding $\Gamma^\bullet$ is empty. To allow the search path to deviate from following a predetermined trajectory given by the use of the greedy merit functions, at each iteration we select j_1 probabilistically, as indicated by the scheme `Select_First_Move()`. Each move is, in turn, classified as either a regular, diversified or intensified move depending upon the way j_1 is selected.

Let $\mathbf{x}^*$ denote the best solution found so far and let $0 < \gamma_1 < \gamma_2 < 1$. Let $j\bullet$ indicate a 1-neighborhood move that sets the $j-$th component of $\mathbf{x}^t$ to 1 if $\bullet = +$ (a set covering move) and to 0 if $\bullet = -$ (a set releasing move) and let $\overline{\bullet}$ denote the move in the opposite direction of $\bullet$. Then, each composite move from $\mathbf{x}^t$ to $\mathbf{x}^{t+1}$ is comprised of a sequence of a finite number of 1-neighborhood moves, and the choice of the first move plays a critical role in the proposed meta-strategy.

Procedure `Select_First_Move()`;
Input: $\bullet$ ($= +$ or $-$), $\mathbf{x}^*$, $\mathbf{x}^t$, $\mathcal{TL}$
Output: j_1
  generate a random number γ in $[0, 1]$
  if $\gamma \in [0, \gamma_1]$ **then**
    select a move in $\Gamma^\bullet$ {`normal scheme`}
  else if $\gamma \in (\gamma_1, \gamma_2]$ **then**
    randomly select j_1 among $j \in N$, $j\bullet \notin \mathcal{TL}$ {`random scheme`}
  else
    $I_d\bullet := \left\{ j \in N, j\bullet \notin \mathcal{TL} : x_j^t \neq x_j^* \right\}$
    if $I_d\bullet = \emptyset$ **then**
      select a move in $\Gamma^\bullet$ {`memory-based scheme`}
    else
      randomly select j_1 from $I_d\bullet$
    end if
  end if

Remark. In order to allow for a more rigorous search of the solution space, we recur to three different strategies that define three search phases of the algorithm, namely the regular, diversification, and intensification phases. During the regular phase, we use $\gamma_1 = 0.8$ and $\gamma_2 = 0.9$ for the procedure `Select_First_Move()`, in such a way that the normal scheme is privileged above the random and memory-based scheme. For the diversification phase, we increase the probability of selecting a random move by using $\gamma_1 = 0.6$ and $\gamma_2 = 0.9$. Likewise, for the intensification phase, we use $\gamma_1 = 0.6$ and $\gamma_2 = 0.7$, thus granting a higher chance to the selection of a memory-based move.

Finally, the overall definition of a composite move is illustrated in Procedure `Composite_Move_Assignment()`. Denote by $\bullet\mathbf{e}_l$ a unit vector whose $l-$th component is -1 if $\bullet = -$ and $+1$ otherwise. We first identify the portion of the search space that is being explored. It the boundary of the "allowed infeasible region" has been reached or if feasibility has been restored (lines 2,3), then the sub-phase is inverted and the process is restarted. On the other hand, if the algorithm is currently in an ascending or descending phase, mainly within the $\overline{X}$ region, the first (ascending or descending) move is executed (line 5). The next steps are aimed at identifying a set of moves that go in the opposite direction of the first move. For example, if the algorithm is ascending into the "allowed infeasible space", we want to identify a set of descending moves in such a way that the net effect is still to uncover rows. To illustrate, if the first ascending move is such that column $j_1 \in B$ is set to 0, with the consequence that $|I_j^-|$ rows will be uncovered, a set of descending moves ζ^+ will be identified in such a way that the number of rows uncovered by j_1 is higher than the number of rows covered by all the moves in ζ^+. This is accomplished as illustrated in lines 6–10. Finally, lines 10 and 11 show how the composite move is executed and how the tabu list is updated.

Procedure `Composite_Move_Assignment()`;
Input: $\bullet$, $\mathbf{x}^t$, $\mathcal{TL}$, (core) problem instance
Output: $\bullet$, $\mathbf{x}^{t+1}$, $\mathcal{TL}$

1: call `Select_First_Move()` {`identify` j_1 `in` $\Gamma^\bullet$}
2: **if** $(\mathbf{x}^t + \bullet\mathbf{e}_{j_1} \notin X \cup \overline{X})$ or $(\mathbf{x}^t \in X)$ **then**
3: $\bullet \leftarrow \overline{\bullet}$, go to line 1 {`invert search direction`}
4: **end if**
5: $\mathcal{TL} \leftarrow \mathcal{TL} \cup \{\overline{j}_1\}$ {`set first move as tabu`}
6: **if** $|I_{j_1^t}^\bullet| \geq \sum_{j \in \Gamma^{\overline{\bullet}}} |I_j^{\overline{\bullet}}|$ **then**
7: $\zeta^{\overline{\bullet}} := \Gamma^{\overline{\bullet}}$ {`identify set of opposite moves`}
8: **else**
9: $\zeta^{\overline{\bullet}} := \left\{ j_1, \ldots, j_i \in \Gamma^{\overline{\bullet}} : |I_{j_1^t}^\bullet| \geq \sum_{j=j_1}^{j_i} |I_j^{\overline{\bullet}}|,\ |I_{j_1^t}^\bullet| < \sum_{j=j_1}^{j_i+1} |I_j^{\overline{\bullet}}| \right\}$
10: **end if**
11: $\mathbf{x}^{t+1} \leftarrow \mathbf{x}^t + \bullet\mathbf{e}_{j_1} + \sum_{j \in \zeta^{\overline{\bullet}}} \overline{\bullet}\mathbf{e}_j$ {`execute composite move`}
12: $\mathcal{TL} \leftarrow \mathcal{TL} \cup \{\overline{j}_l : j_l \in \zeta^{\overline{\bullet}}\}$ {`update tabu status`}

Remark. Note in the above that each move is selected in such a way that a monotonic property in the search path is preserved with respect to $|M^0|$, a measurement of the amount of infeasibility associated with $\mathbf{x}^t$.

Remark. Since the algorithm is especially designed to handle large-scale instances of SCP, we always work on a subset of columns $N_C \subset N$ and we employ pricing techniques to add or remove columns to and from N_C (See section7). Consequently, each occurrence of N in the definition of neighborhoods must be replaced by N_C.

4. Lagrangian Relaxation & Greedy Heuristics

The best known primal heuristic is the greedy one, which uses the reduced cost information provided by the dual phase to construct a prime cover. Balas and Ho, 1980, presented a list of scores based upon the column cost per row covered to create a prime cover. Vaasko and Wilson, 1984, selected a column to be added to the partial cover according to the value of a score function, randomly chosen among a pool of functions based upon the column cost per row covered. At every iteration the primal heuristic is run 30 times with randomly chosen score functions. Beasley, 1990, proposed a Lagrangian based primal heuristic scheme that extended the partial cover of the Lagrangian problem to a prime cover. A score based upon the column cost per row covered is used to rank the columns. Fisher and Kedia, 1990, proposed as score the reduced cost computed using only the multipliers of rows left uncovered, rather than the actual reduced cost. Bricker and Techapicjetvanich, 1993, studied the effectiveness of five different primal heuristic scores, based upon the column cost per row covered and the reduced cost per row covered, both the real and the modified reduced cost. Balas and Carrera, 1996, coupled the approach of Vaasko and Wilson, 1984, with a primal scheme that creates a prime cover extending the partial cover of the Lagrangian phase by choosing columns based upon their reduced cost. The primal scheme, as a byproduct, produces an improved dual vector.

Let M^0 denote the set of rows left uncovered by $\mathbf{x}$, and B denotes the set of columns fixed to 1 in the current (partial) cover $\mathbf{x}$. Let $|I_j \cap M^0|$ be the number of rows currently uncovered that would be covered by setting x_j to 1. During the dual Lagrangian phase we use the score

$$s(j) = \frac{c_j - \sum_{i \in I_j \cap M^0} u_i}{|I_j \cap M^0|},$$

as in Fisher and Kedia, 1990, within the simple heuristic described in Procedure `Greedy_Heuristic()`.

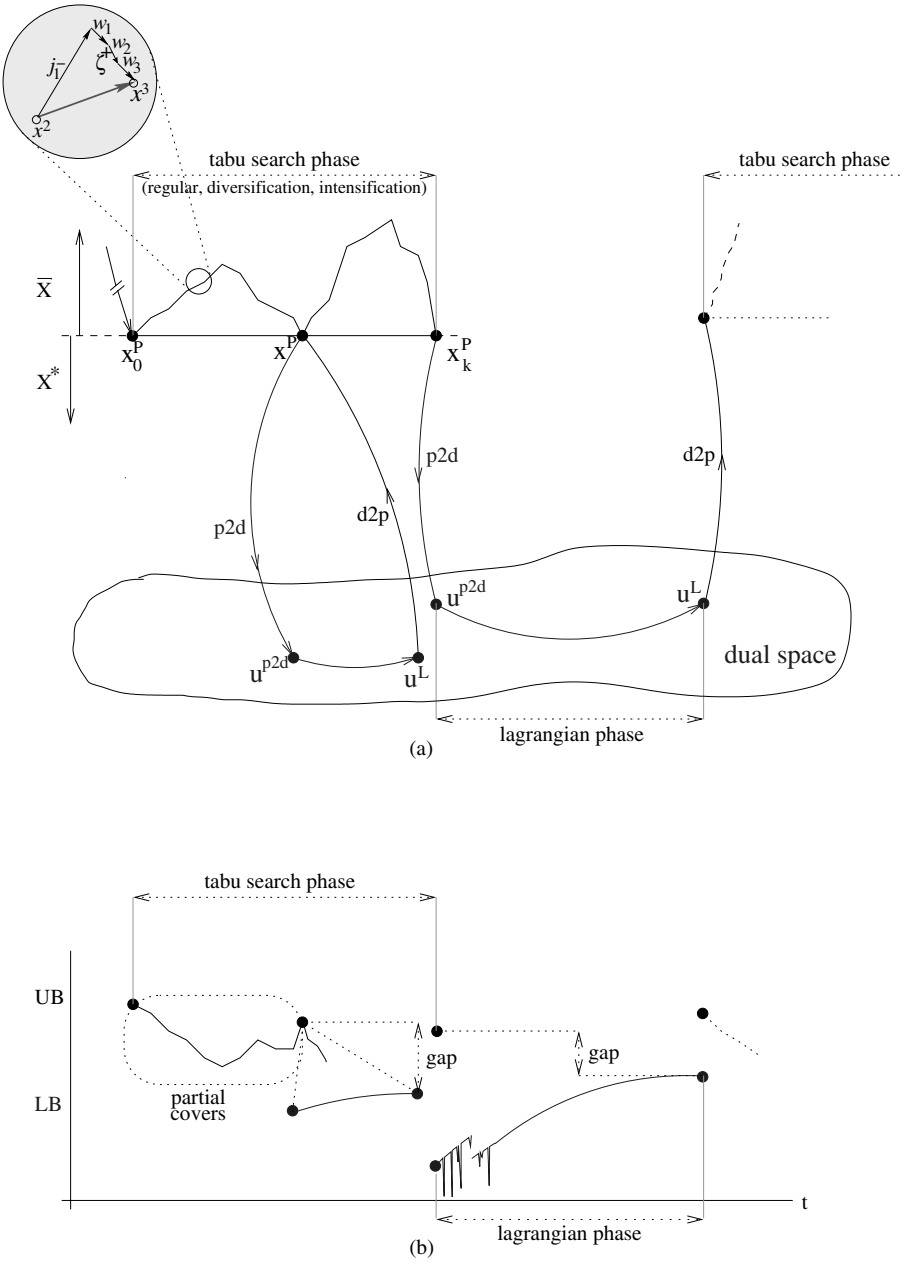

Figure 3.1. Dynamic Primal-and-Dual scheme: (a) from the feasibility point of view; (b) from the objective function value point of view.

Procedure `Greedy_Heuristic()`;
Input: u
Output: x
$M^0 \leftarrow M$; $\mathbf{x} \leftarrow \mathbf{0}$; $B \leftarrow \emptyset$ {`initialization`}
while $M^0 \neq \emptyset$ **do**
$j \leftarrow \underset{j \in N_C \setminus B}{\operatorname{argmin}}\{s(j)\}$ {`make a cover`}
$x_j \leftarrow 1$; $M^0 \leftarrow M^0 \setminus I_j$; $B \leftarrow B \cup \{j\}$ {`updating`}
end while
remove redundancy in $\mathbf{x}$ {`prime cover is obtained`}

5. Subgradient Optimization

The Lagrangian relaxation of SCP is defined as

$$L(\mathbf{x}, \mathbf{u}) = \min_{\mathbf{x} \in \{0,1\}^n} \sum_{j=1}^{n} r_j x_j + \sum_{i=1}^{m} u_i,$$

where r_j (the reduced cost for j, $j = 1, \ldots, n$) is defined as $c_j - \sum_{i \in I_j} u_i$ and requires $\mathbf{u}$ such that an optimal vector $\mathbf{x}^L$ minimizing the Lagrangian function can be computed by a standard technique:

$$x_j^L = \begin{cases} 1, & \text{if } c_j - \sum_{i \in I_j} u_i < 0 \\ 0, & \text{otherwise} \end{cases}, \; j \in N \tag{3.1}$$

It is worth noting that, since vector $\mathbf{x}^L$ is optimal to the Lagrangian problem, $L(\mathbf{u})$ provides a valid lower bound for SCP. For this reason, we are interested in finding the vector $\mathbf{u}$ that solves the Lagrangian dual problem, which is

$$L_D(\mathbf{u}) = \max_{\mathbf{u} \in \mathbb{R}_+^m} L(\mathbf{x}, \mathbf{u}).$$

Most successful approaches for SCP in the literature solve a series of Lagrangian relaxations of SCP and use the subgradient optimization technique to generate a near-optimal vector $\mathbf{u}$ for $L_D(\mathbf{u})$. For subgradient optimization, we use the formula of Held and Karp, 1971:

$$u_i^{k+1} = \max\left\{ u_i^k + \lambda \frac{UB - LB}{\|s(u^k)\|^2} s_i(u^k), \, 0 \right\}, \; i \in M, \tag{3.2}$$

where UB and LB are the upper and lower bounds of the optimum of SCP, λ is the step size parameter, and $s_i(\mathbf{x}^k) = 1 - \sum_{j \in J_i} x_j$ is

the component i of the subgradient. As in Caprara et al., 1999, $\mathbf{u}^0$ is initialized as

$$u_i^0 = \min_{j \in J_i} \frac{c_j}{|I_j|}, \; i \in M \tag{3.3}$$

and λ is updated after every $p = 20$ iterations, utilizing the best and worst lower bounds information obtained during the last p iterations. In addition, if the lower bound improvement in the last $4p$ iterations is below the threshold limit of 1%, we apply a "perturbation scheme" based upon the primal-to-dual scheme of Section 6 to enforce a drastic modification of the vector $\mathbf{u}$. We summarize the steps of the Lagrangian optimization phase in Procedure `Lagrangian_Optimization()`.

6. Primal-to-Dual Scheme

Let $\mathbf{x}^P$ and $\mathbf{x}^L$ denote a prime cover for SCP and a Lagrangian solution for a given vector $\mathbf{u} \in \mathbb{R}_+^m$, respectively. Denote by $z(\bullet)$ and $L(\bullet, \mathbf{u})$ the objective value of SCP and the value of the Lagrangian function evaluated at $\bullet$, respectively. Let $B^P = \{j \in N : x_j^P = 1\}$, $B^L = \{j \in N : x_j^L = 1\}$, $B^{LP} = B^L \setminus B^P$ and $B^{PL} = B^P \setminus B^L$.

Lema 3.1. *Suppose that* $\mathbf{x}^P \in \{0,1\}^n$, $\mathbf{x}^L \in \{0,1\}^n$, *and* $\mathbf{u} \in \mathbb{R}_+^m$ *satisfy* $u_i(\sum_{j \in J_i} x_j^P - 1) = 0$, *for all* $i \in M$. *Then,* $z(\mathbf{x}^P) - L(\mathbf{x}^L, \mathbf{u}) = \sum_{j \in B^{PL}} r_j - \sum_{j \in B^{LP}} r_j$.

Proof. We have

$$\begin{aligned}
L(\mathbf{x}^L, \mathbf{u}) &= \sum_{i \in M} u_i + \sum_{j \in B^L} (c_j - \sum_{i \in I_j} u_i) \\
&= \sum_{i \in M} u_i \sum_{j \in J_i} x_j^P + \sum_{j \in B^L} c_j - \sum_{j \in B^L} \sum_{i \in I_j} u_i \\
&= \sum_{j \in B^L} c_j + \sum_{j \in B^P} \sum_{i \in I_j} u_i - \sum_{j \in B^L} \sum_{i \in I_j} u_i \\
&= \sum_{j \in B^L} c_j + \sum_{j \in B^{PL}} \sum_{i \in I_j} u_i - \sum_{j \in B^{LP}} \sum_{i \in I_j} u_i,
\end{aligned}$$

Procedure `Lagrangian_Optimization()`;
Input: LB, UB, $\mathbf{x}^*$, $\mathbf{u}^0$
Output: LB, UB, $\mathbf{x}^*$

```
k ← 0                                   {Lagrangian iteration counter}
w ← 0                                   {perturbation scheme counters}
x^old = 0, λ^0 = 0.1
while lower bound termination tolerance is not met do
  if (k mod 20) = 0 then
    if LB_best − LB_worst > 0.01 LB_best then
      λ^k ← 0.5λ^k                                  {modify step size}
      w ← 0                  {reset perturbation scheme counter}
    else
      w ← w + 1          {increase perturbation scheme counter}
      if w < 4 then
        λ^k ← 1.5λ^k
      else
        λ^k = 0.1                      {apply perturbation scheme}
        call p2d(x^k, u^k)                              {see Section 6}
        if x^old = x^k then
          δ = random(0, 0.1u_max), where u_max = min_{i∈M}{u_i}
          u_i ← δu_i for randomly chosen 10% of u
        else
          x^old = x^k
        end if
      end if
    end if
    LB_best = LB_worst = L(x^L, u)          {set best and worst LB}
  end if                                    {new step size available}
  k ← k + 1              {increase Lagrangian iteration counter}
  update u via (3.2)           {perform k^th Lagrangian iteration}
  solve Lagrangian relaxation via (3.1)              {x^L is obtained}
  if L(x^L, u) > LB then
    LB ← L(x^L, u)                    {update lower bound on SCP}
  end if
  if L(x^L, u) > LB_best then
    LB_best ← L(x^L, u)                  {update best lower bound}
  else if L(x^L, u) < LB_worst then
    LB_worst ← L(x^L, u)                 {update worst lower bound}
  end if
end while              {lower bound termination tolerance met}
```

where the second equality is obtained via $u_i(\sum_{j\in J_i} x_j^P - 1) = 0, \forall i \in M$. Now, we have

$$\begin{aligned} z(\mathbf{x}^P) - L(\mathbf{x}^L, \mathbf{u}) &= \sum_{j\in B^P} c_j - \sum_{j\in B^L} c_j - \sum_{j\in B^{PL}} \sum_{i\in I_j} u_i + \sum_{j\in B^{LP}} \sum_{i\in I_j} u_i \\ &= \sum_{j\in B^{PL}} c_j - \sum_{j\in B^{LP}} c_j - \sum_{j\in B^{PL}} \sum_{i\in I_j} u_i + \sum_{j\in B^{LP}} \sum_{i\in I_j} u_i \\ &= \sum_{j\in B^{PL}} (c_j - \sum_{i\in I_j} u_i) - \sum_{j\in B^{LP}} (c_j - \sum_{i\in I_j} u_i) \\ &= \sum_{j\in B^{PL}} r_j - \sum_{j\in B^{LP}} r_j. \end{aligned}$$

□

Theorem 3.1 (Sufficient Conditions)**.** *Suppose that* $\mathbf{x}^P \in \{0,1\}^n$, $\mathbf{x}^L \in \{0,1\}^n$, *and* $\mathbf{u} \in \mathbb{R}_+^m$ *satisfy:*

$$\begin{aligned} &\textit{(i)} && u_i(\sum_{j\in J_i} x_j^P - 1) = 0, && \forall i \in M \\ &\textit{(ii)} && r_j = c_j - \sum_{i\in I_j} u_i = 0, && \forall j \in B^P \\ &\textit{(iii)} && r_j = c_j - \sum_{i\in I_j} u_i \geq 0, && \forall j \in N \setminus B^P \end{aligned}$$

Then, $\mathbf{x}^P$ *solves SCP to optimality.*

Proof. We need to show that both feasibility and optimality are ensured. Feasibility of $\mathbf{x}^P$ is enforced via conditions *(ii)* and *(iii)*, while optimality is ensured by conditions *(i)* and *(ii)*, along with $\mathbf{x}^* \in \{0,1\}^n$. □

The sufficient optimality conditions of $\mathbf{x}^*$ for SCP in Theorem 3.1 can be exploited in the derivation of a mechanism that constructs a "feasible" dual solution $\mathbf{u}$ that properly reflects the importance of each constraint of SCP with respect to the characteristics of $\mathbf{x}^P$.

First, note that Conditions (ii) and (iii) of Theorem 3.1, along with the requirement $\mathbf{u} \in \mathbb{R}_+^m$ give the feasibility of $\mathbf{u}$ to the dual linear program of the linearized SCP. Conditions (i) and (ii), along with $\mathbf{x}^* \in \{0,1\}^n$ ensure that the primal and dual solutions are optimal to their respective programs. Let $M^1 := \{i \in M : \sum_{j\in J_i} x_j^P = 1\}$ be the set of rows covered only once by a given solution $\mathbf{x}^P$. Furthermore, let $N_C \subseteq N$ be the set of columns in the current core problem, with $|N_C| \ll |N|$. Consider the following linear program:

$$\text{(p2d):} \quad \left| \begin{array}{lll} \min \quad g = & \displaystyle\sum_{j \in B^P \setminus B^L} (c_j - \sum_{i \in I_j} u_i) & \\ \text{s.t.} & c_j - \displaystyle\sum_{i \in I_j \cap M^1} u_i = 0, & j \in B^L \cap B^P \\ & c_j - \displaystyle\sum_{i \in I_j \cap M^1} u_i \geq 0, & j \in N_C \setminus B^P \\ & u_i \geq 0, & i \in M^1 \end{array} \right.$$

It is worth noting that (p2d) is a LP with $|(N_C \setminus B^P) \cup (B^P \cap B^L)|$ rows and $|M^1|$ columns. Also, note that the two non-trivial constraints of (p2d) set $u_i = 0$ for all $i \in M \setminus M^1$, and, through the minimization process, (p2d) modifies the remaining components of the vector $\mathbf{u}$ feasible to the dual of the linearized SCP that satisfies the sufficiency conditions of Theorem 3.1 "as much as possible" to yield $\mathbf{u}$ that reflects the characteristics of $\mathbf{x}^P$. It is easy to see that, if (p2d) has a feasible solution, such solution is dual feasible and, consequently, $\sum_i u_i$ provides a valid lower bound for SCP.

The following is an obvious consequence of (p2d) and Theorem 3.1:

Corollary 3.1. *If the optimum of (p2d) is equal to zero, then* $\mathbf{x}^P$ *solves SCP.*

The following also holds true:

Theorem 3.2. *Of all dual feasible* $\mathbf{u} \in \mathbb{R}^m_+$, $\mathbf{u}^*$ *obtained from solving (p2d) minimizes the gap* $z(\mathbf{x}^P) - L(\mathbf{x}^L, \mathbf{u}^*)$ *with respect to* $\mathbf{x}^P$ *and* $\mathbf{x}^L$ *and Condition (i) of Theorem 3.1.*

Proof. The dual feasibility of $\mathbf{u}^*$ is immediate. The formulation of (p2d) and Theorem 3.1 easily show that $z(\mathbf{x}^P) - L(\mathbf{x}^L, \mathbf{u}^*)$ is minimized by the $\mathbf{x}^P$ and $\mathbf{u}^*$ pair. □

7. Variable Fixing, Pricing and Core Problem Generation

In this section we present the variables fixing schemes for SCP. When probing x_j at 1, not only the Lagrangian multipliers of all rows $i \in I_j$ must be set to 0 but also all r_q, $q \in J_i$ for every $i \in I_j$, must be reduced to properly update the importance of the columns after setting $x_j = 1$. Let:

$$\delta_j^+ := \sum_{i \in I_j} u_i \times (|q \in J_i : r_q \leq 0| - 1)$$

The proposed score is embedded in the `Fixing_to_Zero()` scheme as in Balas and Carrera, 1996:

Procedure `Fixing_to_Zero()`;
Input: N, N_C, LB, UB, **u**
Output: N, N_C
  for $j \in N_C \setminus B^P$ **do**
    if $\lceil LB + r_j + \delta_j^+ \rceil \geq UB$ **then**
      $N_C \leftarrow N_C \setminus \{j\}$ {`eliminate column from core problem`}
      $N \leftarrow N \setminus \{j\}$ {`permanently eliminate column`}
    end if
  end for

Remark. Ceria et al., 1998, fixed a variable to zero if its reduced cost is greater than the gap between upper and lower bound. Balas and Carrera, 1996, computed a factor Δ_j for every column $j \in N \setminus B$ defined as the improvement in the value of the vector **u** obtained by fixing x_j to one. Subsequently, one column j is fixed to zero if $\lceil LB + r_j + \Delta_j \rceil \geq UB$.

To fix a column $j \in B^P$ permanently at 1 compute, for each $i \in I_j^-$, the variation of u_i required in order for at least another column $q \in J_i$, $q \neq j$ to have a non-positive reduced cost. This amount of modification required by u_i is

$$\delta_j^- = \sum_{i \in I_j^-} \min_{q \in J_i} \{r_q\},$$

and x_j, $j \in B^P$, can be permanently fixed to 1 if $\lceil LB - r_j + \delta_j^- \rceil \geq UB$. Procedure `Fixing_to_One()` summarizes the scheme used.

Procedure `Fixing_to_One()`;
Input: N, N_C, LB, UB, F, **u**
Output: N, N_C, F
  for $j \in B^P$ **do**
    if $\lceil LB - r_j + \delta_j^- \rceil \geq UB$ **then**
      $x_j \leftarrow 1$
      $N_C \leftarrow N_C \setminus \{j\}$ {`eliminate column from core problem`}
      $F \leftarrow F \cup \{j\}$ {`include column in fixed columns set` F}
    end if
  end for

To define core problems $N_C \subseteq N$, we employ a pricing scheme that resembles the one presented in Caprara et al., 1999. We first add to the core problem N_C all the columns whose reduced cost is less than 0.1. Subsequently, whenever possible, for each row $i \in M$, we add enough columns $j \in J_i$ to N_C in such a way that each row is covered by at least

5 columns in the core problem. These columns are added according to the reduced cost value.

8. Computational Results on SCP Benchmarks

In this section we present the results obtained by testing the algorithm on benchmark problems. The algorithm was implemented in C++ and compiled with the GNU C++ compiler with the -O2 option. The (p2d) problem is solved using the linear programming solver `Clp()` of COIN-OR Library (Lougee-Heimer, 2003). The computing platform used is a Linux workstation with Intel Pentium 4 1.1GHz processor and 256 Mb of RAM memory.

The parameters value for the tabu search metaheuristic are: $\theta = 2$ (number of excursion into the infeasible region for each TS phase), $\alpha = 0.1$ (maximum infeasibility allowed) and $\tau \in [\tau_{\min}, \tau_{\max}]$ (tabu tenure), where:

$$\tau_{\max} = \alpha \times |\mathbf{x}|, \;\; \tau_{\min} = 0.1\tau_{\max}$$

The value of τ is set to $\tau_{\min}$ every time a new best solution is found and increased every time dominated solutions are visited. The rationale behind such a choice is that, on the one hand, we want to thoroughly explore promising regions, in which "good" feasible solutions are found, while, on the other hand, we aim at escaping from unattractive regions by increasing the tabu tenure, thus forcing the algorithm to move toward a different region.

Computational results for Beasley's OR Library (Beasley, 1990) are not reported because the algorithm always finds the optimal, or the best known, solution. We only report, in Table 3.1, the results of Beasley's OR Library RAIL problems. The table shows that to 4 out of 7 instances the gap between upper and lower bound has been further reduced. In addition, for the two biggest instances a new best result is found, which indicates that the algorithm is especially suited for very large scale problems. Finally, Table 3.2 reports the results on the instances appeared in Wedelin, 1995. Out of 6 instances, for 4 of them the algorithm finds the optimal solution, and for the last two it finds a solution that is better than any other solution found so far.

9. Conclusions

We have presented a new dynamic scheme for large scale set covering problems. The backbone of the algorithm is a new primal-to-dual mechanism that, given any prime cover, constructs the dual feasible vector that better reflects the quality of the prime cover. Using this new mechanism, the algorithm updates the status of the search in the dual space

any time a new prime cover is found and vice versa, dynamically linking the primal intensive phase with the dual intensive phase.

When tested on benchmark problems, the algorithm improved the best known results on 6 instances, 2 of them by providing a better lower bound, 2 by finding a solution that is better that any other solution found so far, and 2 by improving both upper and lower bound.

Owing to the intensive use of primal-based schemes, the algorithm is especially suited for those instances of SCP with a number of rows much larger than the number of columns. Considering a classification problem, where a set of observations is partitioned into true and false, one wants to classify future observations based upon the value of certain attributes. The problem of selecting the smallest support set of attributes needed to classify a population can be formulated as SCP. However, if we indicate with m the number of observations, equally divided between positive and negative observations, the number of rows of SCP is of the order of $\mathcal{O}(m^2)$, leading to SCPs with $m \gg n$. For this reason, some new applications of SCP, such as probe selection problem for hybridization experiment as well as attributes identification and patterns selection in logical analysis of data, can be better tackled with a primal intensive approach rather that via the traditional Lagrangian based approach. This approach could be fostered by the design of a parallel algorithm for very large instances of SCP and, hence, applied to large problems in data mining and genetics.

Finally, it is also worth noting that the technique proposed in Section 3 of this paper, dealing with the swap of columns within and outside of the current solution is a generalization of oscillation mechanisms as well as k-flip mechanisms, such as the ones of Glover and Kochenberger, 1996, Chu and Beasley, 1998, Caserta et al., 2006 or Yagiura et al., 2006. The results obtained on SCPs by these authors along with the promising results of the proposed scheme endorse the idea that oscillating mechanisms (continually crossing the boundaries of the feasible region) are very powerful ingredients of a metaheuristic scheme when it comes to solving large scale combinatorial optimization problems.

References

Balas, E. and Carrera, M. C. (1996). A Dynamic Subgradient-Based Branch-and-Bound Procedure for Set Covering Problem. *Operations Research*, 44(6):875–890.

Balas, E. and Ho, A. (1980). Set Covering Algorithms Using Cutting Planes, Heuristics and Subgradient Optimization: a Computational Study. *Mathematical Programming Study*, 112:37–60.

Table 3.1. Results on the RAIL test instances from Beasley's OR-Library

Name	Size	Best in Literature			PD-SCP		
		LB	UB	Time	LB	UB	Time
RAIL582	582×55,515	210	211	570[♯]	210	211	131
RAIL507	507×63,009	173	174	817[†]	173	174	139
RAIL516	516×47,311	182	182	3000[♯]	182	182	217
RAIL2536	2536×1,081,841	685	691	10000[§]	**687**	691	338
RAIL2586	2,586×920,683	937	948	1183[§]	**939**	948	399
RAIL4284	4284 ×1,092,610	1051	1065	10000[§]	**1055**	**1063**	1022
RAIL4872	4,872×968,672	1,509	1,534	4566[§]	**1514**	**1532**	1166

[♯]: Caprara et al. (1999) - time in PC486/33 CPU seconds.
[§]: Caprara et al. (1999) - time in HP735/125 CPU seconds.
[†]: Ceria et al. (1998) - time in PC486/33 CPU seconds.

Table 3.2. Results on instances from Wedelin (1995)

Name	Range	Best in Literature		PD-SCP	
		UB	Time	UB	Time
b727scratch	29×157	94,400 [†,§]	0.3	94,400	0.1
alitalia	118×1,165	27,258,300[†,§]	6.2	27,258,300	2.1
a320	199×6,931	12,620,100[†,§]	79.5	12,620,100	37.3
a320coc	235×18,753	14,495,500[†]	1,023.7	14,495,500	228.1
sasjump	742×10,370	7,339,537 [§]	396.3	**7, 339, 521**	221.7
sasd9imp2	1,366×25,032	5,262,190 [†]	1,579.7	**5, 262, 140**	1,066.3

[§]: Caprara et al. (1999) - time given in DECstation 5000/240 CPU seconds.
[†]: Wedelin (1995) - time given in DECstation 5000/240 CPU seconds.

Beasley, J. E. (1990). A Lagrangian Heuristic for Set Covering Problems. *Naval Research Logistics*, 37:151–164.

Borneman, J., Chrobak, M., Della Vedova, G., Figueroa, A., and Jiang, T. (2001). Probe Selection Algorithms with Applications in the Analysis of Microbial Community. *Bioinformatics - Discovery Notes*, 1(1):1–9.

Boros, E., Hammer, P. L., Ibaraki, T., Kogan, A., Mayoraz, E., and Muchnik, I. (1996). An Implementation of Logical Analysis of Data. Technical report, Rutgers University, New Brunswick, NJ - 08903-5062.

Bricker, D. and Techapicjetvanich, K. (1993). Investigation of Lagrangian Heuristics for Set Covering Problems. Technical Report Iowa City, IA, 52242, University of Iowa.

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

Caserta, M., Quiñonez Rico, E., and Márquez Uribe, A. (2006). A Cross Entropy Algorithm for the Knapsack Problem with Setups. *Computers and Operations Research*.

Caserta, M. and Ryoo, H. S. (2001). Efficient Tabu Search-Based Procedure for Optimal Redundancy Allocation in Complex System Reliability. In *Proc. 5th Intl Conference on Optimization: Techniques and Applications*, pages 592–599.

Ceria, S., Nobili, P., and Sassan, A. (1998). A Lagrangian-based Heuristic for Large-scale Set Covering Problems. *Mathematical Programming Ser B*, 81:215–228.

Chu, P. C. and Beasley, J. E. (1998). A Genetic Algorithm for the Multidimensional Knapsack Problem. *Journal of Heuristics*, 4(1):63–86.

Dom, M., Guo, J., Niedermeier, R., and Wernicke, S. (2006). Minimum Membership Set Covering and the Consecutive Ones Property. In Arge, L. and Freivalds, R., editors, *SWAT 2006 - 10th Scandinavian Workshop on Algorithm Theory*, volume 4059 of *Springer LNCS*, page 337.

Downey, R. G. and Fellows, M. R. (1999). *Parameterized Complexity*. Monographs in Computer Science. Springer.

Fisher, M. L. and Kedia, P. (1990). Optimal Solutions of Set Covering/Partitioning Problems using Dual Heuristics. *Management Science*, 36:674–688.

Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: a Guide to the Theory of NP-Completeness*. Freeman.

Glover, F. and Kochenberger, G. A. (1996). *Critical Event Tabu Search for Multidimensional Knapsack Problems*, chapter In Meta-Heuristics:

Theory and Applications – I.H. Osman and J.P. Kelly, pages 407–427. Kluwer Academic Publishers.

Held, M. and Karp, R. M. (1971). The Traveling Salesman Problem and Minimum Spanning Tree: Part II. *Mathematical Programming*, 1:6–25.

Lougee-Heimer, R. (2003). The Common Optimization INterface for Operations Research. *IBM Journal of Research and Developmentg*, 47:57–66.

Niedermeier, R. (2006). *Invitation to Fixed-Parameter Algorithms*. Oxforf University Press.

Vaasko, F. J and Wilson, G. R. (1984). An Efficient Heuristic for Large Set Covering Problems. *Naval Research Logistics*, 31:163–171.

Wedelin, D. (1995). An Algorithm for Large Scale 0-1 Integer Programming with Applications to Airline Crew Scheduling. *Annals of Operational Research*, 57:283–301.

Weihe, K. (1998). Covering Trains by Stations or the Power of Data Reduction. In Battiti, R. and Bertosi, A. A., editors, *OnLine Proceedings of the 1st Workshop on Algorithms and Experiments (ALEX'98) - Trento, Italy*, pages 1–8.

Yagiura, M., Kishida, M., and Ibaraki, T. (2006). A 3-Flip Neighborhood Local Search for the Set Covering Problem. *European Journal of Operational Research*, 172:472–499.