A Heuristic Algorithm for the Set Multicover Problem with Generalized Upper Bound Constraints

Shunji Umetani[1(✉)], Masanao Arakawa[2], and Mutsunori Yagiura[3]

[1] Osaka University, Suita 565-0871, Japan
umetani@ist.osaka-u.ac.jp
[2] Fujitsu Limited, Kawasaki 211-8588, Japan
arakawa.masanao@jp.fujitsu.com
[3] Nagoya University, Nagoya 464-8601, Japan
yagiura@nagoya-u.jp

Abstract. We consider an extension of the set covering problem (SCP) introducing (i) multicover and (ii) generalized upper bound (GUB) constraints that arise in many real applications of SCP. For this problem, we develop a 2-flip neighborhood local search algorithm with a heuristic size reduction algorithm, in which a new evaluation scheme of variables is introduced taking account of GUB constraints. According to computational comparison with the latest version of a mixed integer programming solver, our algorithm performs quite effectively for various types of instances, especially for very large-scale instances.

1 Introduction

The set covering problem (SCP) is one of representative combinatorial optimization problems. We are given a ground set of m elements $i \in M = \{1, \ldots, m\}$, n subsets $S_j \subseteq M$ ($|S_j| \geq 1$) and costs $c_j (> 0)$ for $j \in N = \{1, \ldots, n\}$. We say that $X \subseteq N$ is a cover of M if $\bigcup_{j \in X} S_j = M$ holds. The goal of SCP is to find a minimum cost cover X of M. The SCP is formulated as a 0–1 integer programming (IP) problem as follows:

$$\begin{aligned} \text{min.} \quad & \sum_{j \in N} c_j x_j \\ \text{s.t.} \quad & \sum_{j \in N} a_{ij} x_j \geq 1, \quad i \in M, \\ & x_j \in \{0, 1\}, \qquad j \in N, \end{aligned} \tag{1}$$

where $a_{ij} = 1$ if $i \in S_j$ holds and $a_{ij} = 0$ otherwise, and $x_j = 1$ if $j \in X$ holds and $x_j = 0$ otherwise, respectively.

The SCP is often referred in the literature that it has many important applications [2], e.g., crew scheduling, vehicle routing, facility location, and logical analysis of data. However, it is often difficult to formulate problems in real

G. Nicosia and P. Pardalos (Eds.): LION 7, LNCS 7997, pp. 75–80, 2013.
DOI: 10.1007/978-3-642-44973-4_9,

applications into the SCP, because they often have additional side constraints in practice. Most practitioners accordingly formulate them into general mixed integer programming (MIP) problem and apply general purpose solvers, which are usually less efficient compared to solvers specially tailored to SCP.

In this paper, we consider an extension of SCP introducing (i) multicover and (ii) generalized upper bound (GUB) constraints, which arise in many real applications of SCP. The multicover constraint is a generalization of covering constraint, in which each element $i \in M$ must be covered at least $b_i \in \mathbb{Z}_+$ ($\mathbb{Z}_+$ is the set of non-negative integers) times. GUB constraint is defined as follows. We are given a partition $\{G_1, \ldots, G_k\}$ of N ($\forall h \neq h', G_h \cap G_{h'} = \emptyset, \bigcup_{h=1}^{k} G_h = N$). For each block $G_h \subseteq N$ ($h \in K = \{1, \ldots, k\}$), the number of selected subsets S_j ($j \in G_h$) is constrained to be at most $d_h (\leq |G_h|)$. We call this problem the set multicover problem with GUB constraints (SMCP-GUB).

The SMCP-GUB is NP-hard, and the (supposedly) simpler problem of judging the existence of a feasible solution is NP-complete. We accordingly consider the following formulation of SMCP-GUB that allows violations of the multicover constraints and introduces a penalty function with a penalty weight vector $\boldsymbol{w} = (w_1, \ldots, w_m) \in \mathbb{R}_+^m$:

$$\begin{aligned} \text{min.} \quad & z(\boldsymbol{x}) = \sum_{j \in N} c_j x_j + \sum_{i \in M} w_i y_i \\ \text{s.t.} \quad & \sum_{j \in N} a_{ij} x_j + y_i \geq b_i, && i \in M, \\ & \sum_{j \in G_h} x_j \leq d_h, && h \in K, \\ & x_j \in \{0, 1\}, && j \in N, \\ & y_i \in \{0, \ldots, b_i\}, && i \in M. \end{aligned} \tag{2}$$

For a given $\boldsymbol{x} \in \{0, 1\}^n$, we can easily compute an optimal $\boldsymbol{y}$ by $y_i = \max\{b_i - \sum_{j \in N} a_{ij} x_j, 0\}$. We note that when $\boldsymbol{y}^* = \boldsymbol{0}$ holds for an optimal solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ of SMCP-GUB under the soft multicover constraints, $\boldsymbol{x}^*$ is also optimal under the original (hard) multicover constraints. Moreover, for an optimal solution $\boldsymbol{x}^*$ under hard multicover constraints, $(\boldsymbol{x}^*, \boldsymbol{0})$ is also optimal with respect to soft multicover constraints if the values of w_i are sufficiently large, e.g., if $w_i > \sum_{j \in N} c_j$ holds for all $i \in M$. We accordingly set $w_i = \sum_{j \in N} c_j + 1$ for all $i \in M$.

In this paper, we proposes a 2-flip neighborhood local search algorithm with an efficient mechanism to find improved solutions. The above generalization of SCP substantially extends the variety of its applications. However, GUB constraints often make the pricing method less effective (which is known to be very effective for large-scale instances of SCP), because GUB constraints prevent solutions from containing highly evaluated variables together. To overcome this, we develop a heuristic size reduction algorithm, in which a new evaluation scheme of variables is introduced taking account of GUB constraints.

2 Lagrangian Relaxation and Subgradient Method

For a given vector $\boldsymbol{u} = (u_1, \ldots, u_m) \in \mathbb{R}_+^m$, called the Lagrangian multiplier vector, the Lagrangian relaxation of SMCP-GUB is defined as follows:

$$\begin{aligned}
\text{min. } z_{\text{LR}}(\boldsymbol{u}) &= \sum_{j \in N} c_j x_j + \sum_{i \in M} w_i y_i + \sum_{i \in M} u_i \left(b_i - \sum_{j \in N} a_{ij} x_j - y_i \right) \\
&= \sum_{j \in N} \left(c_j - \sum_{i \in M} a_{ij} u_i \right) x_j + \sum_{i \in M} y_i (w_i - u_i) + \sum_{i \in M} b_i u_i \qquad (3) \\
\text{s.t.} \quad & \sum_{j \in G_h} x_j \le d_h, \qquad h \in K, \\
& x_j \in \{0, 1\}, \qquad j \in N, \\
& y_i \in \{0, \ldots, b_i\}, \quad i \in M,
\end{aligned}$$

where we call $\tilde{c}_j(\boldsymbol{u}) = c_j - \sum_{i \in M} a_{ij} u_i$ the Lagrangian cost associated with column $j \in N$. For any $\boldsymbol{u} \in \mathbb{R}_+^m$, $z_{\text{LR}}(\boldsymbol{u})$ gives a lower bound on the optimal value of SMCP-GUB $z(\boldsymbol{x}^*)$. The problem of finding a Lagrangian multiplier vector $\boldsymbol{u}$ that maximizes $z_{\text{LR}}(\boldsymbol{u})$ is called the Lagrangian dual problem.

A common approach to compute a near optimal Lagrangian multiplier vector $\boldsymbol{u}$ is the subgradient method. When huge instances of SCP are solved, the computing time spent on the subgradient method becomes very large if a naive implementation is used. Caprara et al. [1] developed a variant of pricing method on the subgradient method. They define a dual core problem consisting of a small subset of columns $C_d \subset N$ ($|C_d| \ll |N|$), chosen among those having the lowest Lagrangian costs $\tilde{c}_j(\boldsymbol{u})$ ($j \in C_d$), and iteratively update the dual core problem in a similar fashion to that used for solving large scale LP problems. In order to solve huge instances of SMCP-GUB, we also introduce their pricing method into the basic subgradient method (BSM) described in [3].

3 The 2-flip Neighborhood Local Search Algorithm

The local search (LS) starts from an initial solution $\boldsymbol{x}$ and repeats replacing $\boldsymbol{x}$ with a better solution $\boldsymbol{x}'$ in its neighborhood $\text{NB}(\boldsymbol{x})$ until no better solution is found in $\text{NB}(\boldsymbol{x})$. For a positive integer r, the r-flip neighborhood $\text{NB}_r(\boldsymbol{x})$ is defined by $\text{NB}_r(\boldsymbol{x}) = \{\boldsymbol{x}' \in \{0,1\}^n \mid d(\boldsymbol{x}, \boldsymbol{x}') \le r\}$, where $d(\boldsymbol{x}, \boldsymbol{x}') = |\{j \in N \mid x_j \ne x'_j\}|$ is the Hamming distance between $\boldsymbol{x}$ and $\boldsymbol{x}'$. In other words, $\text{NB}_r(\boldsymbol{x})$ is the set of solutions obtained from $\boldsymbol{x}$ by flipping at most r variables. In our LS, the r is set to 2. In order to improve efficiency, our LS searches $\text{NB}_1(\boldsymbol{x})$ first, and $\text{NB}_2(\boldsymbol{x}) \setminus \text{NB}_1(\boldsymbol{x})$ only if $\boldsymbol{x}$ is locally optimal with respect to $\text{NB}_1(\boldsymbol{x})$.

Yagiura et al. [4] developed an LS with the 3-flip neighborhood for SCP. They derived conditions that reduce the number of candidates in $\text{NB}_2(\boldsymbol{x}) \setminus \text{NB}_1(\boldsymbol{x})$ and $\text{NB}_3(\boldsymbol{x}) \setminus \text{NB}_2(\boldsymbol{x})$ without sacrificing the solution quality. However, those conditions are not applicable to the 2-flip neighborhood for SMCP-GUB because

of GUB constraints. We therefore propose new conditions that reduce the number of candidates in $\mathrm{NB}_2(\boldsymbol{x})\backslash\mathrm{NB}_1(\boldsymbol{x})$ taking account of GUB constraints. As a result, the number of solutions searched by our algorithm becomes $\mathrm{O}(n + k\nu + n'\tau)$ while the size of NB_2 is $O(n^2)$, where $\nu = \max_{j\in N} |S_j|$, $n' = \sum_{j\in N} x_j$ and $\tau = \max_{j\in N} \sum_{i\in S_j} |N_i|$ for $|N_i| = \{j \in N | i \in S_j\}$.

Since the region searched in a single application of LS is limited, LS is usually applied many times. When a locally optimal solution is obtained, a standard strategy of our algorithm is to update penalty weights and to resume LS from the obtained locally optimal solution. We accordingly evaluate solutions with an alternative evaluation function $\hat{z}(\boldsymbol{x})$, where the original penalty weight vector $\boldsymbol{w}$ is replaced with $\hat{\boldsymbol{w}} = (\hat{w}_1, \ldots, \hat{w}_m) \in \mathbb{R}^m_+$. Our algorithm iteratively applies LS, updating the penalty weight vector $\hat{\boldsymbol{w}}$ after each call to LS.

Starting from the original penalty weight vector $\hat{\boldsymbol{w}} \leftarrow \boldsymbol{w}$, the penalty weight vector $\hat{\boldsymbol{w}}$ is updated as follows. Let $\boldsymbol{x}^{\mathrm{best}}$ denote the best feasible solution with respect to the original objective function $z(\boldsymbol{x})$. If the previous locally optimal solution $\boldsymbol{x}$ satisfies $\hat{z}(\boldsymbol{x}) \geq z(\boldsymbol{x}^{\mathrm{best}})$, our algorithm uniformly decreases the penalty weights $\hat{w}_i$ $(i \in M)$. Otherwise, our algorithm increases the penalty weights $\hat{w}_i$ $(i \in M)$ in proportion to the amount of violation of the ith multi-cover constraint.

4 Heuristic Reduction of Problem Sizes

For a near optimal Lagrangian multiplier vector $\boldsymbol{u}$, the Lagrangian costs $\tilde{c}_j(\boldsymbol{u})$ give reliable information on the overall utility of selecting columns $j \in N$ for SCP. Based on this property, the Lagrangian costs $\tilde{c}_j(\boldsymbol{u})$ are often utilized to solve huge instances of SCP, e.g., several heuristic algorithms successively solve a number of subproblems, called primal core problems, consisting of a small subset of columns $C_p \subset N$ ($|C_p| \ll |N|$), which are chosen among those having low Lagrangian costs $\tilde{c}_j(\boldsymbol{u})$ [1,2,4].

The Lagrangian costs $\tilde{c}_j(\boldsymbol{u})$ are unfortunately unreliable for selecting columns $j \in N$ for SMCP-GUB, because GUB constraints often prevent solutions from containing more than d_h variables x_j with the lowest Lagrangian costs $\tilde{c}_j(\boldsymbol{u})$. To overcome this, we develop an evaluation scheme of columns $j \in N$ for SMCP-GUB taking account of GUB constraints. The main idea of our algorithm is that we modify the Lagrangian costs $\tilde{c}_j(\boldsymbol{u})$ to reduce the number of redundant columns $j \in C_p$ resulting from GUB constraints.

For each block G_h $(h \in K)$, let γ_h be the value of the $(d_h + 1)$st lowest Lagrangian cost $\tilde{c}_j(\boldsymbol{u})$ among those for columns in G_h, where we set $\gamma_h \leftarrow 0$ if $d_h = |G_h|$ holds. We then define a score $\hat{c}_j(\boldsymbol{u})$ for a column $j \in G_h$ by $\hat{c}_j(\boldsymbol{u}) = \tilde{c}_j(\boldsymbol{u}) - \gamma_h$ if $\gamma_h < 0$ holds, and $\hat{c}_j(\boldsymbol{u}) = \tilde{c}_j(\boldsymbol{u})$ otherwise. That is, we normalize the Lagrangian costs $\tilde{c}_j(\boldsymbol{u})$ so that at most d_h columns have negative scores $\hat{c}_j(\boldsymbol{u}) < 0$ for each block G_h $(h \in K)$. Let $n' = \sum_{j\in N} x_j$ be the number of selected subsets for a solution $\boldsymbol{x}$. Given a solution $\boldsymbol{x}$ and a Lagrangian multiplier vector $\boldsymbol{u}$, a primal core problem is defined by a subset $C_p \subset N$ consisting of (i) columns $j \in N_i$ with the b_i lowest scores $\hat{c}_j(\boldsymbol{u})$ for each $i \in M$, and (ii) columns $j \in N$ with the $10n'$ lowest scores $\hat{c}_j(\boldsymbol{u})$.

Table 1. The benchmark instances for SMCP-GUB and time limits for our algorithm LS-SR and the MIP solver CPLEX (in seconds)

Instance	Rows	Columns	Density	Instance types ($d_h/\|G_h\|$)				Time limit	
			(%)	Type1	Type2	Type3	Type4	LS-SR	CPLEX
G.1–G.5	1000	10,000	2.0	1/10	10/100	5/10	50/100	600	3600
H.1–H.5	1000	10,000	5.0	1/10	10/100	5/10	50/100	600	3600
I.1–I.5	1000	50,000	1.0	1/50	10/500	5/50	50/500	600	3600
J.1–J.5	1000	100,000	1.0	1/50	10/500	5/50	50/500	600	3600
K.1–K.5	2000	100,000	0.5	1/50	10/500	5/50	50/500	1200	7200
L.1–L.5	2000	200,000	0.5	1/50	10/500	5/50	50/500	1200	7200
M.1–M.5	5000	500,000	0.25	1/50	10/500	5/50	50/500	3000	18,000
N.1–N.5	5000	1,000,000	0.25	1/100	10/1000	5/100	50/1000	3000	18,000

5 Computational Results

We first prepared eight classes of random instances for SCP, where each class has five instances. We denote instances in class G as G.1, ..., G.5, and other instances in classes H–N similarly. The summary of these instances are given in Table 1, where the density is defined by $\sum_{i\in M}\sum_{j\in N} a_{ij}/mn$ and the costs c_j are random integers taken from interval $[1, 100]$. For each SCP instance, we generate four types of SMCP-GUB instances with different values of parameters d_h and $|G_h|$ as shown in Table 1, where all blocks G_h $(h \in K)$ have the same size $|G_h|$ and upper bound d_h for each instance. Here, the right-hand sides of multicover constraints b_i are random integers taken from interval $[1, 5]$.

We compared our algorithm, called the local search algorithm with the heuristic size reduction (LS-SR), with one of the latest mixed integer program (MIP) solver called CPLEX12.3, where they were tested on an IBM-compatible personal computer (Intel Xeon E5420 2.5 GHz, 4 GB memory) and were run on a single thread. Table 1 also shows the time limits in seconds for LS-SR and CPLEX12.3, respectively. We tested two variants of LS-SR: LS-SR1 evaluates variables x_j with the proposed score $\hat{c}_j(\boldsymbol{x})$, and LS-SR2 uses the Lagrangian cost $\tilde{c}_j(\boldsymbol{x})$ in the heuristic reduction of problem sizes. We illustrate in Fig. 1 their comparison for each type of SMCP-GUB instances with respect to the relative gap $\frac{z(\boldsymbol{x})-z_{\mathrm{LP}}}{z_{\mathrm{LP}}} \times 100$, where z_{LP} is the optimal value of LP relaxation for SMCP-GUB. The horizontal axis shows the classes of instances G–N, and the vertical axis shows the average relative gap for five instances of each class.

We first observe that LS-SR1 and LS-SR2 achieve better upper bounds than CPLEX12.3 for types 3 and 4 instances, especially large instances with 10,000 variables or more. One of the main reasons for this is that the proposed algorithms evaluate a series of candidate solutions efficiently while CPLEX12.3 consumes much computing time for solving LP relaxation problems. We also observe that LS-SR1 achieves much better upper bounds than those of LS-SR2 and CPLEX12.3 for types 1 and 2 instances.

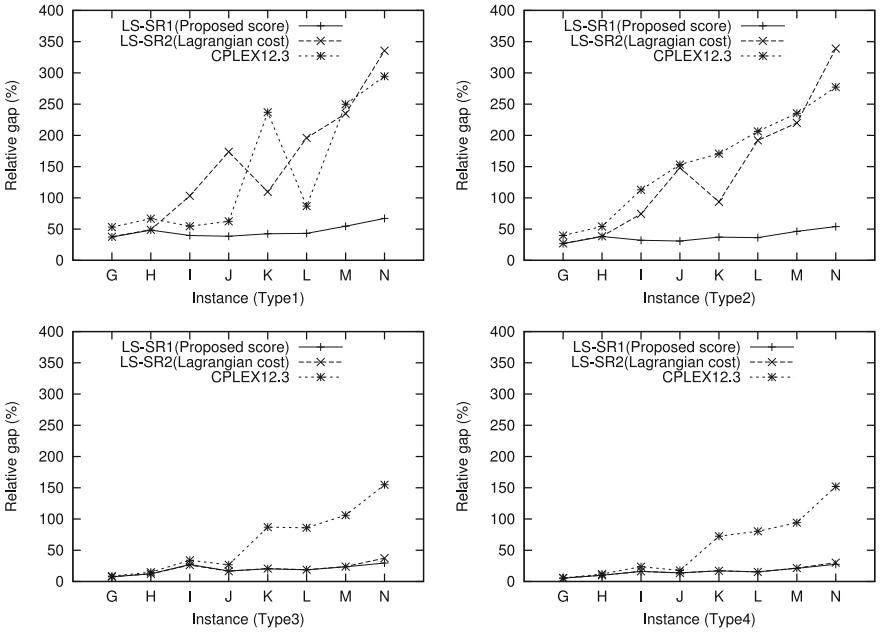

Fig. 1. Comparison of LS-SR and CPLEX12.3 on each instance type

6 Conclusion

In this paper, we considered an extension of SCP called the set multicover problem with the generalized upper bound constraints (SMCP-GUB). For this problem, we develop a 2-flip neighborhood local search algorithm with a heuristic size reduction algorithm, in which a new evaluation scheme of variables is introduced taking account of GUB constraints. According to computational comparison on benchmark instances with the latest version of a MIP solver called CPLEX12.3, our algorithm performs quite effectively for various types of instances, especially for very large-scale instances.

References

1. Caprara, A., Fischetti, M., Toth, P.: A heuristic method for the set covering problem. Oper. Res. **47**, 730–743 (1999)
2. Caprara, A., Toth, P., Fischetti, M.: Algorithms for the set covering problem. Ann. Oper. Res. **98**, 353–371 (2000)
3. Umetani, S., Yagiura, M.: Relaxation heuristics for the set covering problem. J. Oper. Res. Soc. Jpn. **50**, 350–375 (2007)
4. Yagiura, M., Kishida, M., Ibaraki, T.: A 3-flip neighborhood local search for the set covering problem. Eur. J. Oper. Res. **172**, 472–499 (2006)