An approximation algorithm for the partial covering 0–1 integer program

Yotaro Takazawa [*] Shinji Mizuno [†] Tomonari Kitahara [‡]

submitted January, 2016
revised July, 2017

Abstract

The partial covering 0–1 integer program (PCIP) is a relaxed problem of the covering 0–1 integer program (CIP) such that some fixed number of constraints may not be satisfied. This type of relaxation is also discussed in the partial set multi-cover problem (PSMCP) and the partial set cover problem (PSCP). In this paper, we propose an approximation algorithm for PCIP by extending an approximation algorithm for PSCP by Gandhi et al. [5].

keywords: Approximation algorithms, Partial covering 0–1 integer program, Primal-dual method.

1 Introduction

The covering 0–1 integer program (CIP) is a well-known combinatorial optimization problem and formulated as

$$\text{CIP}\left|\begin{array}{lll} \min & \sum_{j\in N} c_j x_j & \\ \text{s.t.} & \sum_{j\in N} u_{ij} x_j \ge d_i, & \forall i \in M, \\ & x_j \in \{0,1\}, & \forall j \in N, \end{array}\right. \tag{1}$$

where $M = \{1,\dots,m\}$, $N = \{1,\dots,n\}$, $c_j \ge 0$ $(j \in N)$, $u_{ij} \ge 0$ $(i \in M, j \in N)$ and $d_i > 0$ $(i \in M)$ are given data and x_j $(j \in N)$ are 0–1 variables. When the

[*]Department of Industrial Engineering and Management, Tokyo Institute of Technology
[†]Department of Industrial Engineering and Economics, Tokyo Institute of Technology
[‡]Department of Industrial Engineering and Economics, Tokyo Institute of Technology

problem is relaxed such that some fixed number $p \in \{0, 1, \ldots, m\}$ of constraints $\sum_{j \in N} u_{ij} x_j \geq d_i$ $(i \in M)$ may not be satisfied, the resulting problem is called the partial covering 0–1 integer program, which is formulated as

$$\text{PCIP}\left|\begin{array}{lll} \min & \sum_{j \in N} c_j x_j & \\ \text{s.t.} & \sum_{j \in N} u_{ij} x_j + d_i t_i \geq d_i, & \forall i \in M, \\ & \sum_{i \in M} t_i \leq p, & \\ & x_j \in \{0, 1\}, & \forall j \in N, \\ & t_i \in \{0, 1\}, & \forall i \in M. \end{array}\right. \tag{2}$$

For a given minimization problem having an optimal solution, an algorithm is called an α-approximation algorithm if it runs in polynomial time and produces a feasible solution whose objective value is less than or equal to α times the optimal value.

PCIP generalizes some important problems for which approximation algorithms are proposed as shown in Table 1, where

$$\begin{array}{rcl} f & = & \max_{i \in M} |\{j \in N \mid u_{ij} > 0\}|, \\ \Delta & = & \max_{i \in N} |\{i \in M \mid u_{ij} > 0\}|, \\ H(\Delta) & = & 1 + \frac{1}{2} + \cdots + \frac{1}{\Delta}, \\ d_{\max} & = & \max_{i \in M} d_i, \\ d_{\min} & = & \min_{i \in M} d_i, \\ \eta & = & \Delta \frac{\max_{j \in N} c_j}{\min_{j \in N} c_j} \frac{d_{\max}}{d_{\min}}, \\ \gamma & = & \frac{m}{m - p\eta}, \\ g & = & \max\left\{\frac{\Delta}{m-p}\left(\frac{1}{f - d_{\max}} + \frac{d_{\max}}{d_{\min}}\right), \frac{f}{d_{\min}} + \left(1 - \frac{1}{d_{\max}}\right)p, p + 1\right\}. \end{array} \tag{3}$$

Table 1: Special cases in PCIP

Problems	Restrictions in PCIP	Approximation ratios
PCIP	-	· $\max\{f, p+1\}$ (this paper)
Covering 0–1 integer program (CIP)	$p = 0$	· f [3, 4, 7] · $O(\log m)$ [6]
Partial set multi-cover problem (PSMCP)	$u_{ij} \in \{0, 1\}$, d_i is a positive integer	· $\gamma H(\Delta)$ [10] · g [9]
Partial set cover problem (PSCP)	$u_{ij} \in \{0, 1\}$, $d_i = 1$	· f [1, 5] · $\frac{f\Delta}{f+\Delta-1}$ [4]

CIP is a widely studied NP-hard problem since it includes fundamental combinatorial optimization problems such as the vertex cover problem, the set cover problem, or the minimum knapsack problem. There are some approximation algorithms for CIP, see Table 1 and Koufogiannakis and Young [7].

The partial set multi-cover problem (PSMCP) is a special case of PCIP where $u_{ij} \in \{0, 1\}$ and d_i is a positive integer for $i \in M$ and $j \in N$. There are a lot of applications of PSMCP such as analysis of influence in social networks [9, 10] and protein identification [8]. Ran et al. [10] give an approximation algorithm with performance ratio $\gamma H(\Delta)$ under the assumption that $m - p > (1 - \frac{1}{\eta})m$ and $c_j > 0$ ($j \in N$). Ran et al. [9] propose an approximation algorithm with performance ratio g defined in (3).

The partial set cover problem (PSCP) is a special case of PSMCP where $d_i = 1$ for $i \in M$. Some approximation algorithms for PSCP are known as shown in Table 1.

Contribution

We present an α-approximation algorithm for PCIP, where

$$\alpha = \max\{f, p + 1\}. \tag{4}$$

Our algorithm is based on an f-approximation algorithm for PSCP by Gandhi et al. [5]. Their algorithm uses a primal-dual method as a subroutine. In our algorithm, we use a primal-dual algorithm based on Carnes and Shmoys [2] for the minimum knapsack problem and its extension to CIP by Takazawa and Mizuno [11].

Ran et al. [9] raised a question of whether an f-approximation algorithm for PSMCP exists or not. Note that such an algorithm exists for CIP and PSCP as in Table 1. Our algorithm achieves the performance ratio f when $f \geq p + 1$, and therefore we partially answer this question.

Assumption and Notation

Without loss of generality, we assume that

- (2) is feasible, and therefore it has an optimal solution,
- $c_1 \leq \cdots \leq c_n$,
- $d_i \geq u_{ij}$ ($i \in M,\ j \in N$),
- $f \geq 2$.

Let $I = (m, n, \boldsymbol{U}, \boldsymbol{d}, \boldsymbol{c}, p)$ be a data of (2), where $\boldsymbol{U}$ is the matrix of u_{ij}. We call I an instance of PCIP. Let PCIP(I) be the problem for instance I and $OPT(I)$

be the optimal value of PCIP(I). For any subset $S \subseteq N$, we define the solution $(\boldsymbol{x}(S), \boldsymbol{t}(S))$ as follows:

$$x_j(S) = \begin{cases} 1 & \text{if } j \in S \\ 0 & \text{if } j \notin S \end{cases} \text{ for any } j \in N \tag{5}$$

and

$$t_i(S) = \begin{cases} 1 & \text{if } \sum_{j \in S} u_{ij} < d_i \\ 0 & \text{if } \sum_{j \in S} u_{ij} \geq d_i. \end{cases} \text{ for any } i \in M. \tag{6}$$

This solution always satisfies the constraints in (2) except for $\sum_{i \in M} t_i \leq p$. Hence $(\boldsymbol{x}(S), \boldsymbol{t}(S))$ is feasible to (2) if and only if $\sum_{i \in M} t_i(S) \leq p$.

2 Main algorithm

Our algorithm is an extension of an f-approximation algorithm for PSCP by Gandhi et al. [5] and consists of two algorithms: the main algorithm and the subalgorithm. The subalgorithm is presented in Section 3. This section is organized as follows:

1. We show a property (Lemma 1) of the solution generated by the subalgorithm.
2. We explain that we can get an α-approximation solution by using the subalgorithm if we know partial information about an optimal solution.
3. We introduce the main algorithm which gives an α-approximation without information about an optimal solution.

For any problem PCIP(I), the subalgorithm checks whether it is feasible or not. If it is feasible, then the algorithm outputs $\tilde{S} \subseteq N$ such that $(\boldsymbol{x}(\tilde{S}), \boldsymbol{t}(\tilde{S}))$ is feasible and has the following property in Lemma 1. The algorithm and the proof of Lemma 1 are shown in Section 3.

Lemma 1. *The subalgorithm presented in Section 3 outputs $\tilde{S} \subseteq N$ such that the solution $(\boldsymbol{x}(\tilde{S}), \boldsymbol{t}(\tilde{S}))$ defined by (5) and (6) is feasible to PCIP(I) and satisfies*

$$\sum_{j \in N} c_j x_j(\tilde{S}) \leq \alpha OPT(I) + c_n.$$

The running time of the subalgorithm is $O(mn^2)$.

For an instance $I = (m, n, \boldsymbol{U}, \boldsymbol{d}, \boldsymbol{c}, p)$ and $h \in \{2, \ldots, n\}$, we consider a sub-problem of PCIP(I), where we add the following constraints to PCIP(I):

$$\begin{array}{ll} x_j = 0 & \text{if } j \geq h+1, \\ x_j = 1 & \text{if } j = h. \end{array}$$

This sub-problem can be expressed as:

$$\begin{array}{lll} \min & \displaystyle\sum_{j \in \{1,\ldots,h-1\}} c_j x_j & \\ \text{s.t.} & \displaystyle\sum_{j \in \{1,\ldots,h-1\}} u_{ij} x_j + d_i t_i \geq d_i - u_{ih}, & \forall i \in M = \{1, \ldots, m\}, \\ & \displaystyle\sum_{i \in M} t_i \leq p, & \\ & x_j \in \{0, 1\}, & \forall j \in \{1, \ldots, h-1\}, \\ & t_i \in \{0, 1\}, & \forall i \in M. \end{array} \tag{7}$$

Hence the instance of this sub-problem can be expressed as follows:

$$I(h) = (m, h-1, \boldsymbol{U}(h), \boldsymbol{d}(h), \boldsymbol{c}(h), p), \tag{8}$$

where

$$\begin{array}{ll} \boldsymbol{U}(h) & = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{h-1}), \\ \boldsymbol{d}(h) & = \boldsymbol{d} - \boldsymbol{u}_h = (d_1 - u_{1h}, \ldots, d_m - u_{mh})^T, \\ \boldsymbol{c}(h) & = (c_1, \ldots, c_{h-1})^T. \end{array}$$

Let S^* be the subset of N such that $(\boldsymbol{x}(S^*), \boldsymbol{t}(S^*))$ is an optimal solution of PCIP(I). From the assumption that $d_i > 0$ for all $i \in M$, there is an index $j \in N$ such that $x_j(S^*) = 1$. Define

$$h^* = \max\{j \in N \mid x_j(S^*) = 1\}.$$

When $h^* = 1$, an optimal solution is $(\boldsymbol{x}(\{h^*\}), \boldsymbol{t}(\{h^*\}))$. In the following discussion, we assume $h^* \geq 2$.

We can get an α-approximation solution for PCIP(I) by using the subalgorithm if we know h^*.

Lemma 2. *Let $\tilde{S}(h)$ be the output by the subalgorithm for the sub-problem PCIP($I(h)$) which is defined by (7). Define $S(h) = \tilde{S}(h) \cup \{h\}$. If $h = h^*$, $S(h^*)$ gives a feasible α-approxiamtion solution for PCIP(I), that is, $(\boldsymbol{x}(S(h^*)), \boldsymbol{t}(S(h^*)))$ is feasible to PCIP(I) and the following inequality holds:*

$$\sum_{j \in N} c_j x_j(S(h^*)) \leq \alpha OPT(I).$$

Proof. $(\boldsymbol{x}(S(h^*)), \boldsymbol{t}(S(h^*)))$ is feasible to PCIP(I) since $(\boldsymbol{x}(\tilde{S}(h^*)), \boldsymbol{t}(\tilde{S}(h^*)))$ is feasible to PCIP($I(h^*)$) from Lemma 1.

We have that $\alpha \geq 2$ since $\alpha = \max\{f, p+1\}$ and $f \geq 2$. From Lemma 1, $c_{h^*} \geq c_{h^*-1}$ and $\alpha \geq 2$, we have that

$$\begin{aligned}\sum_{j \in N} c_j x_j(S(h^*)) &= \left(\sum_{j \in \tilde{S}(h^*)} c_j\right) + c_{h^*} \\ &\leq \alpha OPT(I(h^*)) + c_{h^*-1} + c_{h^*} \\ &\leq \alpha(OPT(I(h^*)) + c_{h^*}) \\ &= \alpha OPT(I).\end{aligned}$$

□

Even though Lemma 2 requires the information about h^*, we don't need it in advance if we execute the subalgorithm for all PCIP($I(h)$) ($h \in \{2, \ldots, n\}$). The main algorithm is presented as follows:

The main algorithm

Input: $I = (m, n, \boldsymbol{U}, \boldsymbol{d}, \boldsymbol{c}, p)$.

Step 1: For $h \in \{2, \ldots, n\}$, set $S(h) = \emptyset$ and $COST(h) = +\infty$ and do the following process: Let $I(h)$ be the data defined by (8). Execute the subalgorithm for PCIP($I(h)$). If the problem is feasible, the algorithm outputs $\tilde{S}(h) \subseteq \{1, \ldots, h-1\}$. In this case, set $S(h) = \tilde{S}(h) \cup \{h\}$ and $COST(h) = \sum_{j \in N} c_j x_j(S(h))$.

Step 2: Set $\hat{h} = \arg\min_{h \in N} COST(h)$ and output $(\boldsymbol{x}(S(\hat{h})), \boldsymbol{t}(S(\hat{h})))$

Theorem 1. *The main algorithm is an α-approximation algorithm for PCIP.*

Proof. The running time of the algorithm is $O(mn^3)$ since the subalgorithm runs in $O(mn^2)$ from Lemma 1 and the main algorithm executes the subalgorithm at most n times. Therefore the main algorithm is a polynomial time algorithm.

$(\boldsymbol{x}(S(\hat{h})), \boldsymbol{t}(S(\hat{h})))$ is clearly feasible to PCIP(I) and from Lemma 2 we obtain that

$$\sum_{j \in N} c_j x_j(S(\hat{h})) \leq \sum_{j \in N} c_j x_j(S(h^*)) \leq \alpha OPT(I).$$

□

3 Subalgorithm

In this section, we show the subalgorithm and prove Lemma 1. The subalgorithm is based on a 2-approximation algorithm for the minimum knapsack problem by Carnes and Shmoys [2] and its extension to CIP by Takazawa and Mizuno [11]. Both of the algorithms use an LP relaxation of CIP proposed by Carr et al. [3]. We apply this relaxation to PCIP and we have the following problem:

$$\begin{array}{lll} \min & \sum_{j\in N} c_j x_j & \\ \text{s.t.} & \sum_{j\in N\setminus A} u_{ij}(A)x_j + d_i(A)t_i \geq d_i(A), & \forall i\in M, \forall A\subseteq N \\ & \sum_{i\in M} t_i \leq p, & \\ & x_j \geq 0, & \forall j\in N, \\ & t_i \geq 0, & \forall i \in M, \end{array} \tag{9}$$

where

$$\begin{array}{rcl} d_i(A) & = & \max\{0, d_i - \sum_{j\in A} u_{ij}\},\ \forall i\in M, \forall A\subseteq N, \\ u_{ij}(A) & = & \min\{u_{ij}, d_i(A)\},\ \forall i\in M, \forall A\subseteq N, \forall j\in N\setminus A. \end{array} \tag{10}$$

Lemma 3. *(9) is a relaxation problem of PCIP, that is, any feasible solution* $(\boldsymbol{x},\boldsymbol{t})$ *for PCIP is feasible to (9).*

Proof. Let $(\boldsymbol{x},\boldsymbol{t})$ be a feasible solution for PCIP. Let $S = \{j\in N \mid x_j = 1\}$ and $M_0 = \{i\in M \mid t_i = 0\}$. Note that $\sum_{j\in S} u_{ij} \geq d_i$ holds for all $i\in M_0$. It suffices to show that for all $i\in M_0$ and $A\subseteq N$,

$$\sum_{j\in S\setminus A} u_{ij}(A) \geq d_i(A). \tag{11}$$

Fix $i\in M_0$ and $A\subseteq N$ such that $d_i(A) > 0$, that is, $d_i(A) = d_i - \sum_{j\in A} u_{ij}$. We consider the case when there is an index $j\in S\setminus A$ such that $u_{ij}(A) = d_i(A)$. In this case, (11) is clearly satisfied. Next, we consider the case when $u_{ij}(A) < d_i(A)$, that is, $u_{ij}(A) = u_{ij}$ for all $j\in S\setminus A$. In this case, we have that

$$-d_i(A) + \sum_{j\in S\setminus A} u_{ij}(A) = -d_i + \sum_{j\in A} u_{ij} + \sum_{j\in S\setminus A} u_{ij} \geq -d_i + \sum_{j\in S} u_{ij} \geq 0.$$

□

The dual of (9) is expressed as

$$
\begin{array}{lll}
\max & \sum_{i\in M}\sum_{A\subseteq N} d_i(A)y_i(A) - pz & \\
\text{s.t.} & \sum_{i\in M}\sum_{A\subseteq N:\, j\notin A} u_{ij}(A)y_i(A) \le c_j, & \forall j \in N, \\
& \sum_{A\subseteq N} d_i(A)y_i(A) \le z, & \forall i \in M, \\
& y_i(A) \ge 0, & \forall A \subseteq N,\ \forall i \in M, \\
& z \ge 0.
\end{array}
\tag{12}
$$

Now, we introduce a useful result for later discussion.

Lemma 4. *Let S be a subset of N such that $(\boldsymbol{x}(S), \boldsymbol{t}(S))$ is infeasible to PCIP(I), $(\boldsymbol{y}, z)$ be a feasible solution to (12). Define $M_1(S) = \{i \in M \mid t_i(S) = 1\}$. If*

(a-1) $\forall j \in N,\ x_j(S) = 1 \Rightarrow \sum_{i\in M}\sum_{A\subseteq N:\, j\notin A} u_{ij}(A)y_i(A) = c_j$,
(a-2) $i \in M_1(S) \Rightarrow \sum_{A\subseteq N} d_i(A)y_i(A) = z$,
(b) $\forall i \in M_1(S),\ \forall A \subseteq N,\ y_i(A) > 0 \Rightarrow \sum_{j\in S\setminus A} u_{ij}(A) \le d_i(A)$,

then the following inequalities hold:

$$
\sum_{j\in N} c_j x_j(S) \le \alpha\left(\sum_{i\in M}\sum_{A\subseteq N} d_i(A)y_i(A) - pz\right) \le \alpha OPT(I). \tag{13}
$$

Proof. For any $A \subseteq N$ and $i \in M$, we have that

$$
\sum_{j\in S\setminus A} u_{ij}(A) \le \sum_{j\in S} u_{ij}(A) \le f d_i(A) \le \alpha d_i(A) \tag{14}
$$

from the definition of f and $u_{ij}(A) \le d_i(A)$ by (10). Since $(\boldsymbol{x}(S), \boldsymbol{t}(S))$ is infeasible, the following inequality holds:

$$
|M_1(S)| \ge p + 1. \tag{15}
$$

From (a-1), the objective function value of $(\boldsymbol{x}(S), \boldsymbol{t}(S))$ is

$$
\sum_{j\in N} c_j x_j(S) = \sum_{j\in S} c_j = \sum_{j\in S}\sum_{i\in M}\sum_{A\subseteq N:\, j\notin A} u_{ij}(A)y_i(A). \tag{16}
$$

Define $M_0(S) = M \backslash M_1(S)$ and we obtain that

$$
\begin{aligned}
& \sum_{j \in S} \sum_{i \in M} \sum_{A \subseteq N : j \notin A} u_{ij}(A) y_i(A) \\
= & \sum_{i \in M} \sum_{A \subseteq N} \sum_{j \in S \backslash A} u_{ij}(A) y_i(A) \\
= & \sum_{i \in M_0(S)} \sum_{A \subseteq N} \sum_{j \in S \backslash A} u_{ij}(A) y_i(A) + \sum_{i \in M_1(S)} \sum_{A \subseteq N} \sum_{j \in S \backslash A} u_{ij}(A) y_i(A) \\
= & \sum_{i \in M_0(S)} \sum_{A \subseteq N} y_i(A) \sum_{j \in S \backslash A} u_{ij}(A) + \sum_{i \in M_1(S)} \sum_{A \subseteq N} y_i(A) \sum_{j \in S \backslash A} u_{ij}(A) \\
\leq & \ \alpha \sum_{i \in M_0(S)} \sum_{A \subseteq N} d_i(A) y_i(A) + \sum_{i \in M_1(S)} \sum_{A \subseteq N} d_i(A) y_i(A),
\end{aligned}
$$

where the last inequality holds from (14) and (b). Hence we have that

$$
\sum_{j \in N} c_j x_j(S) \leq \alpha \sum_{i \in M_0(S)} \sum_{A \subseteq N} d_i(A) y_i(A) + \sum_{i \in M_1(S)} \sum_{A \subseteq N} d_i(A) y_i(A).
$$

Taking the difference between two values in (13),

$$
\begin{aligned}
& \alpha \left(\sum_{i \in M} \sum_{A \subseteq N} d_i(A) y_i(A) - pz \right) - \sum_{j \in N} c_j x_j(S) \\
\geq & \ (\alpha - 1) \sum_{i \in M_1(S)} \sum_{A \subseteq N} d_i(A) y_i(A) - \alpha pz \\
= & \ (\alpha - 1) |M_1(S)| z - \alpha pz \qquad (17) \\
\geq & \ (\alpha - (p+1)) z \geq 0, \qquad (18)
\end{aligned}
$$

where the equality (17) follows from (a-2) and inequalities (18) follow from (15) and (4). Since $(\boldsymbol{y}, z)$ is feasible to (12), the objective value of $(\boldsymbol{y}, z)$ is less than or equal to the optimal value of (9), which is less than or equal to OPT(I). Thus we have that

$$
\alpha \left(\sum_{i \in M} \sum_{A \subseteq N} d_i(A) y_i(A) - pz \right) \leq \alpha OPT(I). \qquad (19)
$$

□

The subalgorithm is presented below. Solutions generated by the algorithm, except for the final solution, satisfy all the conditions in Lemma 4. In the subalgorithm, we use the following symbols:

- a set $S \subseteq N$.
- a solution $(\boldsymbol{y}, z)$ for (12).

- $M_1(S) = \{i \in M | \sum_{j \in S} u_{ij} < d_i\}$.
- $N'(S) = \{j \in N \backslash S \,|\, \sum_{i \in M_1(S)} u_{ij}(S) > 0\}$.
- $\forall j \in N,\ \bar{c}_j = c_j - \sum_{i \in M} \sum_{A \subseteq N : j \notin A} u_{ij}(A) y_i(A)$.

The subalgorithm

Input: $I = (m, n, \boldsymbol{U}, \boldsymbol{d}, \boldsymbol{c}, p)$.

Step 0: Set $S = \emptyset$, $(\boldsymbol{y}, z) = (\boldsymbol{0}, 0)$ and $\bar{\boldsymbol{c}} = \boldsymbol{c}$. Check whether $(\boldsymbol{x}(N), \boldsymbol{t}(N))$ is feasible or not. If it is not feasible, declare INFEASIBLE and stop.

Step 1: Calculate $d_i(S)$ by (10) for $i \in M$. Update $M_1(S)$. If $|M_1(S)| \leq p$, output $\tilde{S} = S$ and stop. Otherwise, calculate $u_{ij}(S)$ by (10) for all $i \in M_1(S)$ and $j \in N$. Update $N'(S)$.

Step 2: For all $j \in N'(S)$, let

$$\delta_j = \frac{\bar{c}_j}{\sum_{i \in M_1(S)} (u_{ij}(S)/d_i(S))}.$$

For every $i \in M_1(S)$ simultaneously, increase $y_i(S)$ at the rate $1/d_i(S)$ as much as possible while maintaining $\sum_{i \in M_1(S)} u_{ij}(S) y_i(S) \leq \bar{c}_j$ for all $j \in N'(S)$. That is, set

$$y_i(S) = \frac{\delta_s}{d_i(S)},$$

where

$$s = \arg\min_{j \in N'(S)} \delta_j$$

for all $i \in M_1(S)$. Update $\bar{c}_j := \bar{c}_j - \sum_{i \in M_1(S)} u_{ij}(A) y_i(S)$ for all $j \in N'(S)$.

Now, let's see the increment of $\sum_{A \subseteq N} d_i(A) y_i(A)$ in this iteration, which is the left side of $\sum_{A \subseteq N} d_i(A) y_i(A) \leq z$ in (12). Note that $\sum_{A \subseteq N} d_i(A) y_i(A)$ is 0 for all $i \in M$ at the beginning of the algorithm. If $i \notin M_1(S)$, the increment is 0 since the algorithm doesn't increase $y_i(S)$ in this iteration. If $i \in M_1(S)$, the increment is $d_i(S) y_i(S)$ and we have that

$$d_i(S) y_i(S) = \delta_s.$$

We can say that for every $i \in M_1(S)$, the algorithm increase $\sum_{A \subseteq N} d_i(A) y_i(A)$ by the same amount by δ_s. On the other hand, it doesn't increase $\sum_{A \subseteq N} d_i(A) y_i(A)$ for any $i \notin M_1(S)$. Therefore, we have that for any $i_1,\ i_2 \in M_1(S)$,

$$\sum_{A \subseteq N} d_{i_1}(A) y_{i_1}(A) = \sum_{A \subseteq N} d_{i_2}(A) y_{i_2}(A)$$

and for any $i \in M_1(S)$ and $i' \notin M_1(S)$,

$$\sum_{A \subseteq N} d_{i'}(A) y_{i'}(A) \leq \sum_{A \subseteq N} d_i(A) y_i(A)$$

Update

$$z := z + \delta_s.$$

By updating z as above, for any $i \in M_1(S)$ and $i' \notin M_1(S)$ the algorithm maintains

$$\sum_{A \subseteq N} d_{i'}(A) y_{i'}(A) \leq \sum_{A \subseteq N} d_i(A) y_i(A) = z.$$

Update $S := S \cup \{s\}$ and go back to Step 1.

We show that solutions generated by the subalgorithm, except for the final solution, satisfy all the conditions in Lemma 4.

Lemma 5. *Let $\tilde{S}$ be the output by the subalgorithm and $\ell \in N$ be the index added to S at the last iteration by the subalgorithm. Let $(\boldsymbol{y}, z)$ be the dual variable at the end of the iteration before ℓ is added. $\tilde{S} \setminus \{\ell\}$ and $(\boldsymbol{y}, z)$ satisfy the conditions in Lemma 4.*

Proof. Define $S' = \tilde{S} \setminus \{\ell\}$.

feasibility: The subalgorithm stops if $|M_i(S)| \leq p$ which implies that $(\boldsymbol{x}(S), \boldsymbol{t}(S))$ is a feasible solution. Thus, $(\boldsymbol{x}(S'), \boldsymbol{t}(S'))$ is infeasible to PCIP(I). $(\boldsymbol{y}, z)$ is feasible to the dual (12) since the subalgorithm starts from the dual feasible solution $(\boldsymbol{y}, z) = (\boldsymbol{0}, 0)$ and maintains dual feasibility at every iteration.

(a-1) and (a-2): (a-1) and (a-2) are satisfied by the way the algorithm updates S and z, respectively.

(b): From Step 2, $y_i(A) > 0$ implies

$$A \subseteq S'.$$

Also, $i \in M_1(S)$ implies

$$\sum_{j \in S'} u_{ij} < d_i.$$

Thus, for all $i \in M_1(S')$ and $A \subseteq N$ such that $y_i(A) > 0$,

$$\sum_{j \in S' \setminus A} u_{ij}(A) \leq \sum_{j \in S' \setminus A} u_{ij} = \sum_{j \in S'} u_{ij} - \sum_{j \in A} u_{ij} < d_i - \sum_{j \in A} u_{ij} \leq d_i(A),$$

where the first and last inequalities follow from (10).

□

Now we can easily prove Lemma 1.

Proof of Lemma 1. $(\boldsymbol{x}(\tilde{S}), \boldsymbol{t}(\tilde{S}))$ is clearly feasible and from Lemma 4 and Lemma 5, we have that

$$\sum_{j \in N} c_j x_j(\tilde{S}) = c_\ell + \sum_{j \in N} c_j x_j(S') \leq \alpha OPT(I) + c_\ell \leq \alpha OPT(I) + c_n.$$

Let's take a look at the running time of the subalgorithm. In the algorithm, the most time consuming parts per iteration are the calculations of $u_{ij}(S)$ in Step 1 and δ_j in Step 2, which take $O(mn)$. Since the number of iterations is at most n, the total running time of the subalgorithm is $O(mn^2)$.

□

4 Conclusion

The partial covering 0–1 integer program (PCIP) is a generalization of the covering 0–1 integer program (CIP) and the partial set cover problem (PSCP). For PCIP, we proposed a $\max\{f, p+1\}$-approximation algorithm, where f is the largest number of non-zero coefficients in the constraints and p is the number of constrains which may not be satisfied. If $f \geq p + 1$, the performance ratio of our algorithm is f and it achieves the best performance ratio for CIP and PSCP. It is an open question whether an f-approximation algorithm exists without any assumption.

Acknowledgment

This research is supported in part by Grant-in-Aid for Science Research (A) 26242027 of Japan Society for the Promotion of Science and Grant-in-Aid for Young Scientist (B) 15K15941.

References

[1] R. Bar-Yahuda: Using homogeneous weights for approximating the partial cover problem, *in Proceedings 10th Annual ACM-SIAM Symposium on Discrete Algorithms* (1999), 71–75.

[2] T. Carnes and D. Shmoys: Primal-dual schema for capacitated covering problems, *Mathematical Programming*, **153** (2015), 289–308.

[3] R. D. Carr, L. Fleischer, V. J. Leung and C. A. Phillips: Strengthening integrality gaps for capacitated network design and covering problems, *Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms* (2000), 106–115.

[4] T. Fujito: On combinatorial approximation of covering 0–1 integer programs and partial set cover, *Journal of combinatorial optimization* **8.4** (2004): 439–452.

[5] R. Gandhi, S. Khuller and A. Srinivasan: Approximation algorithms for partial covering problems, *Journal of Algorithms* **53.1** (2004), 55–84.

[6] S. G. Kolliopoulos and N. E. Young: Tight approximation results for general covering integer programs, *in Proceedings 42nd Annual IEEE Symposium on Foundations of Computer Science* (2001), 522–528.

[7] C. Koufogiannakis and N. E. Young: Greedy δ-approximation algorithm for covering with arbitrary constraints and submodular cost, *Algorithmica*, **66** (2013), 113–152.

[8] Z. He, C. Yang and W. Yu: A partial set covering model for protein mixture identification using mass spectrometry data, *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, **8.2** (2011), 368–380.

[9] Y. Ran, Y. Shi and Z. Zhang: Local ratio method on partial set multicover, *Journal of Combinatorial Optimization*, **34.1** (2017), 302–313. doi:10.1007/s10878–016–0066–0

[10] Y. Ran, Z. Zhang, H. Du and Y. Zhu: Approximation algorithm for partial positive influence problem in social network, *Journal of Combinatorial Optimization*, **33.2** (2017), 791–802. doi:10.1007/s10878–016–0005–0

[11] Y. Takazawa and S. Mizuno: A 2-approximation algorithm for the minimum knapsack problem with a forcing graph, *Journal of Operations Research Society of Japan*, **60.1** (2017), 15–23.

Yotaro Takazawa
Department of Industrial Engineering and Management
Tokyo Institute of Technology
2–12–1 Ohokayama
Meguro-ku Tokyo 152–8552, Japan
E-mail: takazawa.y.ab@m.titech.ac.jp