論文 / 著書情報
Article / Book Information

題目(和文)	
Title(English)	LP-based Approximation Algorithms for General Covering Problems
著者(和文)	高澤陽太朗
Author(English)	Yotaro Takazawa
出典(和文)	学位:博士(工学), 学位授与機関:東京工業大学, 報告番号:甲第11277号, 授与年月日:2019年9月20日, 学位の種別:課程博士, 審査員:水野 眞治,松井 知己,宮川 雅巳,中田 和秀,塩浦 昭義
Citation(English)	Degree:Doctor (Engineering), Conferring organization: Tokyo Institute of Technology, Report number:甲第11277号, Conferred date:2019/9/20, Degree Type:Course doctor, Examiner:,,,,
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

THESIS

LP-based Approximation Algorithms for General Covering Problems

by

Yotaro Takazawa

Supervisor: **Prof. Shinji Mizuno**

Department of Industrial Engineering and Economics
Tokyo Institute of Technology

JULY 2019

Abstract

In covering problems, we find the best solution under certain covering constraints. Covering problems have many applications in operations research and other areas. Since a lot of interesting and important covering problems are NP-hard, approximation algorithms are well studied. For a minimization problem, an algorithm is said to be α-approximation if it efficiently returns a solution whose objective value is at most α times the optimal value.

Recently, more general covering problems are studied from practical requirements. Even though LP-based approximation algorithms are known to be particularly effective for simple covering problems, it is not easy to apply such algorithms to more general covering problems. In this thesis, we overcome this difficulty and develop better LP-based approximation algorithms than the existing ones for general covering problems. Our algorithms are mainly based on strong LP relaxations and an enumeration technique.

First, we develop a 2-approximation algorithm for a generalization of the vertex cover problem, where there is a minimum knapsack constraint. This problem is called the minimum knapsack problem with forcing constraints (MKPFC). Even though exact and heuristic algorithms are proposed for MKPFC, there is no constant approximation algorithm. Second, we give an improved approximation algorithm for the covering 0-1integer program (CIP), which is a natural generalization of the set cover problem. For CIP, there are some f-approximation algorithm where f is the maximum of numbers of non-zero coefficients of constraints. Our algorithm is the first algorithm with approximation ratio strictly less than f. Third, we present a $\max\{f, p+1\}$-approximation algorithm for the partial covering 0-1 integer program (PCIP), where at most p constraints in CIP can be violated. Finally, we give a simple $(p+1)$-approximation algorithm for the partial covering LP, which can be considered as an LP version of PCIP.

Contents

Chapter 1

Introduction

1.1 Backgrounds

In this section, we give backgrounds of our research. First, we explain some covering problems with examples. Second, we review approximation algorithms about covering problems.

1.1.1 Covering problems

In order to satisfy demands of people or facility, we often decide how to place supply services properly in the real world, e.g., location planning of distribution centers to satisfy demands of customers [15]. Such a decision making problem is modeled as an optimization problem with constraints to satisfy (cover) some demands [20]. These optimization problems are called covering problems.

One of the representative covering problems is a set covering problem (SCP). Many practical problems are modeled as SCP especially in industries such as airlines, railroads, hospitals and manufacturing. Main application areas of SCP are flight or railway scheduling, facility location and vehicle rooting. Before we show the setting of SCP, we give an examples as applications of SCP.

Example 1. Factory planning

Consider a company which produces some items and a manager who is planning which factories to use in order to produce the items. We are given a set of items, each of which require different materials or technologies. We are also given a set of candidate factories to use, each of which has a subset of the items set that can be produced. Every factory has a cost to use. The objective of this planning

problem is to find a set of factories with the minimum cost such that every item is produced by at least one factory.

Now, we will model the above problem to SCP. In SCP, we are given a ground set $M = \{1, \ldots, m\}$ and subsets $S_j \subseteq M$ and costs c_j for $j \in N = \{1, \ldots n\}$. The objective of SCP is to find a subset $X \subseteq N$ of the minimum cost such that any element of the ground set M is included in at least one S_j ($j \in X$), that is, $\cup_{j \in X} S_j = M$. The factory planning problem is modeled as SCP as follows:

- M =(a set of the items)
- N =(a set of the candidate factories)
- $S_j = \{i \in M \mid \text{factory } j \in N \text{ can produce item } i\}$
- c_j = (a cost of factory $j \in N$).

While SCP has applications as we see in the above example, the setting of SCP is very simple. In order to apply practical problems, generalizations of SCP are well studied. In this thesis, we study two types of general covering problems:

1. the covering 0-1 integer program, which is a natural generalization of SCP,
2. partial covering problems, where we do not need to satisfy all the covering constraints.

Covering 0-1 integer program (CIP)

First, we deal with CIP, which can model more general situation as follows.

Example 2. General factory planning

In Example 1, we see the factory planning problem as an example of applications of SCP. Now, we can consider more general situation as follows. We are given a set of the items, each of which has a positive demand. Also, we are given a set of the candidate factories to use, each of which has a nonnegative amount that can be produced for every item. Every factory has a cost to use. The objective of this planning problem is to find a set of factories with the minimum cost such that a demand is satisfied for every item.

SCP is no longer able to model a problem of Example 2. CIP is a natural generalization of SCP and can model more general problems including the above

factory planning problem. Generally, CIP is formulated as follows:

$$\text{CIP} \left| \begin{array}{lll} \min & \displaystyle\sum_{j \in N} c_j x_j & \\ \text{s.t.} & \displaystyle\sum_{j \in N} a_{ij} x_j \geq b_i, & \forall i \in M = \{1, \cdots, m\}, \\ & x_j \in \{0, 1\}, & \forall j \in N = \{1, \cdots, n\}, \end{array} \right. \tag{1.1}$$

where b_i, a_{ij}, and c_j ($i \in M,\ j \in N$) are nonnegative. Note that SCP is a special case of CIP where $a_{ij} \in \{0, 1\}$ and $b_i = 1$ ($i \in M,\ j \in N$). The above factory planning problem is modeled as CIP as follows:

- M =(a set of the items)
- N =(a set of the candidate factories)
- a_{ij} = (a amount of item i that factory $j \in N$ can produce)
- b_i = (a demand of item $i \in M$)
- c_j = (a cost of factory $j \in N$).

CIP also has other applications such as location problems [15]. In addition to that, CIP indirectly appears as a subproblem in other applications, e.g. transportation problems [3, 2] and cutting stock problems [24]. For later discussion, we define an important value f as the maximum number of nonzero coefficients of the constraints, that is,

$$f = \max_{i \in M} |\{j \in N \mid a_{ij} > 0\}|. \tag{1.2}$$

Next, we see the following special case of Example 2.

Example 3. Special factory planning

As a special case of Example 2, we consider the following situation. We are given a set of the items, each of which has a positive demand and a set of the candidate factories to use. Also, we are given a set of the candidate factories to use, each of which has a nonnegative amount that can be produced for every item. However, specified one item can be produced by any factories. For any other item, it has a a pair of factories and can only be produced by this pair.

The above problem is modeled as a special case of CIP called the minimum knapsack problem with forcing constraints (MKPFC). Given pairs $E \subseteq N \times N$, MKPFC is formulated as follows:

$$\text{MKPFC} \left| \begin{array}{lll} \min & \displaystyle\sum_{j \in N} c_j x_j & \\ \text{s.t.} & \displaystyle\sum_{j \in N} a_j x_j \geq b, & \\ & x_i + x_j \geq 1, & \forall (i, j) \in E, \\ & x_j \in \{0, 1\}, & \forall j \in N, \end{array} \right.$$

where all the input are nonnegative. Since a problem of Example 3 is easily modeled as MKPFC, we omit the details about modeling. While MKPFC is a special case of CIP, it is a generalization of well known two combinatorial optimization problems; the minimum knapsack problem and the vertex cover problem.

Partial covering problems

In covering problems such as SCP and CIP, all the covering constraints must be satisfied. On the other hand, for some practical problems, it is sufficient to satisfy a certain number or rate of constraints.

Example 4. Factory planning with outliers

In Example 2, we see the general factory planning problem as an example of applications of CIP, where we find a minimum cost factories to use such that a demand for every item is satisfied. In this example, it is sometimes unreasonable to satisfy demands of every item because of items that cost too much compared with the other items. For example, an item needs several factories but these factories can not produce any other items. When there are such items with high cost, the cost of factories needed to satisfy demands of all the items would be much larger than the cost needed to satisfy demands of 90 percent of items. Thus, the objective of this planning problem is to find a set of factories with the minimum cost such that a demand is satisfied for 90 percent of the item.

As this example shows, a shortcoming of the model of the covering problems is that certain elements which is hard to cover can make an optimal solution too expensive. Such elements are known as outliers. From this context, a model where we do not need to satisfy a specified number of constraints, is studied in not only covering problems [23, 42] but also other optimization problems such as linear programming [11, 45], facility location [12], clustering [13]. Especially, a

covering problem, where at most p covering constraints are not satisfied for given integer p, is called a partial covering problem.

In this thesis, we focus on two kinds of partial covering problems. The first one is the partial covering 0-1 integer program (PCIP), where at most given p of the inequality constraints in CIP can be violated. In addition to the same way we model the problem of Example 3 to CIP, a problem of Example 4 is modeled as PCIP by setting $p = \lfloor 0.1m \rfloor$. PCIP is a generalization of not only CIP but also other partial covering problems such as the partial set covering problem and the partial set multi-covering problem. PCIP has applications such as facility location [29] and influence problems in social network [42, 43].

The second one is the partial covering LP (PCLP) which is a partial version of the covering LP. The covering LP is a special case of LP and formulated as $\{\min \boldsymbol{c}^T \boldsymbol{x} \mid \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b}, \boldsymbol{x} \geq \boldsymbol{0}\}$, where $\boldsymbol{A} \in \mathbb{R}^{m \times n}, \boldsymbol{b} \in \mathbb{R}^m$, $\boldsymbol{c} \in \mathbb{R}^n$ and all the data are nonnegative. The covering LP often appears in practice and thus efficient algorithms are well studied, see [4] and references therein. As a partial version of the covering LP, PCLP is studied by Qiu et al. [41] for an application to portfolio optimization.

1.1.2 Approximation algorithms

Approximation algorithms

Many of combinatorial optimization problems including SCP are NP-hard, where there are no polynomial time algorithms to find optimal solutions to such problems unless P=NP. Note that a polynomial time algorithm is the one that runs in time bounded by a polynomial in its input size. In fact, all the covering problems shown above are NP-hard. There are algorithms to deal with NP-hard problems and we can classify these algorithms into two classes.

The first class consists of algorithms that find the optimal solution but do not run in polynomial time. An example of such an algorithm is based on integer programming. In the field of integer programming, researchers study branch-and-cut algorithms which are efficient for particular problems but they are not guaranteed to be efficient for all instances. Branch-and-cut algorithms are well studied for covering problems [3, 5].

The second class consists of algorithms that runs in polynomial time but do not find the optimal solution for all instances. Examples of such an algorithm are heuristics and metaheuristics. In the field of heuristics or metaheuristics, the efficiency of an algorithm is experimentally evaluated. Survey and experimental

results of several heuristic algorithms for SCP are shown in [8, 50]. Note that heuristics do not always output a good solution. In this thesis, we consider another kind of algorithms, called approximation algorithms, in this second class. Approximation algorithms run in polynomial time and output a solution whose objective value is theoretically guaranteed to be close to the optimum value.

For a given maximization or minimization problem, a polynomial time algorithm is said to be α-approximation when it outputs a solution whose objective value is within a factor of α of the optimum value. The parameter α is called the approximation ratio or the approximation guarantee of the algorithm. Note that $\alpha \geq 1$ for minimization problems and $\alpha \leq 1$ for maximization problems. For example, a 2-approximation algorithm for a minimization problem is a polynomial time algorithm that always returns a solution whose objective values is at most twice the optimal values.

Approximation algorithms for covering problems

Approximation algorithms and inapproximability for SCP are well studied. There are well-known two kinds of approximation algorithms for SCP. First one is a greedy algorithm, which selects the most cost-effective set among unselected sets at one iteration. This algorithm is known to be $O(\log m)$-approximation. It is also known that there is no $o(\log m)$-approximation algorithms for SCP [44] unless P=NP, which means the greedy algorithm achieves the best approximation ratio.

Second one is an approximation algorithm based on linear programming (LP), which is focused on in this thesis. LP plays an important role in the design and analysis of approximation algorithms and there are a lot of approximation algorithms based on LP. Such algorithms are roughly divided into two type. The first is a rounding algorithm that approximates a solution by rounding the solution of an LP relaxation of a original problem. The second is the primal-dual algorithm that finds an approximation solution by constructing solutions satisfying the complementary slackness conditions without solving an LP relaxation. LP-based approximation algorithms are particularly effective for SCP and a simple rounding algorithm gives f-approximation, where f is the frequency of SCP defined by $\max_{i \in M} |\{j \in N | i \in S_j\}|$. Note that f is also defined by (1.2) in CIP. SCP is hard to approximate better than a factor of $f - 1 - \epsilon$ for any positive ϵ less P=NP [17].

Approximation algorithms for general covering problems are also studied. When a problem is a generalization of SCP, it has at least the same inapproximability as SCP. Thus, one of the research objectives in approximation algorithms for a generalization of SCP is to develop an approximation algorithm with ap-

proximation ratio close to f or $O(\log m)$ if it is possible. In fact, there are approximation algorithms for CIP with approximation ratios f [10, 21, 22] and $O(\log m)$ [34]. An f-approximation algorithm is very effective when all the constraints are sparse. However, this algorithm has a shortcoming that the approximation ratio gets worse if one constraint is dense in CIP even though the other constraints are sparse, e.g. MKPFC.

While no approximation algorithms for PCIP are studied, there are good approximation algorithms for a special case for PCIP. A partial set covering problem (PSCP) is a special case of PCIP and a partial version of SCP, where at most p elements can not be covered. For PSCP, there are approximation algorithms with approximation ratios f [23] and $O(\log m)$ [19]. Such algorithms are not presented for PCIP which is a generalization of PSCP and CIP. Recently. some approximation algorithms are presented for the partial set multi-covering problem [43, 42], which is a special case of PCIP but a generalization of PSCP. However, their approximation ratios have a gap with the best approximation ratios for SCP since they depend on many coefficients. For PCLP, even though $O(f \log f)$ and $O(\log \Delta \log f)$ approximation algorithms are proposed [16], there are no results inapproximability.

1.2 Objective of thesis

The objective of this thesis is to develop better LP-based approximation algorithms than the existing algorithms for two types of general and important covering problems: CIP and partial covering problems (PCIP and PCLP). It is not easy to develop good LP-based approximation algorithms for general covering problems since an optimal solution of a natural LP relaxation is sometimes far from an optimal solution of the original problem, e.g., see [9]. Thus, in order to develop LP-based approximation algorithms for such problems, strong LP relaxation or other technique are needed. For our purpose, we carefully design approximation algorithms by using two techniques; (1) strong LP relaxations of CIP by Carr et al. [10] and Fujito [21] and (2) an enumeration technique for partial covering problems by Gandhi et al. [23].

1.3 Thesis overview

In Chapter 2, we develop a constant factor approximation algorithm for MKPFC. In Chapter 3, we present an approximation algorithm for CIP whose approximation ratio is better than that of existing algorithms. In Chapter 4, we extend the algorithm in Chapter 3 to PCIP. In Chapter 5, we design a simple algorithm based on LP rounding for CPVLP. In Chapter 6, we conclude this thesis. Note that details of previous research about approximation algorithms for each problem are shown in each chapter. The main results are summarized as follows.

Chapter 2: 2-approximation algorithm for MKPFC

In Chapter 2, we study approximation algorithms for MKPFC, which is known to be a special case of CIP. This problem has only one dense constraint constraint and many sparse constraints. This is an unfavorable case for an existing f-approximation algorithm for CIP since it only gives n-approximation, where n is the number of variables. In this chapter, we develop a primal-dual 2-approximation algorithm by extending an algorithm for the minimum knapsack problem by Carnes and Shmoys [9]. In addition to that, we generalize our algorithm to CIP and propose an f_2-approximation algorithm, where f_2 is the second largest number of the non-zero coefficients in the constraints. The proposed algorithm overcomes the shortcoming of the f-approximation algorithm that the approximation ratio gets worse if one constraint is dense in CIP even though the other constraints are sparse. Part of the contents of this chapter is included in Takazawa et al. [46].

Chapter 3: Improved approximation algorithm for CIP

In Chapter 3, we propose an $\left(f - \frac{f-1}{m}\right)$-approximation algorithm for CIP with $m \geq 2$. No approximation algorithm with approximation ratio strictly less than f is proposed for CIP while such algorithms exist for special cases of CIP such as SCP. Our algorithm is the first algorithm for CIP with approximation ratio strictly less than f when $f \geq 2$. Our algorithm is based on a primal-dual f-approximation algorithm for CIP by Fujito [21] and an enumeration technique by Gandhi et al. [23]. Part of the contents of this chapter is included in Takazawa et al. [47].

Chapter 4: $\max\{f, p+1\}$-approximation algorithm for PCIP

In Chapter 4, we study approximation algorithms for PCIP which is a partial version of CIP. For some special cases of PCIP, approximation algorithms are pre-

sented. However their approximation ratios may get worse even when the size of problem is small. In this chapter, we propose a $\max\{f, p+1\}$-approximation algorithm for PCIP by generalizing our algorithm for CIP in Chapter 2. This approximation ratio achieves the same approximation ratio as f possibly best for SCP when $f \geq p+1$. Part of the contents of this chapter is included in Takazawa et al. [49].

Chapter 5: $(p+1)$-approximation algorithm for PCLP

In Chapter 5, we give a $(p+1)$-approximation algorithm for the PCLP. PCLP is a partial version of the covering LP. Our algorithm is a simple rounding algorithm based on a natural LP relaxation. We also show that we can not get better approximation algorithms when we use the LP-relaxation as a lower bound of the optimal value. Part of the contents of this chapter is included in Takazawa et al. [48].

Chapter 2

A 2-approximation algorithm for the minimum knapsack problem with forcing constraints

2.1 Overview

While the covering 0-1 integer program (CIP) is a natural generalization of the set covering problem (SCP), CIP can be also considered as a generalization of the minimum knapsack problem (MKP). MKP is a special case of CIP where there is only one inequality constraint, that is, $m = 1$ and it is formulated as follows:

$$\begin{aligned} \min \quad & \sum_{j \in N} c_j x_j \\ \text{s.t.} \quad & \sum_{j \in N} a_j x_j \geq b, \\ & x_j \in \{0, 1\}, \qquad \forall j \in V = \{1, \cdots, n\}, \end{aligned} \tag{2.1}$$

where $a_j,\ c_j \geq 0$ $(j \in N)$ and $b > 0$. MKP is one of the most fundamental problems in combinatorial optimization. In addition to the inequality constraint of MKP, other constraints often appear in practice. Thus, knapsack problems with additional constraints are well studied, see the survey of Wilbaut et al. [52].

One of additional constraints often appeared in practice is a forcing constraint, where, given a pair of binary variables, at least one variable of the pair must be set to 1. Given pairs of variables $E \subseteq V \times V$, the minimum knapsack problem with

forcing constraints is formulated as follows:

$$\text{MKPFC} \left| \begin{array}{lll} \min & \displaystyle\sum_{j \in V} c_j x_j & \\ \text{s.t.} & \displaystyle\sum_{j \in V} a_j x_j \geq b, & \\ & x_i + x_j \geq 1, & \forall (i, j) \in E, \\ & x_j \in \{0, 1\}, & \forall j \in V. \end{array} \right. \tag{2.2}$$

A graph $G = (V, E)$ is called a forcing graph. A forcing constraint is recently used not only for the knapsack problem [39] but also for other combinatorial optimization problems such as minimum spanning tree, matching and shortest path problems [14] and the maximum flow problem [38].

The problem MKPFC (2.2) includes the minimum weight vertex cover problem (VCP) as a special case. It is known that VCP is a strongly NP-hard problem and has inapproximability such that the problem is hard to approximate within any constant factor better than 1.36 unless $P = NP$ [17] and 2 under unique games conjecture [32]. It follows that MKPFC is strongly NP-hard and has at least the same inapproximability as VCP. Bar-Yehuda and Even [7] propose a 2-approximation algorithm for VCP.

The maximization version of MKPFC is known as the knapsack problem with disjunctive constraints (KPDC). KPDC is the maximum knapsack problem with disjunctive constraints for pairs of items which cannot be packed simultaneously in the knapsack. Exact and heuristic algorithms for KPDC are studied by [27, 28, 53] and approximation algorithms are proposed by [37, 39]. Any exact algorithm for KPDC can solve MKPFC since MKPFC can be transformed into KPDC by complementing the variables. However, the approach of converting MKPFC into KPDC cannot be used in general when we consider the performance guarantee of approximation algorithms.

For CIP, which is a generalization of MKPFC, there are approximation algorithms with approximation ratios f [10, 21, 22], where f is the largest number of the non-zero coefficients in the constraints. An f-approximation algorithm is very effective when all the constraints are sparse. However, this algorithm has a shortcoming that the approximation ratio gets worse if one constraint is dense in CIP even though the other constraints are sparse. Thus, MKPFC is an unfavorable special case of CIP for an existing f-approximation algorithm and it only gives n-approximation for MKPFC.

Contributions

In this chapter, we develop a primal-dual 2-approximation algorithm for MKPFC by extending an algorithm for MKP by Carnes and Shmoys [9]. Our approximation ratio is much better than n-approximation achieved by existing f-approximation algorithms for CIP. In addition to that, we generalize our algorithm to CIP and propose an f_2-approximation algorithm, where f_2 is the second largest number of the non-zero coefficients in the constraints. The proposed algorithm overcomes the shortcoming of the f-approximation algorithm for CIP that the approximation ratio gets worse if one constraint is dense even though the other constraints are sparse.

The main idea of our algorithm is as follows. First, we consider a strong LP relaxation of CIP by Carr et al. [10] and its dual problem. Then, by generalizing the primal-dual algorithm for MKP by Carnes and Shmoys [9], we design a new primal-dual algorithm for MKPFC. However, a natural generalization of the algorithm will give only f-approximation. Our algorithm overcomes this by carefully deciding the order in which constraints are satisfied in a primal-dual algorithm. To be more precise, the knapsack constraint (the most dense constraint) is the last constraint to be satisfied in our primal-dual algorithm.

2.2 Algorithm and analysis

Carnes and Shmoys [9] use the following LP relaxation of the minimum knapsack problem (2.1), which is presented by Carr et al. [10]:

$$\begin{aligned} \min \quad & \sum_{j \in V} c_j x_j & & \\ \text{s.t.} \quad & \sum_{j \in V \setminus A} a_j(A) x_j \geq b(A), & \forall A \subseteq V, & \\ & x_j \geq 0, & \forall j \in V, & \end{aligned} \tag{2.3}$$

where

$$\begin{aligned} b(A) &= \max\{0, b - \textstyle\sum_{j \in A} a_j\}, & \forall A \subseteq V, \\ a_j(A) &= \min\{a_j, b(A)\}, & \forall A \subseteq V, \forall j \in V \setminus A. \end{aligned} \tag{2.4}$$

It is known that any feasible 0-1 solution of (2.3) is feasible for (2.1).

Similarly, we use the following LP relaxation of MKPFC (2.2):

$$
\begin{array}{lll}
\min & \displaystyle\sum_{j\in V} c_j x_j & \\
\text{s.t.} & \displaystyle\sum_{j\in V\setminus A} a_j(A)x_j \geq b(A), & \forall A \subseteq V, \qquad (2.5.1) \\
& x_i + x_j \geq 1, & \forall \{i,j\} \in E, \quad (2.5.2) \\
& x_j \geq 0, & \forall j \in V.
\end{array}
\tag{2.5}
$$

The dual of (2.5) is represented as

$$
\begin{array}{lll}
\max & \displaystyle\sum_{A\subseteq V} b(A)y(A) + \sum_{\{i,j\}\in E} z_{\{i,j\}} & \\
\text{s.t.} & \displaystyle\sum_{A\subseteq V: j\notin A} a_j(A)y(A) + \sum_{k:\{j,k\}\in E} z_{\{j,k\}} \leq c_j, & \forall j \in V, \qquad (2.6.1) \\
& y(A) \geq 0, & \forall A \subseteq V, \\
& z_{\{i,j\}} \geq 0, & \forall \{i,j\} \in E,
\end{array}
\tag{2.6}
$$

where each dual variable $y(A)$ corresponds to the inequality $\sum_{j\in V\setminus A} a_j(A)x_j \geq b(A)$ and $z_{\{i,j\}}$ corresponds to the forcing constraint for the edge $\{i,j\}$.

Now we show a well-known result for a primal-dual pair of linear programming [18].

Lemma 2.2.1. *Let $\bar{\boldsymbol{x}}$ and $\bar{\boldsymbol{y}}$ be feasible solutions for the following primal and dual linear programming problems:*

$$
\min\left\{\boldsymbol{c}^T\boldsymbol{x} \mid \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b},\ \boldsymbol{x} \geq \boldsymbol{0}\right\} \quad \textit{and} \quad \max\left\{\boldsymbol{b}^T\boldsymbol{y} \mid \boldsymbol{A}^T\boldsymbol{y} \leq \boldsymbol{c},\ \boldsymbol{y} \geq \boldsymbol{0}\right\}.
$$

If, for $\alpha \geq 1$, the conditions

$$
\begin{array}{ll}
\textit{(a):} & \forall j \in \{1,\cdots,n\}, \bar{x}_j > 0 \Rightarrow \sum_{i=1}^m a_{ij}\bar{y}_i = c_j, \\
\textit{(b):} & \forall i \in \{1,\cdots,m\}, \bar{y}_i > 0 \Rightarrow \sum_{j=1}^n a_{ij}\bar{x}_j \leq \alpha b_i
\end{array}
$$

hold, then $\bar{\boldsymbol{x}}$ is a solution within a factor of α of the optimal solution, that is, the primal objective value $\boldsymbol{c}^T\bar{\boldsymbol{x}}$ is less than or equal to α times the optimal value. Note that the primal problem has an optimal solution because both the primal and dual problems are feasible.

By applying Lemma 2.2.1 to the problems (2.5) and (2.6), we have the following lemma and corollary.

Lemma 2.2.2. *Let $\boldsymbol{x}$ and $(\boldsymbol{y}, \boldsymbol{z})$ be feasible solutions for (2.5) and (2.6), respectively. If these solutions satisfy*

$$\begin{array}{ll} \textit{(a):} & \forall j \in V, x_j > 0 \Rightarrow \sum_{A \subseteq V : j \notin A} a_j(A) y(A) + \sum_{k:\{j,k\} \in E} z_{\{j,k\}} = c_j, \\ \textit{(b-1):} & \forall \{i, j\} \in E, z_{\{i,j\}} > 0 \Rightarrow x_i + x_j \leq 2, \\ \textit{(b-2):} & \forall A \subseteq V, y(A) > 0 \Rightarrow \sum_{j \in V \setminus A} a_j(A) x_j \leq 2b(A), \end{array} \tag{2.7}$$

then $\boldsymbol{x}$ is a solution within a factor of 2 of the optimal solution of (2.5).

Corollary 2.2.1. *Let $\boldsymbol{x}$ be a feasible 0-1 solution of (2.5) and $(\boldsymbol{y}, \boldsymbol{z})$ be a feasible solution of (2.6). If these solutions satisfy (2.7), $\boldsymbol{x}$ is a solution within a factor of 2 of the optimal solution of (2.2).*

We propose a polynomial algorithm for calculating $\boldsymbol{x}$ and $(\boldsymbol{y}, \boldsymbol{z})$ which satisfy the conditions in Corollary 2.2.1. The algorithm generates a sequence of points $\boldsymbol{x}$ and $(\boldsymbol{y}, \boldsymbol{z})$ which always satisfy the following conditions:

- $\boldsymbol{x} \in \{0, 1\}^n$.
- $(\boldsymbol{y}, \boldsymbol{z})$ is feasible for (2.6).
- $\boldsymbol{x}$ and $(\boldsymbol{y}, \boldsymbol{z})$ satisfy (2.7).

All the forcing constraints (2.5.2) are satisfied in Step 1 and the other constraints (2.5.1) are met in Step 2. For the points $\boldsymbol{x}$ and $(\boldsymbol{y}, \boldsymbol{z})$ at each step, we use symbols $S = \{j \in V \mid x_j = 1\}$, $\bar{b} = b - \sum_{j \in V} a_j x_j$, and $\bar{c}_j = c_j - (\sum_{A \subseteq V : j \notin A} a_j(A) y(A) + \sum_{k:\{j,k\} \in E} z_{\{j,k\}})$ for $j \in V$. $\bar{E} \subseteq E$ denotes a set of unchecked edges in Step 1. Now we state our algorithm.

Algorithm 1

Input: $G = (V, E)$, a_j, c_j $(j \in V)$ and b.

Output: $\tilde{\boldsymbol{x}}$ and $(\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{z}})$.

Step 0: Let $\boldsymbol{x} = \boldsymbol{0}$ and $(\boldsymbol{y}, \boldsymbol{z}) = (\boldsymbol{0}, \boldsymbol{0})$ be initial solutions. Set $S = \emptyset$, $V' = \{j \in V \mid a_j > 0\}$, $\bar{E} = E$, $\bar{b} = b$, and $\bar{c}_j = c_j$ for $j \in V$.

Step 1: If $\bar{E} = \emptyset$, then go to Step 2. Otherwise choose an edge $e = \{i, j\} \in \bar{E}$. If $x_i + x_j \geq 1$, then update $\bar{E} = \bar{E}\backslash\{e\}$ and go back to the top of Step 1. If $x_i + x_j = 0$, increase $z_{\{i,j\}}$ as much as possible while maintaining feasibility for (2.6.1). Since $z_{\{i,j\}}$ appears in only two constraints of (2.6.1) corresponding to the vertices i and j, we see that

$$z_{\{i,j\}} = \bar{c}_s \quad \text{for} \quad s = \arg\min\{\bar{c}_i,\ \bar{c}_j\}.$$

Update $x_s = 1, S = S \cup \{s\}, \bar{E} = \bar{E}\backslash\{e\}, \bar{c}_i = \bar{c}_i - z_{\{i,j\}}, \bar{c}_j = \bar{c}_j - z_{\{i,j\}}$, and $\bar{b} = \bar{b} - a_s$. Go back to the top of Step 1.

Step 2: If $\bar{b} \leq 0$, then output $\tilde{\boldsymbol{x}} = \boldsymbol{x}$ and $(\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{z}}) = (\boldsymbol{y}, \boldsymbol{z})$ and stop. Otherwise calculate $a_j(S)$ for all $j \in V'\backslash S$ by (2.4), where $b(S) = \bar{b}$. Increase $y(S)$ as much as possible while maintaining feasibility for (2.6.1). Since $y(S)$ appears in constraints of (2.6.1) for $j \in V'\backslash S$ so that $a_j(S) > 0$, we see that

$$y(S) = \frac{\bar{c}_s}{a_s(S)} \quad \text{for} \quad s = \arg\min_{j \in V'\backslash S} \left\{ \frac{\bar{c}_j}{a_j(S)} \right\}.$$

Update $x_s = 1,\ S = S \cup \{s\},\ \bar{c}_j = \bar{c}_j - a_j(S)y(S)$ for any $j \in V'\backslash S$, and $\bar{b} = \bar{b} - a_s$. Go back to the top of Step 2.

For the outputs $\tilde{\boldsymbol{x}}$ and $(\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{z}})$ of Algorithm 1, we have the following results.

Lemma 2.2.3. *$\tilde{\boldsymbol{x}}$ is a feasible 0-1 solution of (2.5) and $(\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{z}})$ is a feasible solution of (2.6).*

Proof. By the assumption that MKPFC (2.2) is feasible, $\boldsymbol{x} = (1, \cdots, 1)$ is feasible for the LP relaxation problem (2.5). Algorithm 1 starts from $\boldsymbol{x} = \boldsymbol{0}$ and updates a variable x_j from 0 to 1 at each iteration until all the constraints in (2.5) are satisfied. Hence $\tilde{\boldsymbol{x}}$ is a feasible 0-1 solution of (2.5).

Algorithm 1 starts from the dual feasible solution $(\boldsymbol{y}, \boldsymbol{z}) = (\boldsymbol{0}, \boldsymbol{0})$ and maintains dual feasibility throughout the algorithm. Hence $(\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{z}})$ is feasible for (2.6). □

Lemma 2.2.4. *$\tilde{\boldsymbol{x}}$ and $(\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{z}})$ satisfy (2.7).*

Proof. Since $\boldsymbol{x} = \boldsymbol{0}$ at the beginning and the algorithm sets $x_j = 1$ only if the j-th constraint in (2.6.1) becomes tight, (a) of (2.7) is satisfied. (b-1) of (2.7) follows from $\tilde{\boldsymbol{x}} \in \{0, 1\}^n$. Thus it suffices to show that (b-2) holds. We consider two cases, whether or not the algorithm stops at the first iteration of Step 2.

If the algorithm stops at the first iteration of Step 2, we obtain a primal feasible solution in Step 1. Then (b-2) holds since $\tilde{y}(A) = 0$ for any $A \subseteq V$. Conversely, if we have a primal feasible solution in Step 1, then the algorithm stops at the first iteration of Step 2 since $\bar{b} \leq 0$ holds.

Suppose that the algorithm does not stop at the first iteration of Step 2. Define $\tilde{S} = \{j \in V \mid \tilde{x}_j = 1\}$. Let $\tilde{x}_\ell$ be the variable which becomes 1 from 0 at the last iteration of Step 2. From Step 2, $\tilde{y}(A) > 0$ implies

$$A \subseteq \tilde{S} \setminus \{\ell\}. \tag{2.8}$$

Since the algorithm does not stop just before setting $\tilde{x}_\ell = 1$, we have

$$\sum_{j \in \tilde{S} \setminus \{\ell\}} a_j < b. \tag{2.9}$$

From (2.8) and (2.9), we observe that for any subset $A \subseteq N$ such that $\tilde{y}(A) > 0$

$$\sum_{j \in (\tilde{S} \setminus \{\ell\}) \setminus A} a_j(A) \leq \sum_{j \in (\tilde{S} \setminus \{\ell\}) \setminus A} a_j = \sum_{j \in \tilde{S} \setminus \{\ell\}} a_j - \sum_{j \in A} a_j < b - \sum_{j \in A} a_j \leq b(A),$$

where the first and last inequality follows from the definitions (2.4) of $a_j(A)$ and $b(A)$. Thus, we have that for any subset $A \subseteq N$ such that $\tilde{y}(A) > 0$

$$\sum_{j \in V \setminus A} a_j(A)\tilde{x}_j = \sum_{j \in \tilde{S} \setminus A} a_j(A) = \sum_{j \in (\tilde{S} \setminus \{\ell\}) \setminus A} a_j(A) + a_\ell(A) < 2b(A),$$

where the last inequality follows from $a_\ell(A) \leq b(A)$. □

Lemma 2.2.5. *The running time of Algorithm 1 is $O(|E| + |V'|^2)$, where $V' = \{j \in V \mid a_j > 0\}$.*

Proof. The running time of one iteration of Step 1 is $O(1)$ and the number of iterations in Step 1 is at most $|E|$. The running time of one iteration of Step 2 is $O(|V'|)$ and the number of iterations in Step 2 is at most $|V'|$. Therefore the running time of the algorithm is $O(|E| + |V'|^2)$. □

The following result follows from Corollary 2.2.1 and Lemmas 2.2.3, 2.2.4, and 2.2.5.

Theorem 2.2.1. *Algorithm 1 is a 2-approximation algorithm for MKPFC (2.2).*

2.3 Generalization to the covering 0-1 integer program

In this section, we generalize Algorithm 1 to the covering 0-1 integer program (CIP), which is represented as

$$\text{CIP}\left|\begin{array}{lll} \min & \displaystyle\sum_{j\in N} c_j x_j & \\ \text{s.t.} & \displaystyle\sum_{j\in N} a_{ij}x_j \geq b_i, & \forall i \in M = \{1,\cdots,m\}, \\ & x_j \in \{0,1\}, & \forall j \in N = \{1,\cdots,n\}, \end{array}\right. \tag{2.10}$$

where b_i, a_{ij}, and c_j ($i \in M,\ j \in N$) are nonnegative. Assume that $\sum_{j\in N} a_{ij} \geq b_i$ for any $i \in M$, so that the problem is feasible. Let f_i be the number of non-zero coefficients in the i-th constraint $\sum_{j\in N} a_{ij}x_j \geq b_i$. Without loss of generality, we assume that $f_1 \geq f_2 \geq \cdots \geq f_m$ and $f_2 \geq 2$. There are some f_1-approximation algorithms for CIP, see Koufogiannakis and Young [35] and references therein. We propose a f_2-approximation algorithm. The minimum knapsack problem with a forcing graph (2.2) is a special case of CIP for which $f_2 = 2$.

We show an LP relaxation problem of CIP by Carr et al. [10]. The relaxation problem is represented as

$$\begin{array}{lll} \min & \displaystyle\sum_{j\in N} c_j x_j & \\ \text{s.t.} & \displaystyle\sum_{j\in N\setminus A} a_{ij}(A)x_j \geq b_i(A), & \forall A \subseteq N,\ \forall i \in M, \\ & x_j \geq 0, & \forall j \in N, \end{array} \tag{2.11}$$

where

$$\begin{array}{rcl} b_i(A) & = & \max\{0, b_i - \sum_{j\in A} a_{ij}\},\ \forall i \in M, \forall A \subseteq N, \\ a_{ij}(A) & = & \min\{a_{ij}, b_i(A)\},\ \forall i \in M, \forall A \subseteq N, \forall j \in N\setminus A. \end{array} \tag{2.12}$$

Carr et al. [10] show that any feasible 0-1 solution of (2.11) is feasible for (2.10). The dual problem of (2.11) can be stated as

$$\begin{array}{lll} \max & \displaystyle\sum_{i\in M}\sum_{A\subseteq N} b_i(A)y_i(A) & \\ \text{s.t.} & \displaystyle\sum_{i\in M}\sum_{A\subseteq N:\, j\notin A} a_{ij}(A)y_i(A) \leq c_j, & \forall j \in N, \\ & y_i(A) \geq 0, & \forall A \subseteq N,\ \forall i \in M. \end{array} \tag{2.13}$$

By applying Lemma 2.2.1 to the LP problems (2.11) and (2.13), we have the following result.

Lemma 2.3.1. *Let $\boldsymbol{x}$ be a feasible 0-1 solution of (2.11) and $\boldsymbol{y}$ be a feasible solution of (2.13). If these solutions satisfy*

$$\begin{array}{ll} \textit{(a):} & \forall j \in N, x_j > 0 \Rightarrow \sum_{i\in M} \sum_{A\subseteq N: j\notin A} a_{ij}(A) y_i(A) = c_j, \\ \textit{(b):} & \forall i \in M,\ \forall A \subseteq N, y_i(A) > 0 \Rightarrow \sum_{j\in N\setminus A} a_{ij}(A) x_j \le f_2 b(A), \end{array} \tag{2.14}$$

then $\boldsymbol{x}$ is a solution within a factor of f_2 of the optimal solution of (2.10).

Our algorithm is presented in Algorithm 2 below. The goal is to find $\boldsymbol{x}$ and $\boldsymbol{y}$ which satisfy the conditions in Lemma 2.3.1. The algorithm generates a sequence of points $\boldsymbol{x}$ and $\boldsymbol{y}$. Throughout the algorithm, the conditions $\boldsymbol{x} \in \{0,1\}^n$, constraints in (2.13), and (2.14) are satisfied. The constraints in (2.11) are satisfied at Step 2. In Algorithm 2, we use the symbols $S = \{j \in N \mid x_j = 1\}$, $b_i(S) = \max\{0, b_i - \sum_{j\in S} a_{ij}\}$ for $i \in M$, and $\bar{c}_j = c_j - \sum_{i\in M} \sum_{A\subseteq N: j\notin A} a_{ij}(A) y_i(A)$ for $j \in N$.

Algorithm 2

Input: $M,\ N,\ a_{ij}, b_i$ and c_j $(i \in M,\ j \in N)$.

Output: $\tilde{\boldsymbol{x}}$ and $\tilde{\boldsymbol{y}}$.

Step 0: Set $\boldsymbol{x} = \boldsymbol{0}$, $\boldsymbol{y} = \boldsymbol{0}$, and $S = \emptyset$. Let $N_i' = \{j \in N \mid a_{ij} > 0\}$ for $i \in M$, $\bar{c}_j = c_j$ for $j \in N$, and $i = m$.

Step 1: If $i = 0$, then output $\tilde{\boldsymbol{x}} = \boldsymbol{x}$ and $\tilde{\boldsymbol{y}} = \boldsymbol{y}$ and stop. Otherwise set $b_i(S) = \max\{0, b_i - \sum_{j\in S} a_{ij}\}$ and go to Step 2.

Step 2: If $b_i(S) = 0$, then update $i = i - 1$ and go to Step 1. Otherwise calculate $a_{ij}(S)$ for any $j \in N_i'\setminus S$ by (2.12). Increase $y_i(S)$ while maintaining dual feasibility until at least one constraint $s \in N_i'\setminus S$ is tight. Namely set

$$y_i(S) = \frac{\bar{c}_s}{a_{is}(S)} \quad \text{for} \quad s = \arg\min_{j\in N_i'\setminus S} \left\{ \frac{\bar{c}_j}{a_{ij}(S)} \right\}.$$

Update $\bar{c}_j = \bar{c}_j - a_{ij}(S) y_i(S)$ for $j \in N'\setminus S$, $x_s = 1$, $S = S \cup \{s\}$, and $b_i(S) = \max\{0, b_i(S) - a_{is}\}$. Go back to the top of Step 2.

In the same way as the proof of Lemma 2.2.3, we have the following result for the outputs $\tilde{\boldsymbol{x}}$ and $\tilde{\boldsymbol{y}}$ of Algorithm 2.

Lemma 2.3.2. *$\tilde{\boldsymbol{x}}$ is a 0-1 feasible solution of (2.11) and $\tilde{\boldsymbol{y}}$ is a feasible solution of (2.13).*

The next lemma is similarly proved as Lemma 2.2.4.

Lemma 2.3.3. *$\tilde{\boldsymbol{x}}$ and $\tilde{\boldsymbol{y}}$ satisfy (2.14).*

Proof. All the conditions in (a) of (2.14) are satisfied by the way the algorithm updates primal variables. It suffices to show that all the conditions in (b) are satisfied. For any $i \in \{2, \cdots, m\}$ and any subset $A \subseteq N$ such that $\tilde{y}_i(A) > 0$, we obtain that

$$\sum_{j \in N \setminus A} a_{ij}(A)\tilde{x}_j \leq f_i b_i(A) \leq f_2 b_i(A),$$

since $a_{ij}(A) \leq b_i(A)$ by the definition (2.12) and the i-th constraint has f_i non-zero coefficients. Then, we consider the case of $i = 1$. In a similar way as the proof in Lemma 2.2.4, for any subset $A \subseteq N$ such that $\tilde{y}_1(A) > 0$, we have

$$\sum_{j \in N \setminus A} a_{1j}(A)\tilde{x}_j \leq 2b_1(A) \leq f_2 b_1(A).$$

□

Lemma 2.3.4. *The running time of Algorithm 2 is $O(f_1(m + n))$.*

Proof. The running time of one iteration of Step 1 is $O(f_1)$ and the number of iterations in Step 1 is at most m. On the other hand, the running time of one iteration of Step 2 is $O(f_1)$ and the number of iterations in Step 2 is at most $m + n$. Therefore the total running time of the algorithm is $O(f_1 m) + O(f_1(m + n)) = O(f_1(m + n))$. □

From the results above, we can obtain the next theorem.

Theorem 2.3.1. *Algorithm 2 is an f_2-approximation algorithm for CIP (2.10).*

Chapter 3

An improved approximation algorithm for the covering 0-1 integer program

3.1 Overview

The covering 0–1 integer program (CIP) is formulated as follows:

$$\text{CIP} \left| \begin{array}{lll} \min & \displaystyle\sum_{j \in N} c_j x_j & \\ \text{s.t.} & \displaystyle\sum_{j \in N} a_{ij} x_j \geq b_i, & \forall i \in M, \\ & x_j \in \{0,1\}, & \forall j \in N, \end{array} \right. \tag{3.1}$$

where $M = \{1, \ldots, m\}$, $N = \{1, \ldots, n\}$, $c_j \geq 0$ ($j \in N$), $a_{ij} \geq 0$ ($i \in M$, $j \in N$) and $b_i \geq 0$ ($i \in M$).

There is no $o(\log m)$-approximation algorithms for CIP unless P = NP since the set covering problem is a special case of CIP [44]. Kolliopoulos and Young [34] present an $O(\log m)$-approximation algorithm for CIP. Let f be the maximum number of non-zero entries in the constraints. For any $f \geq 2$ and $\epsilon > 0$, CIP is hard to approximate better than a factor of $f - 1 - \epsilon$ unless P=NP [17] and $f - \epsilon$ under the unique games conjecture [32]. In this paper, we focus on algorithms whose approximation ratios depend on f.

For CIP, f-approximation algorithms are proposed [10, 21, 22]. In Chapter 2 we present an f_2-appoximation algorithm for CIP, where f_2 is the second largest

number of non-zero entries in the constraints. Note that f_2 is less than or equal to f. Approximation algorithms for generalizations of CIP are also well studied. Koufogiannakis and Young [35] and Pritchard and Chakrabarty [40] give f-approximation algorithms for CIP with general upper bounds on variables. McCormick et al. [36] develop an approximation algorithm for precedence constrained CIP. To the best of our knowledge, there are no approximation algorithms for CIP with approximation ratio strictly less than f when $f \geq 2$.

Table 3.1: Approximation ratios for problems related to CIP

Problem Names	Restrictions on CIP	Approximation Ratios
Covering 0–1 Integer Program (CIP)	-	f (Carr et al. 2000; Fujito 2004; Fujito and Yabuta 2004; Koufogiannakis and Young 2005; Pritchard and Chakrabarty 2011) f_2 (Chapter 2) $f - \frac{f-1}{m}$ (**this thesis**)
Minimum Knapsack Problem (MKP)	$m = 1$	FPTAS (Kellerer et al. 2004)
(Weighted) Set Covering Problem (SCP)	$a_{ij} \in \{0, 1\},\ b_i = 1$	$f\left(1 - (f-1)\frac{\ln\ln\Delta}{\ln\Delta}\right)$ (Halperin 2002)
Vertex Cover Problem (VCP)	$a_{ij} \in \{0, 1\},\ b_i = 1$ $c_j = 1,\ f = 2$	$2 - \frac{\ln\ln\Delta}{\ln\Delta}$ (Halperin 2002) $2 - \Theta(1/\sqrt{\log m})$ (Karakostas 2009)

where $i^* \in M$ is an index such that $|\{j \in N \mid a_{i^*j} > 0\}| = f$ and

$$\begin{aligned} f &= \max_{i \in M} |\{j \in N \mid a_{ij} > 0\}|, \\ f_2 &= \max_{i \in M \setminus i^*} |\{j \in N \mid a_{ij} > 0\}|, \\ \Delta &= \max_{i \in N} |\{i \in M \mid a_{ij} > 0\}|. \end{aligned} \tag{3.2}$$

While there are no approximation algorithms for CIP with approximation ratio better than f, there are such approximation algorithms for special cases of CIP such as the minimum knapsack problem (MKP), the set covering problem (SCP) and the vertex cover problem (VCP), see Table 1. MKP is a special case of CIP when the number of the constraints is only one and can be regarded as the minimization version of the knapsack problem (KP). It is well-known that KP and MKP admit a fully polynomial time approximation scheme (FPTAS) [31]. SCP is a special case of CIP when $a_{ij} \in \{0, 1\}$ and $b_i = 1$ and VCP is a special case of SCP

when $f = 2$ and $c_j = 1$. Halperin [25] gives a $(2 - \ln\ln\Delta/\ln\Delta)$-approximation algorithm for VCP, where Δ is the maximal degree of the graph, and extends this result to SCP. Karakostas [30] obtains a $(2 - \Theta(1/\sqrt{\log m}))$-approximation algorithm for VCP. Both the approaches use SDP-relaxation. Fujito [21] gives approximation algorithms with approximation ratios better than f for some special cases of CIP including SCP and VCP. For more information on approximation algorithms for CIP and its special cases, we refer the reader to Fujito [21].

Contribution

In this chapter, we propose an $\left(f - \frac{f-1}{m}\right)$-approximation algorithm for CIP with $m \geq 2$. When $m = 1$ (MKP), for any fixed $\epsilon > 0$, our algorithm finds a solution whose objective value is less than or equal to $(1 + \epsilon)$ times the optimal value within polynomial in m, n and $n^{\lceil 1/\epsilon \rceil}$, that is, it can be regarded as a polynomial time approximation scheme (PTAS). Our algorithm is the first algorithm for CIP with approximation ratio strictly less than f when $f \geq 2$.

The main idea of our algorithm is as follows. Our algorithm uses on an enumeration technique for PSCP by Gandhi et al [23]. We show that a primal-dual algorithm (called PD) for CIP by Fujito [21] gives f-approximation but it can give $\left(f - \frac{f-1}{m}\right)$-approximation if we know partial information about an optimal solution. Then we propose an algorithm, called the main algorithm, where we get $\left(f - \frac{f-1}{m}\right)$-approximation solution without information about an optimal solution by solving $O(n^2)$ subproblems with PD.

Notation and Assumption

Let $I = (\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c})$ be a data of (3.1), where $\boldsymbol{A}$ is the matrix of a_{ij}. We call I an instance of CIP. Let CIP(I) be the problem for instance I. Let I_0 be an instance to be solved. Without loss of generality, for the problem CIP(I_0), we assume that

- $f \geq 2$,
- CIP(I_0) is feasible.

Let $OPT(I)$ be the optimal value of CIP(I) when CIP(I) is feasible. For any subset $S \subseteq N$, we define a vector $\boldsymbol{x}(S) \in \mathbb{R}^n$ as follows:

$$x_j(S) = \begin{cases} 1 & \text{if } j \in S \\ 0 & \text{if } j \notin S \end{cases} \quad \text{for any } j \in N. \tag{3.3}$$

3.2 Main algorithm

In this section, we present the main algorithm. The main algorithm solves many subproblems of CIP by the algorithm PD in Fujito [21] as a subroutine. The algorithm PD and its analysis are presented in Section 3. This section is organized as follows:

1. We show a property (Lemma 3.2.1) of the solution generated by PD.
2. We explain that we get an $\left(f - \frac{f-1}{m}\right)$-approximation solution by using PD if we know partial information about an optimal solution.
3. We present the main algorithm which gives an $\left(f - \frac{f-1}{m}\right)$-approximation without information about an optimal solution.

First, we state a property of the solution generated by PD. Details of PD and the proof of Lemma 3.2.1 are shown in Section 3.

Lemma 3.2.1. *For any instance I, the algorithm PD presented in Section 3 runs in $O(mn^2)$ and determines feasibility of CIP(I). If CIP(I) is feasible, the algorithm PD produces a feasible solution $\boldsymbol{x}$ satisfying*

$$\sum_{j \in N} c_j x_j \leq \left(f - \frac{f-1}{m}\right) OPT(I) + c_{\max},$$

where $c_{\max} = \max_{j \in N} c_j$.

For any $A \subseteq N$, define

$$\begin{array}{rcl} N(A) & = & \{j \in N \backslash A \mid c_j \leq \min_{j' \in A} c_{j'}\}, \\ \bar{N}(A) & = & \{j \in N \backslash A \mid c_j > \min_{j' \in A} c_{j'}\}. \end{array} \tag{3.4}$$

We see that $\{A,\ N(A),\ \bar{N}(A)\}$ is a partition of N. For an instance I_0 and $A \subseteq N$, we consider a subproblem of CIP(I_0) by fixing some variables as follows:

$$\begin{array}{ll} x_j = 1 & \text{if } j \in A, \\ x_j = 0 & \text{if } j \in \bar{N}(A). \end{array}$$

This subproblem can be expressed as:

$$\begin{array}{lll} \min & \displaystyle\sum_{j \in N(A)} c_j x_j & \\ \text{s.t.} & \displaystyle\sum_{j \in N(A)} a_{ij} x_j \geq \max\left\{b_i - \sum_{j \in A} a_{ij},\ 0\right\}, & \forall i \in M, \\ & x_j \in \{0, 1\}, & \forall j \in N(A). \end{array} \tag{3.5}$$

Note that this problem is also CIP and the number of decision variables is $|N(A)|$. Let $I_0(A)$ be the instance of this subproblem . Let $\bar{\boldsymbol{x}}^A \subseteq \{0,1\}^{|N(A)|}$ be an output by the algorithm PD for the subproblem CIP($I_0(A)$) when CIP($I_0(A)$) is feasible. Define a solution $\boldsymbol{x}^A \subseteq \{0,1\}^n$ for CIP(I_0) as

$$\boldsymbol{x}^A = \begin{cases} \bar{x}_j^A & \text{if } j \in N(A), \\ 1 & \text{if } j \in A, \\ 0 & \text{if } j \in \bar{N}(A). \end{cases} \tag{3.6}$$

Let S^* be a subset of N such that $\boldsymbol{x}(S^*)$ is an optimal solution of CIP(I_0). For any positive integer k such that $k \leq |S^*|$, we define $A_k^* \subseteq S^*$ as $\{j_1, \ldots, j_k\}$ such that $\{c_{j_1}, \ldots, c_{j_k}\}$ is a set of the k largest numbers in $\{c_j \mid j \in S^*\}$. We get an $\left(f - \frac{f-1}{m}\right)$-approximation solution for CIP(I_0) by using PD if we know A_2^*.

Lemma 3.2.2. *If $m \geq 2$, $k = 2$ and $k < |S^*|$, then $\boldsymbol{x}^{A_k^*}$ defined by (3.6) is feasible to CIP(I_0) and the following inequality holds:*

$$\sum_{j \in N} c_j x_j^{A_k^*} \leq \left(f - \frac{f-1}{m}\right) OPT(I_0).$$

Proof. First, we will prove that $\boldsymbol{x}^{A_k^*}$ is feasible to CIP(I_0). The problem CIP($I_0(A_k^*)$) is feasible since a subvector of $\boldsymbol{x}(S^* \backslash A_k^*)$ is a feasible solution for CIP($I_0(A_k^*)$). Thus, PD outputs a feasible solution $\bar{\boldsymbol{x}}^{A_k^*}$ for CIP($I_0(A_k^*)$). Then, a solution $\boldsymbol{x}^{A_k^*}$ defined by (3.6) is clearly feasible to CIP(I_0).

Let $\alpha = f - \frac{f-1}{m}$. Let f' be the maximum number of non-zero entries in the constraints of the subproblem CIP($I_0(A_k^*)$). we see that $f' - \frac{f'-1}{m} \leq \alpha$ from $f' \leq f$. From Lemma 1, the algorithm PD outputs a solution $\bar{\boldsymbol{x}}^{A_k^*}$ for the problem CIP($I_0(A_k^*)$) and the next inequality holds:

$$\sum_{j \in N(A_k^*)} c_j \bar{x}_j^{A_k^*} \leq \left(f' - \frac{f'-1}{m}\right) OPT(I_0(A_k^*)) + \max_{j \in N(A_k^*)} c_j \leq \alpha OPT(I_0(A_k^*)) + \max_{j \in N(A_k^*)} c_j. \tag{3.7}$$

From the definition of the subproblem CIP($I_0(A_k^*)$), we have that

$$OPT(I_0) = \sum_{j \in S^* \backslash A_k^*} c_j + \sum_{j \in A_k^*} c_j = OPT(I_0(A_k^*)) + \sum_{j \in A_k^*} c_j. \tag{3.8}$$

From the definition of $N(A_k^*)$ in (3.4), we obtain that

$$\max_{j \in N(A_k^*)} c_j \leq \min_{j \in A_k^*} c_j \leq \frac{1}{k} \sum_{j \in A_k^*} c_j. \tag{3.9}$$

From (3.7), (3.8) and (3.9), we have that

$$\begin{aligned}
\sum_{j\in N} c_j x_j^{A_k^*} &= \sum_{j\in N(A_k^*)} c_j \bar{x}_j^{A*} + \sum_{j\in A_k^*} c_j \\
&\leq \alpha OPT(I_0(A_k^*)) + \max_{j\in N(A_k^*)} c_j + \sum_{j\in A_k^*} c_j \\
&= \alpha\left(OPT(I_0(A_k^*)) + \sum_{j\in A_k^*} c_j\right) + (1-\alpha)\sum_{j\in A_k^*} c_j + \max_{j\in N(A_k^*)} c_j \\
&\leq \alpha OPT(I_0) + \left(1-\alpha+\frac{1}{k}\right)\sum_{j\in A_k^*} c_j \qquad (3.10)
\end{aligned}$$

Since $m \geq 2$ and $f \geq 2$ from the assumptions, $\alpha = f - \frac{f-1}{m} \geq 3/2$ holds. From this observation and $k = 2$, we obtain that

$$\left(1-\alpha+\frac{1}{k}\right)\sum_{j\in A_k^*} c_j \leq \left(1-\frac{3}{2}+\frac{1}{2}\right)\sum_{j\in A_k^*} c_j = 0.$$

Therefore, we get

$$\sum_{j\in N} c_j x_j^{A_k^*} \leq \alpha OPT(I_0).$$

□

When $m = 1$, we get a $\left(1+\frac{1}{k}\right)$-approximation solution for any fixed positive integer k such that $k < |S^*|$.

Lemma 3.2.3. *If $m = 1$, for any positive integer k such that $k < |S^*|$, $\boldsymbol{x}^{A_k^*}$ is feasible to CIP(I) and the following inequality holds:*

$$\sum_{j\in N} c_j x_j^{A_k^*} \leq \left(1+\frac{1}{k}\right) OPT(I_0).$$

Proof. The proof of Lemma 3.2.2 until (3.10) also holds in this case. By substituting $\alpha = f - \frac{f-1}{m} = 1$ in (3.10), we have that

$$\sum_{j\in N} c_j x_j^{A_k^*} \leq OPT(I_0) + \frac{1}{k}\sum_{j\in A_k^*} c_j \leq \left(1+\frac{1}{k}\right) OPT(I_0),$$

where the last inequality holds since $\sum_{j\in A_k^*} c_j \leq OPT(I_0)$ from $A_k^* \subseteq S^*$. □

Even though Lemma 3.2.2 and Lemma 3.2.3 require the information about A_k^*, we don't need it in advance if we execute PD for all supproblems CIP($I_0(A)$) such that $A \subseteq N$ and $|A| \leq k$. The main algorithm is presented as follows:

The main algorithm

Input: $I_0 = (\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c})$ and a positive integer k.

Step 0: Calculate $\mathcal{D}(k)$ defined by

$$\mathcal{D}(k) = \{A \subseteq N \mid |A| \leq k\ |\}.$$

Step 1: For each $A \in \mathcal{D}(k)$, do the following process:
Let $I_0(A)$ be the data derived from the subproblem (3.5). If $\boldsymbol{x}(A)$ is feasible to CIP(I_0), that is $\sum_{j \in A} a_{ij} \geq b_i$ for any $i \in M$, go to Step 1-A. Otherwise go to Step 1-B.
(**Step 1-A**): Make a solution for CIP(I_0) as follows:

$$\boldsymbol{x}^A = \boldsymbol{x}(A).$$

(**Step 1-B**): Execute the algorithm PD for the subproblem CIP($I_0(A)$) defined by (3.5). If the problem CIP($I_0(A)$) is feasible, the algorithm outputs a feasible solution for CIP($I_0(A)$). Denote this solution by $\bar{\boldsymbol{x}}^A \subseteq \{0,1\}^{|N(A)|}$. By using this solution, make a solution $\boldsymbol{x}^A$ for CIP(I_0) by (3.6):

$$\boldsymbol{x}^A = \begin{cases} \bar{x}_j^A & \text{if } j \in N(A), \\ 1 & \text{if } j \in A, \\ 0 & \text{if } j \in \bar{N}(A). \end{cases}$$

Step 2: Set $\hat{A} = \underset{A \subseteq \mathcal{D}(k)}{\arg\min} \sum_{j \in N} c_j x_j^A$ and output $\boldsymbol{x}^{\hat{A}}$.

Theorem 3.2.1. *Suppose $m \geq 2$ and set $k = 2$ in the main algorithm. Then the main algorithm is an $\left(f - \frac{f-1}{m}\right)$-approximation algorithm for CIP.*

Proof. The running time of one iteration of Step 1 is $O(mn^2)$ from Lemma 3.2.1. The number of iterations of the main algorithm is $O(n^k)$. Thus, the main algorithm runs in polynomial time.

If $k \geq |S^*|$, that implies $S^* \in \mathcal{D}(k)$. We consider when Step 1 is executed for $A = S^*$. In this case, the algorithm goes to Step 1-A since $\boldsymbol{x}(S^*) = \boldsymbol{x}^*$ is feasible to

CIP(I_0) and sets $\boldsymbol{x}^{S^*} = \boldsymbol{x}^*$ at this iteration. Thus, the algorithm outputs an optimal solution.

Next we consider the case when $k < |S^*|$. In this case, $A_k^* \in \mathcal{D}(k)$ holds and that implies the main algorithm executes Step 1-B for the set A_k^* since $\boldsymbol{x}(A_k^*)$ is infeasible to CIP(I_0). From Lemma 3.2.2, we have that

$$\sum_{j \in N} c_j x_j^{A_k^*} \leq \left(f - \frac{f-1}{m} \right) OPT(I_0).$$

Therefore, we get

$$\sum_{j \in N} c_j x_j^{\hat{A}} = \min_{A \subseteq \mathcal{D}(k)} \sum_{j \in N} c_j x_j^A \leq \sum_{j \in N} c_j x_j^{A_k^*} \leq \left(f - \frac{f-1}{m} \right) OPT(I_0).$$

□

When $m = 1$, we have the following result from Lemma 3.2.3 in the same way as the proof of Theorem 1.

Theorem 3.2.2. *For any fixed $\epsilon > 0$, set $k = \lceil 1/\epsilon \rceil$ in the main algorithm. If $m = 1$, then the main algorithm finds a feasible solution whose objective value is less than or equal to $(1 + \epsilon)$ times the optimal value within polynomial in m, n and $n^{\lceil 1/\epsilon \rceil}$.*

3.3 Algorithm PD and proof of Lemma 3.2.1

In this section, we present the algorithm PD proposed by Fujito [21] and prove Lemma 3.2.1. First we show a relaxation problem of CIP, which is utilized by PD. Let

$$\begin{aligned} b_i(A) &= \max\{0, b_i - \textstyle\sum_{j \in A} a_{ij}\}, \ \forall i \in M, \forall A \subseteq N, \\ a_{ij}(A) &= \min\{a_{ij}, b_i(A)\}, \ \forall i \in M, \forall A \subseteq N, \forall j \in N \backslash A, \\ M(A) &= \{i \in M \mid b_i(A) > 0\}, \ \forall A \subseteq N, \\ U_j(A) &= \sum_{i \in M(A)} \frac{a_{ij}(A)}{b_i(A)}, \forall A \subseteq N, \forall j \in N \backslash A \end{aligned} \tag{3.11}$$

Using these symbols, we have the following problem.

$$\begin{aligned} \min \quad & \sum_{j \in N} c_j x_j \\ \text{s.t.} \quad & \sum_{j \in N \backslash A} U_j(A) x_j \geq |M(A)|, \quad \forall A \subseteq N \\ & x_j \geq 0, \qquad\qquad\qquad \forall j \in N. \end{aligned} \tag{3.12}$$

This problem is a relaxation problem of CIP from Proposition 1 in Fujito [21].

Lemma 3.3.1. *(3.12) is a relaxation problem of CIP, that is, any feasible solution $\boldsymbol{x}$ for CIP is feasible to (3.12).*

The dual problem of (3.12) is expressed as

$$\begin{aligned} \max \quad & \sum_{A\subseteq N} |M(A)|y(A) \\ \text{s.t.} \quad & \sum_{A\subseteq N: j\notin A} U_j(A)y(A) \le c_j, \quad \forall j \in N, \\ & y(A) \ge 0, \qquad\qquad\qquad \forall A \subseteq N. \end{aligned} \tag{3.13}$$

Now, we show a useful and well-known result in analysis of the primal-dual method.

Lemma 3.3.2. *Let $\boldsymbol{x} \in \{0,1\}^n$ and let $\boldsymbol{y}$ be a feasible solution to (3.13). For $\alpha \ge 0$, if $\boldsymbol{x}$ and $\boldsymbol{y}$ satisfy*

(a) $\forall j \in N,\ x_j = 1 \Rightarrow \sum_{A\subseteq N: j\notin A} U_j(A)y(A) = c_j$,
(b) $\forall A \subseteq N,\ y(A) > 0 \Rightarrow \sum_{j\in N\setminus A} U_j(A)x_j \le \alpha|M(A)|$,

then the following inequality holds:

$$\sum_{j\in N} c_j x_j \le \alpha OPT(I).$$

Proof. Suppose $\boldsymbol{x}$ and $\boldsymbol{y}$ satisfy the conditions of Lemma 3.3.2. Let $S = \{j \in N \mid x_j = 1\}$. From the condition (a), we have that

$$\sum_{j\in N} c_j x_j = \sum_{j\in S} c_j = \sum_{j\in S} \sum_{A\subseteq N: j\notin A} U_j(A)y(A) = \sum_{A\subseteq N} \sum_{j\in S\setminus A} U_j(A)y(A).$$

From the condition (b), we obtain that

$$\sum_{A\subseteq N} \sum_{j\in S\setminus A} U_j(A)y(A) = \sum_{A\subseteq N} y(A) \sum_{j\in S\setminus A} U_j(A) \le \alpha \sum_{A\subseteq N} |M(A)|y(A).$$

Since $\boldsymbol{y}$ is feasible to (3.13), the objective value of $\boldsymbol{y}$ is less than or equal to the optimal value of (3.12), which is less than or equal to OPT(I). Therefore, we get

$$\sum_{j\in N} c_j x_j \le \alpha \sum_{A\subseteq N} |M(A)|y(A) \le \alpha OPT(I).$$

□

The algorithm PD is presented below. Solutions generated by the algorithm, except for the final solution, satisfy all the conditions in Lemma 3.3.2 for $\alpha = f - \frac{f-1}{m}$.

Algorithm PD

Input: an instance I.

Step 0: Set $\boldsymbol{x} = \mathbf{0}$, $\boldsymbol{y} = \mathbf{0}$ and $\bar{\boldsymbol{c}} = \boldsymbol{c}$. Check whether the solution $(1, \cdots, 1)$ is feasible to CIP(I) or not. If it is not feasible, declare INFEASIBLE and stop.

Step 1: Let

$$\begin{aligned} S &= \{j \in N \mid x_j = 1\}, \\ b_i(S) &= \max\{0, b_i - \textstyle\sum_{j \in S} a_{ij}\}, \ \forall i \in M, \\ a_{ij}(S) &= \min\{a_{ij}, b_i(S)\}, \ \forall i \in M, \forall j \in N \backslash S, \\ M(S) &= \{i \in M \mid b_i(S) > 0\}, \\ U_j(S) &= \textstyle\sum_{i \in M(S)} \frac{a_{ij}(S)}{b_i(S)}, \ \forall j \in N \backslash S, \\ N_{>0}(S) &= \{j \in N \backslash S \mid U_j(S) > 0\}. \end{aligned}$$

If $M(S) = 0$, output $\boldsymbol{x}$, $\boldsymbol{y}$ and stop. Otherwise, go to Step 2.

Step 2: Increase $y(S)$ as much as possible while maintaining dual feasibility for (3.13). That is, set

$$y(S) = \frac{\bar{c}_t}{U_t(S)},$$

where

$$t = \arg\min_{j \in N_{>0}(S)} \frac{\bar{c}_j}{U_j(S)}$$

Set $\bar{c}_j := \bar{c}_j - U_j(S) y(S)$ for all $j \in N_{>0}(S)$. Update $x_t = 1$. Go back to Step 1.

Fujito [21] shows that the algorithm PD is an f-approximation algorithm for CIP since we easily show that outputs of PD satisfies the conditions in Lemma 3.3.2 for $\alpha = f$. In this study, we show that solutions produced by PD satisfies the stronger conditions in Lemma 3.3.2.

Lemma 3.3.3. *Let $\boldsymbol{x}$ be the output by PD and x_ℓ be the variable which becomes 1 from 0 at the last iteration of PD. Let $\tilde{\boldsymbol{x}}$ be the solution obtained by setting $x_\ell = 0$ in $\boldsymbol{x}$. Let $\tilde{\boldsymbol{y}}$ be the dual solution at the end of the iteration before x_ℓ becomes 1. Then $\tilde{\boldsymbol{x}}$ and $\tilde{\boldsymbol{y}}$ satisfy the conditions in Lemma 3.3.2 for $\alpha = f - \frac{f-1}{m}$.*

Proof. Let $\tilde{S} = \{j \in N \mid \tilde{x}_j = 1\}$. $\tilde{\boldsymbol{y}}$ is feasible to the dual (3.13) since PD starts from the dual feasible solution $\boldsymbol{y} = 0$ and maintains dual feasibility at every iteration. Note that $\tilde{\boldsymbol{x}}$ is infeasible to (3.1). $\tilde{\boldsymbol{x}}$ and $\tilde{\boldsymbol{y}}$ satisfies (a) in Lemma 3.3.2 by the way the algorithm updates $\boldsymbol{x}$ and $\boldsymbol{y}$. Therefore it suffices to show that (b) in Lemma 3.3.2 holds, that is, for any $A \subseteq N$ such that $\tilde{y}(A) > 0$, the following holds:

$$\sum_{j \in N \setminus A} U_j(A)\tilde{x}_j = \sum_{j \in \tilde{S} \setminus A} U_j(A) = \sum_{i \in M(A)} \sum_{j \in \tilde{S} \setminus A} \frac{a_{ij}(A)}{b_i(A)} \leq \left(f - \frac{f-1}{m}\right)|M(A)|,$$

where we use the definition of $U_j(A)$ by (3.11).

Now, we fix $A \subseteq N$ such that $\tilde{y}(A) > 0$. From Step 2, $\tilde{y}(A) > 0$ implies

$$A \subseteq \tilde{S}. \tag{3.14}$$

From (3.14) and the definition of $M(A)$ by (3.11), we have that

$$M(\tilde{S}) \subseteq M(A), \tag{3.15}$$

and for any $i \in M(\tilde{S})$

$$\sum_{j \in \tilde{S}} a_{ij} < b_i. \tag{3.16}$$

From (3.11), (3.14) and (3.16), for any $i \in M(\tilde{S})$, we obtain that

$$\sum_{j \in \tilde{S} \setminus A} a_{ij}(A) \leq \sum_{j \in \tilde{S} \setminus A} a_{ij} = \sum_{j \in \tilde{S}} a_{ij} - \sum_{j \in A} a_{ij} < b_i - \sum_{j \in A} a_{ij} \leq b_i(A).$$

Note that $b_i(A) > 0$ for any $i \in M(A)$. By dividing both sides by $b_i(A)$ and taking sum of $i \in M(\tilde{S})$, we get

$$\sum_{i \in M(A) \cap M(\tilde{S})} \sum_{j \in \tilde{S} \setminus A} \frac{a_{ij}(A)}{b_i(A)} < |M(\tilde{S})|, \tag{3.17}$$

where we use $M(\tilde{S}) = M(A) \cap M(\tilde{S})$ from (3.15). From the definition of f and $a_{ij}(A) \leq b_i(A)$, we have that for any $i \in M(A)$,

$$\sum_{j \in S \setminus A} a_{ij}(A) \leq \sum_{j \in S} a_{ij}(A) \leq f b_i(A).$$

By dividing both sides by $b_i(A)$ and taking sum of $i \in M(A)\backslash M(\tilde{S})$, we get

$$\sum_{i\in M(A)\backslash M(\tilde{S})} \sum_{j\in\bar{S}\backslash A} \frac{a_{ij}(A)}{b_i(A)} \leq f(|M(A)| - |M(\tilde{S})|). \tag{3.18}$$

From (3.17) and (3.18), we obtain that

$$\begin{aligned}
\sum_{j\in\bar{S}\backslash A} U_j(A) &= \sum_{i\in M(A)} \sum_{j\in\bar{S}\backslash A} \frac{a_{ij}(A)}{b_i(A)} \\
&= \sum_{i\in M(A)\cap M(\tilde{S})} \sum_{j\in\bar{S}\backslash A} \frac{a_{ij}(A)}{b_i(A)} + \sum_{i\in M(A)\backslash M(\tilde{S})} \sum_{j\in\bar{S}\backslash A} \frac{a_{ij}(A)}{b_i(A)} \\
&< |M(\tilde{S})| + f(|M(A)| - |M(\tilde{S})|) \\
&= (1-f)|M(\tilde{S})| + f|M(A)|.
\end{aligned}$$

Since $\tilde{\boldsymbol{x}}$ is infeasible, $1 \leq |M(\tilde{S})|$ holds. Also, $1 \leq M(A) \leq m$ holds. From $f \geq 2$, we finally obtain that

$$\begin{aligned}
\sum_{j\in\bar{S}\backslash A} U_j(A) &\leq (1-f)|M(\tilde{S})| + f|M(A)| \\
&\leq 1 - f + f|M(A)| \\
&= \left(f - \frac{f-1}{|M(A)|}\right)|M(A)| \\
&\leq \left(f - \frac{f-1}{m}\right)|M(A)|.
\end{aligned}$$

□

Now we easily prove Lemma 3.2.1.

Proof of Lemma 3.2.1. From Lemma 3.3.3, we have that

$$\sum_{j\in N} c_j x_j = \sum_{j\in N} c_j \tilde{x}_j + c_\ell \leq \left(f - \frac{f-1}{m}\right) OPT(I) + c_{\max}.$$

Fujito [21] shows that the algorithm PD runs in $O(mn^2)$ time. □

Chapter 4

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

4.1 Overview

In this chapter, we study the partial covering 0–1 integer program (PCIP), which is a partial version of the covering 0-1 integer program (CIP). PCIP is studied as a model more robust to outliers than CIP (see Example 4 in Chapter 1). For fixed number $p \in \{0, 1, \ldots, m\}$, PCIP is formulated as follows:

$$\text{PCIP} \left| \begin{array}{lll} \min & \displaystyle\sum_{j \in N} c_j x_j & \\ \text{s.t.} & \displaystyle\sum_{j \in N} a_{ij} x_j + b_i z_i \geq b_i, & \forall i \in M, \\ & \displaystyle\sum_{i \in M} z_i \leq p, & \\ & x_j \in \{0, 1\}, & \forall j \in N, \\ & z_i \in \{0, 1\}, & \forall i \in M, \end{array} \right. \tag{4.1}$$

where $M = \{1, \ldots, m\}$, $N = \{1, \ldots, n\}$, $c_j \geq 0$ ($j \in N$), $a_{ij} \geq 0$ ($i \in M$, $j \in N$) and $b_i > 0$ ($i \in M$).

PCIP generalizes some important problems for which approximation algo-

rithms are proposed as shown in Table 4.1, where

$$\begin{aligned} f &= \max_{i\in M} |\{j \in N \mid a_{ij} > 0\}|, \\ \Delta &= \max_{i\in N} |\{i \in M \mid a_{ij} > 0\}|, \\ H(\Delta) &= 1 + \tfrac{1}{2} + \cdots + \tfrac{1}{\Delta}, \\ b_{\max} &= \max_{i\in M} b_i, \\ b_{\min} &= \min_{i\in M} b_i, \\ \eta &= \Delta \frac{\max_{j\in N} c_j}{\min_{j\in N} c_j} \frac{b_{\max}}{b_{\min}}, \\ \gamma &= \tfrac{m}{m-p\eta}, \\ g &= \max\left\{\tfrac{\Delta}{m-p}\left(\tfrac{1}{f-b_{\max}} + \tfrac{b_{\max}}{b_{\min}}\right), \tfrac{f}{b_{\min}} + \left(1 - \tfrac{1}{b_{\max}}\right)p, p+1\right\}. \end{aligned} \tag{4.2}$$

Table 4.1: Special cases in PCIP

Problems	Restrictions in PCIP	Approximation ratios
PCIP	-	$\cdot \max\{f, p+1\}$ (this thesis)
Covering 0–1 Integer Program (CIP)	$p = 0$	$\cdot f$ [10, 21, 35] $\cdot O(\log m)$ [33]
Partial Set Multi-Covering Problem (PSMCP)	$a_{ij} \in \{0, 1\}$, b_i is a positive integer	$\cdot \gamma H(\Delta)$ [43] $\cdot$ g [42]
Partial Set Covering Problem (PSCP)	$a_{ij} \in \{0, 1\}$, $b_i = 1$	$\cdot f$ [6, 23] $\cdot \frac{f\Delta}{f+\Delta-1}$ [21] $\cdot O(\log m)$ [26]

Since PCIP is a generalization of the set covering problem (SCP), one of the research objectives in approximation algorithms for PCIP is to develop an approximation algorithm with approximation ratio close to f or $O(\log m)$ if it is possible. While no approximation algorithms for PCIP are studied, there are good approximation algorithms for a special case for PCIP. A partial set covering problem (PSCP) is a special case of PCIP and a partial version of SCP, where at most p elements can not be covered. Note that PSCP is a special case of PCIP where $a_{ij} \in \{0, 1\}$ and $b_i = 1$ for $i \in M$ and $j \in N$. For PSCP, there are approximation algorithms with approximation ratios f [23] and $O(\log m)$ [19]. Such algorithms are not presented for PCIP which is a generalization of PSCP and CIP. Recently. some approximation algorithms are presented for the partial set multi-covering problem [43, 42], which is a special case of PCIP where $a_{ij} \in \{0, 1\}$ and b_i is a positive integer for $i \in M$ and $j \in N$. There are applications of PSMCP such as analysis of influence in social networks [42, 43] and protein identification [26]. Ran et

al. [43] 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. [42] propose an approximation algorithm with performance ratio g defined in (4.2). However, their approximation ratios have a gap with the best approximation ratios for SCP since they depend on some coefficients.

Contribution

We present an α-approximation algorithm for PCIP by an approximation algorithm for CIP shown in Chapter 2, where

$$\alpha = \max\{f, p + 1\}. \tag{4.3}$$

This approximation ratio achieves the same approximation ratio as f possibly best for SCP when $f \geq p + 1$.

The main idea of our algorithm is as follows. A natural generalization of the algorithm for CIP shown in Chapter 2 will no give approximation guarantee. Thus, as in Chapter 3, our algorithm uses on an enumeration technique for PSCP by Gandhi et al [23]. First we develop an primal-dual algorithm for PCIP, which is called the subalgorithm. This algorithm is similar with an f-approximation algorithm in Chapter 2. This subalgorithm does not give approximation guarantee but it can give α-approximation if we know partial information about an optimal solution. Then we propose an algorithm, called main algorithm, where we can get α-approximation solution without information about an optimal solution by solving $O(n)$ subproblems with the subalgorithm.

Assumption and Notation

Without loss of generality, we assume that

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

Let $I = (m, n, \boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}, p)$ be a data of (4.1), where $\boldsymbol{A}$ is the matrix of a_{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 $(\boldsymbol{x}(S), \boldsymbol{z}(S))$ as follows:

$$x_j(S) = \begin{cases} 1 & \text{if } j \in S \\ 0 & \text{if } j \notin S \end{cases} \quad \text{for any } j \in N \tag{4.4}$$

and

$$z_i(S) = \begin{cases} 1 & \text{if } \sum_{j \in S} a_{ij} < b_i \\ 0 & \text{if } \sum_{j \in S} a_{ij} \geq b_i. \end{cases} \quad \text{for any } i \in M. \tag{4.5}$$

This solution always satisfies the constraints in (4.1) except for $\sum_{i \in M} z_i \leq p$. Hence $(\boldsymbol{x}(S), \boldsymbol{z}(S))$ is feasible to (4.1) if and only if $\sum_{i \in M} z_i(S) \leq p$.

4.2 Main algorithm

Our algorithm is an extension of an f-approximation algorithm for PSCP by Gandhi et al. [23] 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 4.2.1) of the solution generated by the subalgorithm.
2. We explain that we get an α-approximation solution by using the subalgorithm if we know partial information about an optimal solution.
3. We present 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 $(\boldsymbol{x}(\tilde{S}), \boldsymbol{z}(\tilde{S}))$ is feasible and has the following property in Lemma 4.2.1. The algorithm and the proof of Lemma 4.2.1 are shown in Section 3.

Lemma 4.2.1. *The subalgorithm presented in Section 3 outputs $\tilde{S} \subseteq N$ such that the solution $(\boldsymbol{x}(\tilde{S}), \boldsymbol{z}(\tilde{S}))$ defined by (4.4) and (4.5) 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 $I = (m, n, \boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}, p)$ and $h \in \{2, \ldots, n\}$, we consider a sub-problem of PCIP(I), where we add the following constraints to PCIP(I):

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

This sub-problem can be expressed as:

$$\begin{aligned} \min \quad & \sum_{j \in \{1,\ldots,h-1\}} c_j x_j \\ \text{s.t.} \quad & \sum_{j \in \{1,\ldots,h-1\}} a_{ij} x_j + b_i z_i \geq b_i - a_{ih}, \quad \forall i \in M = \{1, \ldots, m\}, \\ & \sum_{i \in M} z_i \leq p, \\ & x_j \in \{0,1\}, \qquad \forall j \in \{1, \ldots, h-1\}, \\ & z_i \in \{0,1\}, \qquad \forall i \in M. \end{aligned} \tag{4.6}$$

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

$$I(h) = (m, h-1, \boldsymbol{A}(h), \boldsymbol{b}(h), \boldsymbol{c}(h), p), \tag{4.7}$$

where

$$\begin{aligned} \boldsymbol{A}(h) &= (\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{h-1}), \\ \boldsymbol{b}(h) &= \boldsymbol{b} - \boldsymbol{a}_h = (b_1 - a_{1h}, \ldots, b_m - a_{mh})^T, \\ \boldsymbol{c}(h) &= (c_1, \ldots, c_{h-1})^T. \end{aligned}$$

Let S^* be the subset of N such that $(\boldsymbol{x}(S^*), \boldsymbol{z}(S^*))$ is an optimal solution of PCIP(I). From the assumption that $b_i > 0$ 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 $(\boldsymbol{x}(\{h^*\}), \boldsymbol{z}(\{h^*\}))$. In the following discussion, we assume $h^* \geq 2$.

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

Lemma 4.2.2. *Let $\tilde{S}(h)$ be the output by the subalgorithm for the sub-problem PCIP($I(h)$) which is defined by (4.6). Define $S(h) = \tilde{S}(h) \cup \{h\}$. If $h = h^*$, $S(h^*)$ gives a feasible α-approxiamtion solution for PCIP(I), that is, $(\boldsymbol{x}(S(h^*)), \boldsymbol{z}(S(h^*)))$ 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. $(\boldsymbol{x}(S(h^*)), \boldsymbol{z}(S(h^*)))$ is feasible to PCIP(I) since $(\boldsymbol{x}(\tilde{S}(h^*)), \boldsymbol{z}(\tilde{S}(h^*)))$ is feasible to PCIP($I(h^*)$) from Lemma 4.2.1.

We have that $\alpha \geq 2$ since $\alpha = \max\{f, p+1\}$ and $f \geq 2$. From Lemma 4.2.1, $c_{h^*} \geq c_{h^*-1}$ 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 4.2.2 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: $I = (m, n, \boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}, p)$.

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 (4.7). 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 $(\boldsymbol{x}(S(\hat{h})), \boldsymbol{z}(S(\hat{h})))$

Theorem 4.2.1. *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 4.2.1 and the main algorithm executes the subalgorithm at most n times. Therefore the main algorithm is a polynomial time algorithm.

$(\boldsymbol{x}(S(\hat{h})), \boldsymbol{z}(S(\hat{h})))$ is clearly feasible to PCIP(I) and from Lemma 4.2.2 we obtain that

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

□

4.3 Subalgorithm

In this section, we show the subalgorithm and prove Lemma 4.2.1. The subalgorithm is based on a 2-approximation algorithm for the minimum knapsack problem by Carnes and Shmoys [9] and its extension to CIP by Takazawa and Mizuno [46]. Both of the algorithms use an LP relaxation of CIP proposed by Carr et al. [10]. We apply this relaxation to PCIP and we have the following problem:

$$\begin{array}{lll} \min & \displaystyle\sum_{j\in N} c_j x_j & \\ \text{s.t.} & \displaystyle\sum_{j\in N\setminus A} a_{ij}(A)x_j + b_i(A)z_i \geq b_i(A), & \forall i \in M, \forall A \subseteq N \\ & \displaystyle\sum_{i\in M} z_i \leq p, & \\ & x_j \geq 0, & \forall j \in N, \\ & z_i \geq 0, & \forall i \in M, \end{array} \tag{4.8}$$

where

$$\begin{array}{rcl} b_i(A) & = & \max\{0, b_i - \sum_{j\in A} a_{ij}\},\ \forall i \in M, \forall A \subseteq N, \\ a_{ij}(A) & = & \min\{a_{ij}, b_i(A)\},\ \forall i \in M, \forall A \subseteq N, \forall j \in N\setminus A. \end{array} \tag{4.9}$$

Lemma 4.3.1. *(4.8) is a relaxation problem of PCIP, that is, any feasible solution $(\boldsymbol{x}, \boldsymbol{z})$ for PCIP is feasible to (4.8).*

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

$$\sum_{j\in S\setminus A} a_{ij}(A) \geq b_i(A). \tag{4.10}$$

Fix $i \in M_0$ and $A \subseteq N$ such that $b_i(A) > 0$, that is, $b_i(A) = b_i - \sum_{j\in A} a_{ij}$. We consider the case when there is an index $j \in S\setminus A$ such that $a_{ij}(A) = b_i(A)$. In this case, (4.10) is clearly satisfied. Next, we consider the case when $a_{ij}(A) < b_i(A)$, that is, $a_{ij}(A) = a_{ij}$ for all $j \in S\setminus A$. In this case, we have that

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

□

The dual of (4.8) is expressed as

$$\begin{aligned}
\max \quad & \sum_{i\in M}\sum_{A\subseteq N} b_i(A)y_i(A) - pz \\
\text{s.t.} \quad & \sum_{i\in M}\sum_{A\subseteq N:\, j\notin A} a_{ij}(A)y_i(A) \le c_j, && \forall j \in N, \\
& \sum_{A\subseteq N} b_i(A)y_i(A) \le z, && \forall i \in M, \\
& y_i(A) \ge 0, && \forall A \subseteq N,\ \forall i \in M, \\
& z \ge 0.
\end{aligned} \tag{4.11}$$

Now, we introduce a useful result for later discussion.

Lemma 4.3.2. *Let S be a subset of N such that $(\boldsymbol{x}(S), \boldsymbol{z}(S))$ is infeasible to PCIP(I), $(\boldsymbol{y}, z)$ be a feasible solution to (4.11). Define $M_1(S) = \{i \in M \mid z_i(S) = 1\}$. If*

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

then the following inequalities hold:

$$\sum_{j\in N} c_j x_j(S) \le \alpha\left(\sum_{i\in M}\sum_{A\subseteq N} b_i(A)y_i(A) - pz\right) \le \alpha OPT(I). \tag{4.12}$$

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

$$\sum_{j\in S\setminus A} a_{ij}(A) \le \sum_{j\in S} a_{ij}(A) \le f b_i(A) \le \alpha b_i(A) \tag{4.13}$$

from the definition of f and $a_{ij}(A) \le b_i(A)$ by (4.9). Since $(\boldsymbol{x}(S), \boldsymbol{z}(S))$ is infeasible, the following inequality holds:

$$|M_1(S)| \ge p + 1. \tag{4.14}$$

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

$$\sum_{j\in N} c_j x_j(S) = \sum_{j\in S} c_j = \sum_{j\in S}\sum_{i\in M}\sum_{A\subseteq N:\, j\notin A} a_{ij}(A)y_i(A). \tag{4.15}$$

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

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

where the last inequality holds from (4.13) and (b). Hence we have that

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

Taking the difference between two values in (4.12),

$$\begin{aligned}
& \alpha \left(\sum_{i \in M} \sum_{A \subseteq N} b_i(A) y_i(A) - pz \right) - \sum_{j \in N} c_j x_j(S) \\
\geq & (\alpha - 1) \sum_{i \in M_1(S)} \sum_{A \subseteq N} b_i(A) y_i(A) - \alpha p z \\
= & (\alpha - 1) |M_1(S)| z - \alpha p z \qquad (4.16) \\
\geq & (\alpha - (p+1)) z \geq 0, \qquad (4.17)
\end{aligned}$$

where the equality (4.16) follows from (a-2) and inequalities (4.17) follow from (4.14) and (4.3). Since $(\boldsymbol{y}, z)$ is feasible to (4.11), the objective value of $(\boldsymbol{y}, z)$ is less than or equal to the optimal value of (4.8), which is less than or equal to OPT(I). Thus we have that

$$\alpha \left(\sum_{i \in M} \sum_{A \subseteq N} b_i(A) y_i(A) - pz \right) \leq \alpha OPT(I). \qquad (4.18)$$

□

The subalgorithm is presented below. Solutions generated by the algorithm, except for the final solution, satisfy all the conditions in Lemma 4.3.2. In the subalgorithm, we use the following symbols:

- a set $S \subseteq N$.
- a solution $(\boldsymbol{y}, z)$ for (4.11).
- $M_1(S) = \{i \in M | \sum_{j \in S} a_{ij} < b_i\}$.
- $N'(S) = \{j \in N \backslash S | \sum_{i \in M_1(S)} a_{ij}(S) > 0\}$.
- $\forall j \in N,\ \bar{c}_j = c_j - \sum_{i \in M} \sum_{A \subseteq N : j \notin A} a_{ij}(A) y_i(A)$.

The subalgorithm

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

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

Step 1: Calculate $b_i(S)$ by (4.9) for $i \in M$. Update $M_1(S)$. If $|M_1(S)| \leq p$, output $\tilde{S} = S$ and stop. Otherwise, calculate $a_{ij}(S)$ by (4.9) for all $i \in M_1(S)$ and $j \in N$. Update $N'(S)$.

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

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

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

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

where

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

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

Now, let us see the increment of $\sum_{A \subseteq N} b_i(A) y_i(A)$ in this iteration, which is the left side of $\sum_{A \subseteq N} b_i(A) y_i(A) \leq z$ in (4.11). Note that $\sum_{A \subseteq N} b_i(A) y_i(A)$ 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 $b_i(S) y_i(S)$ and we have that

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

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

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

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

$$\sum_{A \subseteq N} b_{i'}(A) y_{i'}(A) \leq \sum_{A \subseteq N} b_i(A) y_i(A)$$

Update

$$z \coloneqq z + \delta_s.$$

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

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

Update $S \coloneqq S \cup \{s\}$ 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 4.3.2.

Lemma 4.3.3. *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 4.3.2.*

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

feasibility: The subalgorithm stops if $|M_i(S)| \leq p$ which implies that $(\boldsymbol{x}(S), \boldsymbol{z}(S))$ is a feasible solution. Thus, $(\boldsymbol{x}(S'), \boldsymbol{z}(S'))$ is infeasible to PCIP(I). $(\boldsymbol{y}, z)$ is feasible to the dual (4.11) 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

$$\sum_{j \in S'} a_{ij} < b_i.$$

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

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

where the first and last inequalities follow from (4.9).

□

Now we easily prove Lemma 4.2.1.

Proof of Lemma 4.2.1. $(\boldsymbol{x}(\tilde{S}), \boldsymbol{z}(\tilde{S}))$ is clearly feasible and from Lemma 4.3.2 and Lemma 4.3.3, 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 us take a look at the running time of the subalgorithm. In the algorithm, the most time consuming parts per iteration are the calculations of $a_{ij}(S)$ 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 $O(mn^2)$.

□

Chapter 5

Approximation algorithms for the partial covering linear program

5.1 Overview

The covering LP is a special case of LP and formulated as $\{\min \boldsymbol{c}^T\boldsymbol{x} \mid \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b}, \boldsymbol{x} \geq \boldsymbol{0}\}$, where $\boldsymbol{A} \in \mathbb{R}^{m\times n}, \boldsymbol{b} \in \mathbb{R}^m$, $\boldsymbol{c} \in \mathbb{R}^n$ and all the data are nonnegative. The covering LP often appears in practice and thus efficient algorithms are well studied [4]. In this chapter, we study approximation algorithms for the partial covering LP (PCLP), where at most p covering constraints are not satisfied for given integer p. PCLP can be regarded as an LP version of the partial covering 0-1 integer program studied in Chapter 4. PCLP is formulated as a mixed-integer program (MIP) as follows:

$$
\begin{aligned}
\min \quad & \sum_{j\in N} c_j x_j & & \\
\text{s.t.} \quad & \sum_{j\in N} a_{ij}x_j + b_i z_i \geq b_i & & \forall i \in M = \{1,\dots,m\}, \\
& \sum_{i\in M} z_i \leq p, & & \\
& x_j \geq 0 & & \forall j \in N = \{1,\dots,n\}, \\
& z_i \in \{0,1\} & & \forall i \in M.
\end{aligned}
$$

where $c_j \geq 0$ $(j \in N)$, $a_{ij} \geq 0$ $(i \in M, j \in N)$, $b_i \geq 0$ $(i \in M)$ and $k \in \{0,1,\dots,m-1\}$. Without loss of generality, assume $b_i > 0$ for $i \in I$ since a inequality, $\sum_{i\in M} a_{ij}x_j + b_i z_i \geq b_i$, is redundant when $b_i = 0$. Thus, we consider the following problem, where b_i is scaled to 1.

$$
\begin{aligned}
\min \quad & \sum_{j \in N} c_j x_j \\
\text{s.t.} \quad & \sum_{j \in N} a_{ij} x_j + z_i \geq 1 && \forall i \in M = \{1, \ldots, m\}, \\
& \sum_{i \in M} z_i \leq p, \\
& x_j \geq 0 && \forall j \in N = \{1, \ldots, n\}, \\
& z_i \in \{0, 1\} && \forall i \in M.
\end{aligned}
\tag{5.1}
$$

When p or $m - p$ are constant, we get an optimal solution in polynomial time. In general case, Qiu et al. [41] show that PCLP is strongly NP-hard. They also show that it is hard to solve some realistic instances by CPLEX and they develop an efficient exact algorithm based on MIP. Dinitz and Gupta [16] present two approximation algorithms which are based on a reduction to a problem similar to the set cover problem and use a Lagrangian relaxation approach. Their approximation ratios are $O(f \log f)$ and $O(\log \Delta \log f)$ where $f = \max_{i \in M} |\{j \in N \mid a_{ij} > 0\}|$ and $\Delta = \max_{j \in N} |\{i \in M \mid a_{ij} > 0\}|$.

Contributions

In this chapter, we present an LP rounding $(p + 1)$-approximation algorithm for PCLP. We also show that the integerality gap of the LP is $p + 1$. This implies we can not get a better approximation algorithm as long as we use an LP-relaxation as a lower bound of the optimal value [51]. We also present a simple $(m - p)$-approximation algorithm.

Recently and independently with our research, Ahmed and Xie [1] present a $(p + 1)$-approximation algorithm based on bisection search for a more general problem than PCLP.

5.2 A $(p + 1)$-approximation algorithm

In this section, we present a $(p+1)$-approximation algorithm for CVKLP based on a natural LP relaxation and show this approximation ratio matches the integrality

gap of the LP. A straightforward LP relaxation of (5.1) is expressed as follows.

$$\begin{array}{lll} \min & \displaystyle\sum_{j\in N} c_j x_j & \\ \text{s.t.} & \displaystyle\sum_{j\in N} a_{ij}x_j + z_i \geq 1 & \forall i \in M, \\ & \displaystyle\sum_{i\in M} z_i \leq k, & \\ & x_j \geq 0 & \forall j \in N = \{1,\ldots,n\}, \\ & 1 \geq z_i \geq 0 & \forall i \in M = \{1,\ldots,m\}. \end{array} \tag{5.2}$$

In our algorithm, we first solve the LP relaxation problem and obtain an optimal LP solution $(\boldsymbol{x}^L, \boldsymbol{z}^L)$. Then, by rounding and scaling the LP solution $(\boldsymbol{x}^L, \boldsymbol{z}^L)$, we obtain a MIP solution $(\boldsymbol{x}^M, \boldsymbol{z}^M)$. The algorithm is presented as follows.

Algorithm 1

Step 1: Solve the LP relaxation problem (5.2) and let an optimal solution for (5.2) be $(\boldsymbol{x}^L, \boldsymbol{z}^L)$.

Step 2: Let $(i_1, \ldots, i_m)$ be a permutation of M such that

$$z^L_{i_1} \geq z^L_{i_2} \geq \ldots \geq z^L_{i_m}$$

and set

$$\delta = \frac{1}{1 - z^L_{i_{p+1}}}.$$

Note that $z^L_{i_{p+1}}$ is less than 1 from the proof of Theorem 1 shown later.

Step 3: Set the solution $(\boldsymbol{x}^M, \boldsymbol{z}^M)$ as follows.

$$\boldsymbol{x}^M = \delta \boldsymbol{x}^L$$

and

$$z^M_i = \begin{cases} 1 & \text{if } i \in \{i_1 \ldots i_p\}, \\ 0 & \text{if } i \in \{i_{p+1} \ldots i_m\}. \end{cases}$$

Output $(\boldsymbol{x}^M, \boldsymbol{z}^M)$.

We show that the output of the algorithm is feasible to (5.1).

Lemma 5.2.1. *$(\boldsymbol{x}^M, \boldsymbol{z}^M)$ is a feasible solution of* (5.1)*.*

Proof. It suffices to show $\sum_{j\in N} a_{ij}x_j^M \geq 1$ for all $i \in \{i_{p+1} \dots i_m\}$. Fix any $i \in \{i_{p+1} \dots i_m\}$. If $z_i^L = 0$, then we have that

$$\sum_{j\in N} a_{ij}x_j^M = \delta \sum_{j\in N} a_{ij}x_j^L \geq \delta.$$

Let us consider the case where $z_i^L > 0$. From Step 2 we obtain that

$$\sum_{j\in N} a_{ij}x_j^M = \delta \sum_{j\in N} a_{ij}x_j^L = \frac{1}{1 - z_{i_{p+1}}^L} \sum_{j\in N} a_{ij}x_j^L.$$

Since $(\boldsymbol{x}^L, \boldsymbol{z}^L)$ is a feasible solution of (5.2), $\sum_{j\in N} a_{ij}x_j^L \geq 1 - z_i^L$ holds. Therefore we get

$$\sum_{j\in N} a_{ij}x_j^M = \frac{1}{1 - z_{i_{p+1}}^L} \sum_{j\in N} a_{ij}x_j^L \geq \frac{1 - z_i^L}{1 - z_{i_{p+1}}^L} \geq 1.$$

□ □

Then, we show that $(\boldsymbol{x}^M, \boldsymbol{z}^M)$ is a $(p + 1)$-approximation solution of (5.1).

Theorem 5.2.1. $(\boldsymbol{x}^M, \boldsymbol{z}^M)$ *is a feasible solution of* (5.1) *such that*

$$\sum_{j\in N} c_j x_j^M \leq \delta OPT \leq (p + 1)OPT,$$

where OPT is the optimal value of (5.1).

Proof. Let OPT_{LP} be the optimal value of (5.2). Then we have that

$$\sum_{j\in N} c_j x_j^M = \delta OPT_{LP} \leq \delta OPT.$$

Hence, it suffices to show that $\delta \leq p+1$. Suppose that $z_{i_{p+1}}^L > \frac{k}{p+1}$. Then, we obtain that

$$\sum_{i=1}^{m} z_i^L \geq \sum_{\ell=1}^{p+1} z_{i_\ell}^L > (p + 1)\frac{k}{p + 1} = k$$

since $z_{i_1}^L \geq \dots \geq z_{i_{p+1}}^L$ holds from Step 2. Then $(\boldsymbol{x}^L, \boldsymbol{z}^L)$ is infeasible to (5.2) and this is a contradiction. Therefore, we have that $z_{i_{p+1}}^L \leq \frac{k}{p+1}$ and

$$\delta = \frac{1}{1 - z_{i_{p+1}}^L} \leq \frac{1}{1 - \frac{k}{p+1}} = p + 1.$$

□ □

Next, we show that the integrality gap of the LP relaxation (5.2) is $p + 1$. The integrality gap of the relaxation problem (5.2) is defined as the supremum of the ratio of the optimal values of (5.1) and (5.2) for all instances [51] . From the proof of Theorem 1, we have that

$$OPT \leq \sum_{j \in N} c_j x_j^M \leq (p+1) OPT_{LP}.$$

Therefore, we say that the integrality gap is bounded above by $p + 1$. Then, we show that this bound is tight by introducing the following instance.

$$\begin{array}{lll} \min & x_1 & \\ \text{s.t.} & x_1 + z_i \geq 1 & \forall i \in \{1, \ldots, p+1\}, \\ & (p+1)x_1 + z_i \geq 1 & \forall i \in \{p+2, \ldots, m\}, \\ & \sum_{i=1}^{m} z_i \leq p & \\ & x_1 \geq 0, & \\ & z_i \in \{0, 1\} & \forall i \in \{1, \ldots, m\}, \end{array} \tag{5.3}$$

where $m \geq 2$ and $k \in \{1, \ldots, m-1\}$. When we set x_1 to be less than 1, we can not satisfy $m - p$ constraints for any $\boldsymbol{z}$. Hence, an optimal solution of this problem is $(x_1^*, \boldsymbol{z}^*) = (1, \boldsymbol{0})$. The objective value of $(x_1^*, \boldsymbol{z}^*)$ is 1. Now consider an LP relaxation of (5.3) and a feasible solution $(x_1^L, \boldsymbol{z}^L)$ of the LP such that $x_1^L = \frac{1}{p+1}$ and

$$z_i^L = \begin{cases} \frac{k}{p+1} & \text{if } i \in \{1, \ldots, p+1\}, \\ 0 & \text{if } i \in \{p+2, \ldots, m\}. \end{cases}$$

The objective value of $(x_1^L, \boldsymbol{z}^L)$ is $\frac{1}{p+1}$. Thus, the optimal value of (5.3) is at least $p + 1$ times that of its LP relaxation. From this observation and Theorem 1, the integrality gap of the problem (5.2) is $p + 1$.

5.3 An $(m - p)$-approximation algorithm

The approximation ratio of the algorithm is an increasing function of p. However, since our approximation ratio matches the integrality gap, in order to get a better approximation algorithm, we need a strong LP relaxation of PCLP or other approaches. On the other hand, when p is close to m, we easily get a better approximation algorithm, that is, an $(m - p)$-approximation algorithm as follows.

Algorithm 2

Step 1: For all $i \in M$ solve the problem, $\min \sum_{j \in N} c_j x_j$ such that $\sum_{j \in N} a_{ij} x_j \geq 1$ and $x_j \geq 0$ ($j \in N$). Let $\boldsymbol{x}^i$ be an optimal solution and $d_i = \sum_{j \in N} c_j x_j^i$ be the otpimal value.

Step 2: Let $(i_1, \ldots, i_m)$ be a permutation of M such that

$$d_{i_1} \leq d_{i_2} \leq \ldots \leq d_{i_m}.$$

Step 3: Set the solution $(\boldsymbol{x}^M, \boldsymbol{z}^M)$ as follows.

$$\boldsymbol{x}^M = \sum_{\ell=1}^{m-p} \boldsymbol{x}^{i_\ell}$$

and

$$z_i^M = \begin{cases} 0 & \text{if } i \in \{i_1 \ldots i_{m-p}\}, \\ 1 & \text{if } i \in \{i_{m-p+1} \ldots i_m\}. \end{cases}$$

Output $(\boldsymbol{x}^M, \boldsymbol{z}^M)$.

We easily show that the algorithm is an $(m - p)$-approximation algorithm.

Theorem 5.3.1. *Algorithm 2 is an $(m - p)$-approximation algorithm PCLP.*

Proof. Let $(\boldsymbol{x}^M, \boldsymbol{z}^M)$ be the output of Algorithm 2. Then, $(\boldsymbol{x}^M, \boldsymbol{z}^M)$ is an feasible solution of (5.1) since it clearly satisfies $m-p$ constraints. Suppose $d_{i_{m-p}}$ is greater than the optimal value of (5.1) ($= OPT$). Then, an optimal solution of (5.1) satisfies at least $m - p$ constraints but the objective value is less than $d_{i_{m-p}}$. This is the contradiction with the definition of $d_{i_{m-p}}$. We have that

$$\sum_{j \in N} c_j x_j^M = \sum_{\ell=1}^{m-p} d_{i_\ell} \leq (m - p) d_{i_{m-p}} \leq (m - p) OPT.$$

□

Chapter 6

Conclusion

Covering problems are NP-hard optimization problems with covering constraints. Even though LP-based approximation algorithms are known to be particularly effective for simple covering problems, it is not easy to apply LP-based methods to more general covering problems.

This thesis is devoted to develop better LP-based approximation algorithms than the existing ones for four covering problems, which are studied from practical requirements: (1) the minimum knapsack problem with forcing constraints (MKPFC), (2) the covering 0-1 integer program (CIP), (3) the partial covering 0-1 integer program (PCIP), (4) the partial covering linear program (PCLP).

(1) MKPFC is a generalization of the two fundamental covering problems, the minimum knapsack problem (MKP) and the vertex cover problem. For MKPFC, we propose a 2-approximation algorithm for the minimum knapsack problem with a forcing graph. The approximation ratio of the algorithm is the same as that of the algorithms for the minimum knapsack problem by Carnes and Shmoys[9] and for the minimum vertex cover problem by Bar-Yehuda and Even [7]. Then we generalize the algorithm to CIP and propose a f_2-approximation algorithm, where f_2 is the second largest number of non-zero coefficients in the constraints. The proposed algorithm overcomes the shortcoming of the existing algorithm that the approximation ratio gets worse if one constraint is dense in CIP even though the other constraints are sparse.

(2) CIP is a natural generalization of the set covering problem. For CIP, several f-approximation algorithms exists where f is the maximum number of non-zero entries in the constraints. We propose an $\left(f - \frac{f-1}{m}\right)$-approximation algorithm for CIP when $m \geq 2$, where m is the number of constraints. This is the first algorithm for CIP whose approximation ratio is strictly less than f when $f \geq 2$. Our algo-

rithm solves subproblems of CIP by using an f-approximation algorithm for CIP by Fujito [21] as a subroutine.

(3) PCIP is a generalization of CIP, where a fixed number p of the constraints can be violated. For some special cases of PCIP, approximation algorithms are presented so far. However their approximation ratios may get worse even when the size of problem is small. In this thesis, we propose a $\max\{f, p+1\}$-approximation algorithm for PCIP by generalizing our algorithm for CIP in Chapter 2 and an algorithm for the partial set covering problem by Gandhi et al. [23].

(4) PCLP is an LP version of PCIP, where a binary variables are relaxed to be continuous. For PCLP, we give a $(p+1)$-approximation algorithm. Our algorithm is a simple rounding algorithm based on a natural LP relaxation. We also show that we can not get better approximation algorithms when we use the LP-relaxation as a lower bound of the optimal value.

Bibliography

[1] Shabbir Ahmed and Weijun Xie. Relaxations and approximations of chance constraints under finite distributions. *Mathematical Programming*, 170(1):43–65, 2018.

[2] Laurent Alfandari and Anass Nagih. Airline crew pairing optimization. *Applications of Combinatorial Optimization*, pages 1–22, 2014.

[3] Laurent Alfandari, J Sadki, Agnès Plateau, and Anass Nagih. Hybrid column generation for large-size covering integer programs: Application to transportation planning. *Computers & Operations Research*, 40(8):1938–1946, 2013.

[4] Zeyuan Allen-Zhu and Lorenzo Orecchia. Nearly linear-time packing and covering LP solvers. *Mathematical Programming*, 175(1-2):307–353, 2019.

[5] Egon Balas and Maria C Carrera. A dynamic subgradient-based branch-and-bound procedure for set covering. *Operations Research*, 44(6):875–890, 1996.

[6] Reuven Bar-Yehuda. Using homogeneous weights for approximating the partial cover problem. *Journal of Algorithms*, 39(2):137–144, 2001.

[7] Reuven Bar-Yehuda and Shimon Even. A linear-time approximation algorithm for the weighted vertex cover problem. *Journal of Algorithms*, 2(2):198–203, 1981.

[8] Alberto Caprara, Matteo Fischetti, and Paolo Toth. A heuristic method for the set covering problem. *Operations research*, 47(5):730–743, 1999.

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

[10] Robert D Carr, Lisa K Fleischer, Vitus J Leung, and Cynthia A Phillips. Strengthening integrality gaps for capacitated network design and covering problems. In *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, pages 106–115. Society for Industrial and Applied Mathematics, 2000.

[11] Timothy M Chan. Low-dimensional linear programming with violations. *SIAM Journal on Computing*, 34(4):879–893, 2005.

[12] Moses Charikar, Samir Khuller, David M Mount, and Giri Narasimhan. Algorithms for facility location problems with outliers. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 642–651. Society for Industrial and Applied Mathematics, 2001.

[13] Ke Chen. A constant factor approximation algorithm for k-median clustering with outliers. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, volume 8, pages 826–835, 2008.

[14] Andreas Darmann, Ulrich Pferschy, Joachim Schauer, and Gerhard J Woeginger. Paths, trees and matchings under disjunctive constraints. *Discrete Applied Mathematics*, 159(16):1726–1735, 2011.

[15] Mark S Daskin. *Network and Discrete Location: Models, Algorithms, and Applications*. John Wiley & Sons, 2011.

[16] Michael Dinitz and Anupam Gupta. Packing interdiction and partial covering problems. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 157–168. Springer, 2013.

[17] Irit Dinur and Samuel Safra. On the hardness of approximating vertex cover. *Annals of Mathematics*, 162(1):439–485, 2005.

[18] Ding-Zhu Du, Ker-I Ko, and Xiaodong Hu. *Design and analysis of approximation algorithms*, volume 62. Springer Science & Business Media, 2011.

[19] Tapio Elomaa and Jussi Kujala. Covering analysis of the greedy algorithm for partial cover. In *Algorithms and Applications*, pages 102–113. Springer, 2010.

[20] Reza Zanjirani Farahani, Nasrin Asgari, Nooshin Heidari, Mahtab Hosseininia, and Mark Goh. Covering problems in facility location: A review. *Computers & Industrial Engineering*, 62(1):368–407, 2012.

[21] Toshihiro Fujito. On combinatorial approximation of covering 0-1 integer programs and partial set cover. *Journal of Combinatorial Optimization*, 8(4):439–452, 2004.

[22] Toshihiro Fujito and Takatoshi Yabuta. Submodular integer cover and its application to production planning. In *International Workshop on Approximation and Online Algorithms*, pages 154–166. Springer, 2004.

[23] Rajiv Gandhi, Samir Khuller, and Aravind Srinivasan. Approximation algorithms for partial covering problems. *Journal of Algorithms*, 53(1):55–84, 2004.

[24] Paul C Gilmore and Ralph E Gomory. A linear programming approach to the cutting-stock problem. *Operations Research*, 9(6):849–859, 1961.

[25] Eran Halperin. Improved approximation algorithms for the vertex cover problem in graphs and hypergraphs. *SIAM Journal on Computing*, 31(5):1608–1623, 2002.

[26] Zengyou He, Can Yang, and Weichuan 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):368–380, 2011.

[27] Mhand Hifi and Nabil Otmani. An algorithm for the disjunctively constrained knapsack problem. *International Journal of Operational Research*, 13(1):22–43, 2012.

[28] Mhand Hifi, Sagvan Saleh, and Lei Wu. A fast large neighborhood search for disjunctively constrained knapsack problems. In *International Symposium on Combinatorial Optimization*, pages 396–407. Springer, 2014.

[29] Gerbrich Hoekstra and Frank Phillipson. Multi-service capacitated facility location problem with partial covering in smart cities. In *Conference: 19th International Conference on Innovations for Community Services (I4CS)*, 06 2019.

[30] George Karakostas. A better approximation ratio for the vertex cover problem. *ACM Transactions on Algorithms (TALG)*, 5(4):41, 2009.

[31] Hans Kellerer, Ulrich Pferschy, and David Pisinger. Multidimensional knapsack problems. In *Knapsack Problems*, pages 235–283. Springer, 2004.

[32] Subhash Khot and Oded Regev. Vertex cover might be hard to approximate to within 2- ϵ. *Journal of Computer and System Sciences*, 74(3):335–349, 2008.

[33] Stavros G Kolliopoulos and Neal E Young. Tight approximation results for general covering integer programs. In *Proceedings 2001 IEEE International Conference on Cluster Computing*, pages 522–528. IEEE, 2001.

[34] Stavros G Kolliopoulos and Neal E Young. Approximation algorithms for covering/packing integer programs. *Journal of Computer and System Sciences*, 71(4):495–505, 2005.

[35] Christos Koufogiannakis and Neal E Young. Greedy δ-approximation algorithm for covering with arbitrary constraints and submodular cost. *Algorithmica*, 66(1):113–152, 2013.

[36] S Thomas McCormick, Britta Peis, José Verschae, and Andreas Wierz. Primal–dual algorithms for precedence constrained covering problems. *Algorithmica*, 78(3):771–787, 2017.

[37] Ulrich Pferschy and Joachim Schauer. The knapsack problem with conflict graphs. *Journal of Graph Algorithms and Applications*, 13(2):233–249, 2009.

[38] Ulrich Pferschy and Joachim Schauer. The maximum flow problem with disjunctive constraints. *Journal of Combinatorial Optimization*, 26(1):109–119, 2013.

[39] Ulrich Pferschy and Joachim Schauer. Approximation of knapsack problems with conflict and forcing graphs. *Journal of Combinatorial Optimization*, 33(4):1300–1323, 2017.

[40] David Pritchard and Deeparnab Chakrabarty. Approximability of sparse integer programs. *Algorithmica*, 61(1):75–93, 2011.

[41] Feng Qiu, Shabbir Ahmed, Santanu S Dey, and Laurence A Wolsey. Covering linear programming with violations. *INFORMS Journal on Computing*, 26(3):531–546, 2014.

[42] Yingli Ran, Yishuo Shi, and Zhao Zhang. Local ratio method on partial set multi-cover. *Journal of Combinatorial Optimization*, 34(1):302–313, 2017.

[43] Yingli Ran, Zhao Zhang, Hongwei Du, and Yuqing Zhu. Approximation algorithm for partial positive influence problem in social network. *Journal of Combinatorial Optimization*, 33(2):791–802, 2017.

[44] Ran Raz and Shmuel Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability pcp characterization of np. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 475–484. ACM, 1997.

[45] Thomas Roos and Peter Widmayer. k-violation linear programming. *Information Processing Letters*, 52(2):109–114, 1994.

[46] Yotaro Takazawa and Shinji Mizuno. A 2-approximation algorithm for the minimum knapsack problem with a forcing graph. *Journal of the Operations Research Society of Japan*, 60(1):15–23, 2017.

[47] Yotaro Takazawa, Shinji Mizuno, and Tomonari Kitahara. An approximation algorithm for the partial covering 0–1 integer program. *Discrete applied mathematics*. (in press).

[48] Yotaro Takazawa, Shinji Mizuno, and Tomonari Kitahara. Approximation algorithms for the covering-type k-violation linear program. *Optimization Letters*. (in press).

[49] Yotaro Takazawa, Shinji Mizuno, and Tomonari Kitahara. An improved approximation algorithm for the covering 0–1 integer program. *Pacific Journal of Optimization*. (in press).

[50] Shunji Umetani and Mutsunori Yagiura. Relaxation heuristics for the set covering problem. *Journal of the Operations Research Society of Japan*, 50(4):350–375, 2007.

[51] Vijay V Vazirani. Introduction to LP-duality. In *Approximation Algorithms*, pages 93–107. Springer, 2003.

[52] Christophe Wilbaut, Said Hanafi, and Said Salhi. A survey of effective heuristics and their application to a variety of knapsack problems. *IMA Journal of Management Mathematics*, 19(3):227–244, 2008.

[53] Takeo Yamada, Seija Kataoka, and Kohtaro Watanabe. Heuristic and exact algorithms for the disjunctively constrained knapsack problem. *Information Processing Society of Japan Journal*, 43(9), 2002.