



# Approximation Algorithms for the Precedence Constrained Minimum Knapsack and Capacitated Covering Integer Programs

Antonios Skarlatos  
AL1180016

**Examination committee:**

*Archontia C. Giannopoulou, Department of  
Informatics and Telecommunications, National and  
Kapodistrian University of Athens.*  
*Stavros Kolliopoulos, Department of Informatics and  
Telecommunications, National and Kapodistrian  
University of Athens.*  
*Dimitris Fotakis, School of Electrical and Computer  
Engineering, National Technical University of  
Athens.*

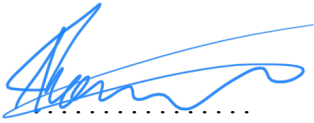
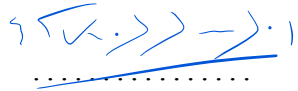
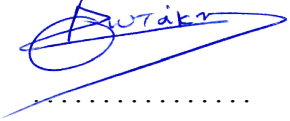
**Supervisor:**

*Stavros Kolliopoulos, Professor,  
Department of Informatics and  
Telecommunications,  
National and Kapodistrian University of  
Athens.*



Η παρούσα Διπλωματική Εργασία  
εκπονήθηκε στα πλαίσια των σπουδών  
για την απόκτηση του  
**Μεταπτυχιακού Διπλώματος Ειδίκευσης**  
**«Αλγόριθμοι, Λογική και Διακριτά Μαθηματικά»**  
που απονέμει το  
**Τμήμα Πληροφορικής και Τηλεπικοινωνιών**  
του  
**Εθνικού και Καποδιστριακού Πανεπιστημίου Αθηνών**

Εγκρίθηκε την 12/03/2021 από Εξεταστική Επιτροπή  
αποτελούμενη από τους:

<u>Όνοματεπώνυμο</u>	<u>Βαθμίδα</u>	<u>Υπογραφή</u>
1. Γιαννοπούλου Αρχοντία	Επ. Καθηγήτρια	
2. Κολλιόπουλος Σταύρος	Καθηγητής	
3. Φωτάκης Δημήτριος	Αν. Καθηγητής	



## ABSTRACT

In this thesis the main problem that we study is the knapsack problem and in particular a generalization of it which is the precedence constrained minimum knapsack problem. Initially we present the minimum knapsack problem with the intention to introduce some useful techniques that help us to develop a primal-dual algorithm for it. Afterwards, our attention is focused on the precedence constrained minimum knapsack problem, which is the knapsack problem with an extra constraint that the choice of elements must respect an ordering given through a partial order. We are studying some formulations of the problem together with their properties. Moreover we develop a rounding approximation algorithm for 0-1 PCKP and we present some new results for the PCKP with general multiplicity constraints. In the end of the PCKP chapter, an already known inapproximability result is presented a little differently. Finally, we study the capacitated covering integer programs and motivated by known techniques, we give some results for the special case of 0-1 CIP.



## ΣΥΝΟΨΗ

Στην διπλωματική το κύριο πρόβλημα μελέτης είναι το πρόβλημα του knapsack και ειδικότερα μία γενίκευσή του στην οποία υπάρχει μία έννοια προτεραιότητας ως προς την επιλογή των αντικειμένων. Αρχικά παρουσιάζουμε το πρόβλημα του minimum knapsack με σκοπό να εισάγουμε κάποιες χρήσιμες τεχνικές με τις οποίες μπορούμε να κατασκευάσουμε primal-dual αλγορίθμους για το πρόβλημα. Ύστερα, εστιάζουμε την προσοχή μας στο γενικευμένο πρόβλημα precedence constrained minimum knapsack, το οποίο είναι παρόμοιο με το knapsack με τον επιπλέον περιορισμό ότι η επιλογή των αντικειμένων πρέπει να σέβεται μία έννοια προτεραιότητας που ορίζεται μέσω μίας μερικής διάταξης. Για το πρόβλημα αυτό, μελετάμε κάποια γραμμικά προγράμματα μαζί με τις ιδιότητές τους. Ακόμα, κατασκευάζουμε έναν προσεγγιστικό αλγόριθμο στρογγυλοποίησης για το 0-1 PCKP και παρουσιάζουμε κάποια νέα αποτελέσματα για το PCKP για την περίπτωση που μπορούμε να διαλέξουμε κάποιο αντικείμενο και πάνω από μία φορά. Στο τέλος του PCKP κεφαλαίου, ένα ήδη γνωστό αποτέλεσμα κάτω φράγματος παρουσιάζεται με λίγο διαφορετικό τρόπο. Τέλος, μελετάμε τα capacitated covering ακέραια προγράμματα και εμπνευσμένοι από γνωστές τεχνικές, δίνουμε κάποια αποτελέσματα για την ειδική περίπτωση 0-1 CIP.





# Contents

<b>1</b>	<b>Introduction and Outline</b>	<b>1</b>
1.1	Approximation Algorithms for Optimization Problems . . . . .	1
1.2	Outline of the thesis and our contributions . . . . .	3
<b>2</b>	<b>The Minimum Knapsack Problem</b>	<b>7</b>
<b>3</b>	<b>The Precedence Constrained Minimum Knapsack Problem</b>	<b>13</b>
3.1	Definitions and Notations . . . . .	13
3.2	The PCKP problem and some valid formulations . . . . .	14
3.3	Pitch-1 inequalities . . . . .	18
3.4	Rounding algorithm for 0-1 PCKP . . . . .	20
3.5	PCKP with general multiplicity constraints . . . . .	22
3.6	Inapproximability of PCKP . . . . .	27
<b>4</b>	<b>Capacitated Covering Integer Programs</b>	<b>31</b>
4.1	Introduction . . . . .	31
4.2	Results for the 0-1 CIP . . . . .	32
<b>5</b>	<b>Conclusion and open problems</b>	<b>39</b>



# CHAPTER 1

## INTRODUCTION AND OUTLINE

### 1.1 Approximation Algorithms for Optimization Problems

In computer science, usually we have to deal with some discrete optimization problems for which we should find a feasible solution with the best possible cost. For these kind of problems, the ideal situation would be to develop exact and efficient algorithms. With the terms exact and efficient algorithm, what we actually mean is an algorithm that produces the best possible outcome and runs in polynomial time on the input size respectively. Unfortunately, for many interesting optimization problems we cannot expect to achieve both, accuracy and efficiency, unless  $P = NP$ . Therefore a common approach for this kind of problems is to relax the requirement of finding an optimal solution and instead, we just try to find a solution which is close to the optimal one. Hence rather than developing an exact algorithm, now our attention is focused on trying to develop an approximation algorithm for the same problem.

In general an optimization problem has an objective value that we want to maximize or minimize, while at the same time a set of constraints must be satisfied.

**Definition 1.1.** An *optimization problem* is a quadruple  $(I, sol, cost, goal)$  such that:

- $I$  is the set of instances.
- For an instance  $x \in I$ ,  $sol(x)$  is the set of feasible solutions of  $x$ .
- For an instance  $x \in I$  and a feasible solution  $y \in sol(x)$ , the value  $cost_x(y)$  is a positive real number denoting the cost of the solution.
- $goal$  is an operator which is either min or max.

The goal of an optimization problem for an instance  $x$ , is to find an optimal solution. That is, we want to find a  $y \in sol(x)$  such that:

$$cost_x(y) = goal\{cost_x(y') \mid y' \in sol(x)\}$$

For an optimal solution  $y$ , we denote as  $OPT(x) = cost_x(y)$  the cost of the optimal solution for the instance  $x$ .

As the quality of the solution is measured in terms of the cost, our desire for an optimization problem is to build a fast approximation algorithm which for every instance of the problem, produces a solution with cost close enough to the optimal cost of the instance. Moreover, if there is a proof that every time the produced solution is  $\rho$  times far away from the optimal one, then we say that this algorithm is a  $\rho$ -approximation algorithm.

**Definition 1.2.** A  $\rho$ -approximation algorithm for an optimization problem is a polynomial time algorithm that for all instances of the problem produces a feasible solution whose value is within a factor of  $\rho$  of the value of an optimal solution.

For example, let  $A$  be a polynomial time algorithm for a minimization problem. For an instance  $x$  we define as  $cost(A(x))$  the cost of the produced solution by  $A$ . Then if for every instance  $x$  it holds that  $cost(A(x)) \leq \rho \cdot OPT(x)$ , the algorithm  $A$  is a  $\rho$ -approximation algorithm.

In this thesis we are going to focus on minimization problems and we will try to explore approximation algorithms for these. A very common approach that is followed in order to develop an approximation algorithm, is to use the theory behind linear programming. The strategy here is to formulate the initial optimization problem as an integer program and make use of its linear programming relaxation which usually can be solved in polynomial time. However, we are interested only in integer solutions and thus, two common ways to end up with an integer solution are either to convert the linear solution to an integer one, or to construct an integer solution using primal-dual algorithms based on the dual of the linear program. In any case, with the standard techniques of the linear programming theory, our intention is to bound the cost of the produced integer solution by  $\rho$  times of the optimal linear cost. The underlying reason is that this value is a lower bound of the optimal integer cost and usually the only one that we can efficiently compute in order to estimate the accuracy of the algorithm.

As we are focusing on minimization problems, the related integer programs that we focus on, are the so-called *covering integer programs* of the following form:

$$\begin{aligned} & \text{minimize } c^T x \\ & \text{subject to } Ax \geq D \\ & \quad x \in \mathbb{Z}_+^n \end{aligned}$$

where  $A \in \mathbb{Z}_+^{m \times n}$  is a non-negative integral matrix with the value  $u_{ij}$  in its  $i_{th}$  row and  $j_{th}$  column,  $D \in \mathbb{Z}_+^m$  is a demand vector with entries  $D_i$ ,  $c \in \mathbb{Z}_+^n$  is a cost vector with entries  $c_j$  and  $d \in \mathbb{Z}_+^n$  is a multiplicities vector with entries  $d_j$ . The values of  $m$  and  $n$  denote the number of the constraints and the number of the variables respectively and in the case that we have more than one constraints, we use the index  $i$  for the constraints and the index  $j$  for the variables. If we also allow constraints of the type  $x \leq d$ , we get the *capacitated covering integer programs (CIP)*. Also in many cases we are going to work with a 0-1 CIP, which is a regular CIP with the difference that the variables  $x_j$  are only allowed to get the value zero or one, which is equivalent to say that the multiplicity constraint vector  $d$  is a vector of all ones.

Given an optimization problem and an integer program that formulates it, sometimes it is not possible to use any of the known techniques of the linear programming theory in order to construct a bounded integer solution. For every instance of the problem we would like to bound the cost of the produced integer solution in terms of the optimal linear cost. However, if for every arbitrarily large value there is an instance

such that the gap between its optimal integer solution with its optimal linear solution is that big, the standard analysis of the cost fails and we cannot develop a  $\rho$ -approximation algorithm. Therefore for an integer program that formulates an optimization problem, a very important property is its integrality gap.

**Definition 1.3.** Consider an integer program  $P$  for an optimization problem with a set of instances  $I$ . For an instance  $x \in I$ , let us denote by  $OPT_{IP}(x)$  and  $OPT_{LP}(x)$  the cost of the optimal integer solution of  $P$  and the cost of the optimal linear solution of the linear programming relaxation of  $P$ , for that instance respectively. The *integrality gap* of the integer program  $P$  is equal to the value  $\max_{x \in I} \left( \max \left( \frac{OPT_{IP}(x)}{OPT_{LP}(x)}, \frac{OPT_{LP}(x)}{OPT_{OPT}(x)} \right) \right)$ .

Consequently, a formulation with an unbounded integrality gap is not very helpful in our attempt to develop a  $\rho$ -approximation algorithm. To handle this issue though, a solution would be to extend the integer program by adding some extra valid inequalities which intuitively approach the underlying problem in a better way, and eventually to end up with a new integer program with a fixed integrality gap.

**Definition 1.4.** For an integer program  $P$  a new *inequality is valid*, if every feasible integer solution of  $P$  still remains feasible even after the addition of the new inequality to  $P$ .

The purpose of adding valid inequalities, assuming we work with a minimization problem, is because we would like to make infeasible very cheap linear solutions in terms of the cost, but at the same time to maintain all the previous feasible integer solutions. The new resulting integer program  $B$  is then a valid relaxation and its linear relaxation  $C$  is also a valid linear relaxation of the initial integer program  $A$ . The reason that we need to maintain all the feasible integer solutions, is because we actually want to maintain the original optimal integer solution, as this would imply that the optimal linear solution of  $C$  is still a lower bound of the original optimal integer solution of  $A$ . Therefore, assuming that we could develop a polynomial time algorithm that constructs a feasible integer solution for the original minimization problem, with cost  $\rho$  times the cost of the optimal linear solution of  $C$ , we could apply the standard analysis of the cost and prove that the algorithm is indeed a  $\rho$ -approximation algorithm. Furthermore, such an algorithm would prove that the integrality gap of  $B$  is at most  $\rho$ , because the cost of the optimal integer solution of  $B$  is a lower bound of the optimal integer cost of  $A$ .

Further background on approximation algorithms can be found in [18], [19]. All the problems that we are studying are NP-hard, and this is the reason that we are interested in developing approximation algorithms for them. Background in the theory of NP-hardness and computational intractability can be found in [7].

## 1.2 Outline of the thesis and our contributions

**Problems Studied.** Initially in the second chapter, we study the *minimum knapsack problem*. In this problem we are given a set of elements with a value and a cost for each one, and we are asked to select the minimum, in terms of the cost, subset of elements such that, the sum of their values cover a specified demand. In the third chapter, we study a generalization of the minimum knapsack problem which is the *precedence constrained minimum knapsack problem (PCKP)*. The input and the goal in the PCKP problem is the same as in the minimum knapsack problem with an extra restriction. In the input we are also given a partial order on elements, and every feasible solution must

respect the partial order, namely, in order to pick an element we must first pick all its predecessors with respect to the partial order. In the fourth chapter we study the CIP problem in which we must satisfy not only one constraint but a set of them, given by a matrix.

**Previous Work.** The minimum knapsack problem was studied by Carr et al. [3] and they developed a 2-approximation rounding algorithm for the problem, using the knapsack cover inequalities. Later on, Carnes and Shmoys [2] developed a faster 2-approximation primal-dual algorithm for the minimum knapsack problem. This primal-dual algorithm is the base for the primal-dual algorithms that were developed by McCormick et al. in [12], where they studied the PCKP problem. The integrality gap of the formulations for the PCKP problem, even with the knapsack cover inequalities can be unbounded. For this reason, they introduced the precedence knapsack cover inequalities with the help of which they developed a  $w(\mathcal{P})$ -approximation primal-dual algorithm for the 0-1 case, that is, when each element can be picked at most one time. Here  $\mathcal{P}$  is a partial order that defines the ordering of the selection and  $w(\mathcal{P})$  is the size of the maximum antichain in  $\mathcal{P}$ . For the PCKP problem with general multiplicity constraints, in which each element can be picked more than one time, McCormick et al. [12] present a pseudo-polynomial algorithm with approximation ratio equal to  $w(\mathcal{P}) \cdot \Delta$ , where  $\Delta = \max_j d_j$  is an upper bound on the multiplicity variables and they state that it remains open to find an algorithm with strongly polynomial bounds. Moreover they give an inapproximability result, which says that the PCKP problem does not admit PTAS under standard complexity assumptions. For the 0-1 CIP problem, Fujito [6] and Carr et al. [3] provide  $f$ -approximation algorithms, where  $f$  is the maximum number of non-zero coefficients in a row of the matrix of the constraints. In [16] though, they present an  $(f - \frac{f-1}{m})$ -approximation primal-dual algorithm, which has slightly better ratio but they pay an extra  $O(n^2)$  factor in the time complexity. Here  $n$  and  $m$  are the number of the variables and constraints respectively.

**Outline and our Contributions.** In the second chapter we present results from the literature for the minimum knapsack problem. In the third chapter we focus on the PCKP problem, and in the beginning we study the formulation of McCormick et al. [12] with the intention to understand its properties. Afterwards, we study a slightly different formulation for the PCKP problem and we develop a  $w(\mathcal{P})$ -approximation rounding algorithm for the 0-1 case. In Section 3.5 we present a polynomial algorithm with strongly polynomial approximation ratio. Finally in the end of the third chapter we provide a more detailed proof for the inapproximability result of the PCKP problem [12]. The last chapter is devoted to the capacitated covering integer programs and specifically to the 0-1 case. There, we give some results and observations which are related to the Fujito and Carr et al. linear programming relaxations and algorithms.

Let us explain in detail our contributions in this thesis, in order to help the reader to distinguish them from the previous work. The first results come from Theorem 3.1, Lemma 3.2 and Corollary 3.3. There we prove that the formulation of McCormick et al. [12] for the PCKP problem has integrality gap equal to  $\Omega(w(\mathcal{P}))$ . This result shows that every algorithm based on this formulation cannot have much better ratio than their  $w(\mathcal{P})$ -approximation primal-dual algorithm. Next, in Theorem 3.4 we study a slightly simpler formulation than the one which was suggested by McCormick et al. [12] and in Theorem 3.5 we prove that in the worst case the two formulations have the same ratio. Also in Lemma 3.6 we show that the combination of the two formulations is

unnecessary, as the simpler one that we suggest, is implied by the one suggested at [12]. In the next section, Lemma 3.7 shows a useful property of the formulations which is that the feasible minimal solutions in terms of their cost, have a specific structure. Using that lemma, through Lemma 3.8 and Theorem 3.9, a new  $w(\mathcal{P})$ -approximation rounding algorithm for the 0-1 PCKP problem is presented. In the Section 3.5, a set of results is presented until we reach in the main result of Theorem 3.14 which says that there exists an  $O(\min\{|U|, w^2(\mathcal{P})\})$ -approximation primal-dual algorithm for the general PCKP problem with multiplicity constraints. This solves the open problem stated by [12], about whether there exists an algorithm for the general PCKP problem with strongly polynomial bounds. As regards the inapproximability of the PCKP problem, we flesh out the details of the proof of Theorem 6 [12] (cf. the proof of Theorem 3.17). Finally in the last chapter, in Theorem 4.1 we prove that the already known algorithm of Fujito for the 0-1 CIP problem, has a better approximation ratio which is equal to  $f - \frac{f-2}{m}$ . The same result is also proved in Theorem 4.3 with the primal-dual algorithm that we develop for the 0-1 CIP problem and it is based on the Carr et.al relaxation. This ratio is slightly worse than the  $(f - \frac{f-1}{m})$  of [16], but can be achieved without the extra  $O(n^2)$  factor in the time complexity. However because of Lemma 4.2 the bound of the algorithms is almost tight.



*1.2. OUTLINE OF THE THESIS AND OUR CONTRIBUTIONS*

---

## CHAPTER 2

### THE MINIMUM KNAPSACK PROBLEM

The knapsack problem is one of the most studied problems in combinatorial optimization and usually in the literature it is presented as a maximization problem in the packing variant. In this section though, we are going to focus on the minimization version of it, in the covering variant, with the intention of presenting some techniques that have been developed and used for this problem, but could be applied for other problems as well.

In the *minimum knapsack problem* we are given some elements with a cost and a value for each one and also a specified demand. The goal of the problem is to select the minimum cost set of elements such that their total value is at least the demand. Formally let  $U$  be a set of elements,  $c : U \rightarrow \mathbb{Z}_+$  and  $u : U \rightarrow \mathbb{Z}_+$  be the cost and the value functions respectively and  $D$  be the specified demand. Then the goal is to find a subset of elements  $S \subseteq U$ , such that  $\sum_{e \in S} u(e) \geq D$ , while minimizing the sum  $\sum_{e \in S} c(e)$ . Since the problem is NP-hard [9] we should not expect to build a polynomial algorithm on its input size, but we should rather approach it in a different way. In the rest of this chapter we are going to present the steps, in order to build an 2-approximation algorithm for the problem.

Instead of functions, we could visualize  $c$  and  $u$  as vectors, where their  $i_{th}$  component  $c_i$  and  $u_i$  corresponds to the cost and the value of the element  $i \in U$  respectively. Hence it becomes obvious that the minimum knapsack is a special case of CIP, because we can formulate an instance of the minimum knapsack as a CIP instance:

$$\begin{aligned} & \text{minimize } \sum_{i \in U} c_i \cdot x_i \\ & \text{subject to } \sum_{i \in U} u_i \cdot x_i \geq D \\ & \quad x \in \{0, 1\}^{|U|} \quad (P_{KP}) \end{aligned}$$

and  $x_i = 1$  if and only if the element  $i \in U$  has been selected.

The formulation  $(P_{KP})$  is what we call the natural integer program for the minimum knapsack. However with this formulation there is an important obstacle to applying the standard techniques of the linear programming theory. This issue has to do with the integrality gap, which it happens to be unbounded. Actually the integrality gap of this formulation can be as large as the demand  $D$ , as stated in Lemma 2.1, and as a result

---

one cannot develop an approximation algorithm with constant guarantee based on this formulation.

**Lemma 2.1.** [3] The integrality gap of  $(P_{KP})$  can be as large as the demand  $D$ .

*Proof.* Let two elements  $e_1, e_2$  and a specified demand  $D$ . The costs and the values of the elements are equal to:

$$\begin{aligned} c(e_1) &= 0, & u(e_1) &= D - 1 \\ c(e_2) &= 1, & u(e_2) &= D \end{aligned}$$

The corresponding integer program for this instance would be the following:

$$\begin{aligned} &\text{minimize } x_2 \\ &\text{subject to } (D - 1) \cdot x_1 + D \cdot x_2 \geq D \\ &\quad x_1, x_2 \in \{0, 1\} \end{aligned}$$

We can observe that including the first element in our solution does not increase the cost of the solution but increases the total value, and thus it is always beneficial to pick this element. However, there is a residual demand that we must cover and this is equal to one. In the integer version we must necessarily pick the second element as well, getting an integer solution with total cost equal to one. On the other hand, as the value of the second element is equal to  $D$ , in the linear version we are allowed to select only a fraction of  $\frac{1}{D}$  of the second element in order to cover the residual demand, and end up with a linear solution with total cost equal to  $\frac{1}{D}$ . Consequently the cost of the optimal integer solution is  $D$  times larger than the cost of the optimal linear solution and thus the integrality gap of the above formulation can be as large as the demand  $D$ .  $\square$

In order to handle this issue and make use of the techniques from the linear programming theory, Carr et al. [3] introduced the so-called knapsack cover inequalities. These inequalities are extremely useful, because not only they strengthen the linear programming relaxation, reducing the integrality gap to the constant two, but also they help us to develop 2-approximation algorithms for the minimum knapsack problem. Furthermore, knapsack cover inequalities are not restricted only to this problem, but they can also be applied to many other optimization problems, including the general CIP problem.

Let us now define the knapsack cover inequalities and try to give an intuition for the reasons that they are powerful and they reduce the integrality gap. Consider a minimum knapsack instance with elements from  $U$ , demand  $D$  and  $u$  be the vector of values. For a subset of elements  $A \subseteq U$ , we denote by:

$$D(A) = \max\left\{D - \sum_{i \in A} u_i, 0\right\}$$

the residual demand that we must cover if we have already picked all the elements from  $A$ . We know that every feasible solution  $x$  satisfies the constraint  $\sum_{i \in U} u_i \cdot x_i \geq D$ . For a subset of elements  $A \subseteq U$ , let us split the previous constraint into two sums as

follows:

$$\begin{aligned} \sum_{i \in U \setminus A} u_i \cdot x_i + \sum_{i \in A} u_i \cdot x_i &\geq D \Leftrightarrow \\ \sum_{i \in U \setminus A} u_i \cdot x_i &\geq D - \sum_{i \in A} u_i \cdot x_i \Rightarrow \\ \sum_{i \in U \setminus A} u_i \cdot x_i &\geq D(A) \end{aligned}$$

The first two constraints are equivalent, and the last one is weaker than the other two because  $D(A)$  assumes that each element which is part of  $A$  has been fully selected. If we would add the last inequality, all the previous feasible linear solutions would still remain feasible and so the integrality gap would not be reduced. The key observation here, is that as we are only interested in integer solutions, if the value of an element is larger than the residual demand, that is, it holds that  $u_i > D(A)$ , we could set  $u_i$  equal to  $D(A)$  for this constraint without cutting off any integer solution, while at the same time maybe we would lose some linear solutions. It is important at this point to remind us that  $D(A)$  is a non-negative number. Intuitively, by setting the coefficients of the left hand side to as small as possible, we are forcing the vector  $x$  to get higher values, closer to one, and thus very small linear solutions become infeasible, reducing that way the integrality gap. The transformation of the values for a subset of element  $A \subseteq U$ , will be denoted as:

$$u_i(A) = \min\{u_i, D(A)\}$$

and it can be thought as the effective value of an element given that the elements in  $A$  are part of the solution. The new strengthened integer program for the minimum knapsack is:

$$\begin{aligned} &\text{minimize } \sum_{i \in U} c_i \cdot x_i \\ &\text{subject to } \sum_{i \in U \setminus A} u_i(A) \cdot x_i \geq D(A), \forall A \subseteq U \\ &x \in \{0, 1\}^{|U|} \quad (P_{KP1}) \end{aligned}$$

The claim is that the integer program  $(P_{KP1})$  formulates the minimum knapsack problem. The first thing that we should do is to prove formally that indeed the knapsack cover inequalities are valid inequalities and as a result  $(P_{KP1})$  is a valid relaxation of  $(P_{KP})$ . In order to prove this, from the definition of a valid inequality, we should prove that every feasible integer solution of  $(P_{KP})$  remains feasible after the addition of the new extra inequalities and thus, it is also feasible to  $(P_{KP1})$ .

**Lemma 2.2.** [3] The formulation  $(P_{KP1})$  is a valid relaxation of  $(P_{KP})$ .

*Proof.* Let  $x$  be an integer feasible solution for an instance of the minimum knapsack with support vector  $S = \{i \mid x_i = 1\}$ . It is true that  $\sum_{i \in S} u_i \geq D$ . Let us assume that there is a subset  $A \subseteq U$  such as the corresponding inequality is not satisfied by  $x$ . For this subset  $A$ , it must be true that  $D(A) = D - \sum_{i \in A} u_i > 0$  and the unsatisfied inequality will look like:

$$\sum_{i \in U \setminus A} u_i(A) \cdot x_i < D(A) \Rightarrow \sum_{i \in S \setminus A} u_i(A) < D(A)$$

---

For each  $i \in S \setminus A$ , the value  $u_i$  cannot be greater or equal to  $D(A)$ , because  $u_i(A)$  would be equal to  $D(A)$  and the inequality would be satisfied. Consequently, it holds that  $u_i < D(A)$  and so  $u_i(A) = u_i$ . Hence the sum is equal to:

$$\sum_{i \in S \setminus A} u_i(A) = \sum_{i \in S \setminus A} u_i < D(A) \Rightarrow \sum_{i \in S \setminus A} u_i + \sum_{i \in A} u_i < D$$

But then:

$$\sum_{i \in S} u_i \leq \sum_{i \in S \setminus A} u_i + \sum_{i \in A} u_i < D$$

which contradicts the fact that  $x$  is an integer feasible solution of  $(P_{KP})$ . Therefore  $x$  must satisfy all the extra knapsack cover inequalities and so,  $(P_{KP1})$  is a valid relaxation of  $(P_{KP})$ .  $\square$

As a next step, we should show that  $(P_{KP1})$  is indeed more powerful formulation than the natural formulation  $(P_{KP})$ . A way to show something like that, is by developing an approximation algorithm with a constant guarantee. Carr et al. [3] developed the bucketing algorithm which is a rounding 2-approximation algorithm, and proved that the integrality gap of  $(P_{KP1})$  is at most 2. Moreover they gave an infinite family of instances with gap at least  $2 - \frac{2}{|U|}$ , and proved that the integrality gap of  $(P_{KP1})$  is very close to 2. Therefore using this formulation, we should not expect to develop an approximation algorithm with better constant guarantee than 2, and so their bound is almost tight.

However in this section, our attention will be focused on the 2-approximation primal-dual algorithm that was developed by Carnes and Shmoys [2] which is also based on the formulation  $(P_{KP1})$ , thus it can also be used in order to prove similar results for the integrality gap of the formulation. Consider the linear relaxation of  $(P_{KP1})$ :

$$\begin{aligned} & \text{minimize} && \sum_{i \in U} c_i \cdot x_i \\ & \text{subject to} && \sum_{i \in U \setminus A} u_i(A) \cdot x_i \geq D(A), \forall A \subseteq U \\ & && x \geq 0 \end{aligned}$$

and the dual of this linear program:

$$\begin{aligned} & \text{maximize} && \sum_{A \subseteq U} D(A) \cdot y(A) \\ & \text{subject to} && \sum_{A \subseteq U: i \notin A} u_i(A) \cdot y(A) \leq c_i, \forall i \in U \\ & && y(A) \geq 0, \forall A \subseteq U \end{aligned}$$

The algorithm follows the primal-dual schema. The algorithm initially starts with a feasible dual solution and an infeasible primal integer solution and tries to construct a feasible primal integer solution without violating the feasibility of the produced linear dual solution. This is due to the fact that we would like to bound the cost of the primal integer solution in terms of a constant factor of the cost of the linear dual solution, as this is enough in order to prove that the approximation algorithm has a constant guarantee.

The algorithm initially sets  $S = \emptyset$  and  $y = 0$ , where  $S$  is the set of the chosen elements and  $y$  is the dual vector. As long as the set  $S$  is an infeasible primal integer

solution, we increase the dual variable  $y(S)$  as much as possible until a constraint in the dual linear program becomes tight. Let  $i$  be the element for which the corresponding inequality in the dual linear program becomes tight. Then we add  $i$  to the set  $S$  and the same process is repeated. The pseudocode of the algorithm *Algorithm 1* is presented below:

---

```

S = ∅, y = 0

while (D(S) > 0 && |S| < n) {
    Increase y(S) until a dual constraint becomes tight for item i
    S = S ∪ {i}
}

return (S, y)

```

---

The proof that the algorithm is indeed a 2-approximation algorithm follows from the two next lemmas. In the first lemma we prove that the algorithm always finds a feasible pair  $(S, y)$ , assuming that there is one, and in the second lemma we prove that the cost of the set  $S$  is at most two times larger than the cost of  $y$ , proving in that way that the algorithm has an approximation factor of 2.

**Lemma 2.3.** [3] For a feasible minimum knapsack instance, the primal-dual pair  $(S, y)$  which is produced by the algorithm is feasible.

*Proof.* Let  $S$  be the current set in a step of the algorithm. The algorithm increases the dual variable  $y(S)$  as much as possible until a constraint  $i$  becomes tight. The variable  $y(S)$  does not participate in any inequality  $i \in S$  and so, for every previous element that already belongs to  $S$ , the corresponding inequality that has become tight is not affected. Therefore in the case that the current set  $S$  is not yet a feasible solution, we can proceed with  $y(S)$  because it participates only in inequalities where the corresponding element is not yet included in  $S$ .  $\square$

**Lemma 2.4.** [3] The primal-dual algorithm has an approximation factor of 2.

*Proof.* Let  $(S, y)$  be a feasible pair, where  $S$  is the set of elements that the algorithm has picked and  $y$  is the dual vector that the algorithm has produced. The total cost of  $S$  is equal to:

$$\text{cost}(S) = \sum_{i \in S} c_i = \sum_{i \in S} \sum_{A \subseteq U: i \notin A} u_i(A) \cdot y(A)$$

where the last equality holds because the algorithm has picked elements for which the corresponding inequality is tight. We would like to bound the cost of our solution, by some factor times the cost of the dual solution  $y$ . Hence we can rearrange the sums in order to get something similar to the cost function of the dual linear program.

$$\text{cost}(S) = \sum_{A \subseteq U} \sum_{i \in S \setminus A} u_i(A) \cdot y(A) = \sum_{A \subseteq U} y(A) \cdot \sum_{i \in S \setminus A} u_i(A) \quad (1)$$

Now we will try to bound the second sum. Let  $l$  be the last element selected by the algorithm. Because of the way the algorithm selects elements,  $y(A) > 0$  implies that  $A \subseteq S \setminus \{l\}$ , and so we can focus our attention only in these special subsets of elements. Moreover it must be true that:

$$\sum_{i \in S \setminus \{l\}} u_i < D$$

---

because otherwise the algorithm would have stopped earlier. Therefore as  $u_i(A) = \min\{u_i, D(A)\}$ , we can conclude that for a fixed  $A \subseteq U$ , such that  $y(A) > 0$ , it holds that:

$$\begin{aligned}
\sum_{i \in S \setminus A} u_i(A) &= \sum_{i \in (S \setminus \{l\}) \setminus A} u_i(A) + u_l(A) && \leq \\
&= \sum_{i \in (S \setminus \{l\}) \setminus A} u_i + u_l(A) && = \\
&= \sum_{i \in S \setminus \{l\}} u_i - \sum_{i \in A} u_i + u_l(A) && < \\
&= D - \sum_{i \in A} u_i + u_l(A) = D(A) + u_l(A) && \leq \\
&= 2 \cdot D(A)
\end{aligned}$$

As a consequence, from (1) we get that for the cost of  $S$  it holds that:

$$cost(S) \leq 2 \cdot \sum_{A \subseteq U} y(A) \cdot D(A)$$

and as the cost of a feasible dual solution is a lower bound of the optimal primal integer solution, the algorithm has an approximation factor of 2.  $\square$

To sum up, combining the two previous lemmas, for a feasible instance of the minimum knapsack the primal-dual algorithm produces a feasible pair  $(S, y)$  such that the cost of  $S$  exceeds at most two times the cost of  $y$ . As a corollary, because the cost of a dual solution is a lower bound of the cost of the optimal primal linear solution, we obtain that for the  $(P_{KP1})$ , its optimal integer solution is at most two times larger than its optimal linear solution. Thus the integrality gap of  $(P_{KP1})$  is at most two.

# CHAPTER 3

## THE PRECEDENCE CONSTRAINED MINIMUM KNAPSACK PROBLEM

### 3.1 Definitions and Notations

In this section we are going to give some useful definitions and notations that will be used later. Consider a partially ordered set  $\mathcal{P} = (U, \preceq)$ , where  $U$  is a set of elements and  $\preceq$  is an order relation for pair of elements. Two elements  $i, j \in U$  are *comparable* if  $i \preceq j$  or  $j \preceq i$  holds and *incomparable* otherwise. A subset  $A \subseteq U$  of elements is called a *chain* if for every pair of elements  $i, j \in A$  it holds that  $i, j$  are comparable. Likewise, a subset  $A \subseteq U$  of elements is called an *antichain* if for every pair of elements  $i, j \in A$  it holds that  $i, j$  are incomparable. The *size* of the maximum antichain will be used as a guarantee in our approximation algorithms and will be denoted as  $w(\mathcal{P})$ .

A subset  $A \subseteq U$  is called an *ideal* if for every element  $i \in A$  it is true that all the elements  $j$  with the property  $j \preceq i$  are also part of  $A$ . In other words, for an ideal  $A$ ,  $i \in A$  and  $j \preceq i$  implies  $j \in A$ . An ideal  $A$  is also said to be *closed* under  $\mathcal{P}$ . For a partial order  $\mathcal{P}$  the set of all the ideals of  $\mathcal{P}$  will be denoted as  $\mathcal{L}(\mathcal{P})$ .

We define  $\mathcal{P}(A) = (U \setminus A, \preceq)$  for an ideal  $A \in \mathcal{L}(\mathcal{P})$  to be the partial order  $\mathcal{P}$  restricted to the elements that do not belong to  $A$  and  $\min \mathcal{P}(A) = \{i \in U \setminus A \mid \nexists j \in U \setminus A \text{ such that } j \prec i\}$  to be the set of minimal items of the partial order  $\mathcal{P}(A)$ . These notations will be useful later, as for a set  $S$  which will not be yet a feasible solution, we will need to pick an extra element in order to update the set  $S$ . To keep the set  $S$  closed under  $\mathcal{P}$ , a minimal element should be chosen with respect to  $\mathcal{P}(S)$  or in other words an element from the set  $\min \mathcal{P}(S)$ .

Finally, for an ideal  $A \in \mathcal{L}(\mathcal{P})$  and an element  $j \in U \setminus A$ , we define  $X_j(A) = \{i \in U \setminus A \mid i \preceq j \wedge i \in \min \mathcal{P}(A)\}$  as the set of elements that are simultaneously minimals of  $\mathcal{P}(A)$  and comparable with the element  $j$ . This set of elements will be used later, when we will project the value of the element  $j$  uniformly onto the minimal and comparable with it, items with respect to  $\mathcal{P}(A)$ , with the aim to add new valid inequalities.



### 3.2 The PCKP problem and some valid formulations

In the minimum knapsack we can choose the elements in an arbitrary order. A possible extension of this problem would be to add the restriction that the chosen elements should respect a specific order. This is the *precedence constrained minimum knapsack problem* (PCKP) and is defined as follows. Let  $U$  be a set of elements,  $c : U \rightarrow \mathbb{Z}_+$  and  $u : U \rightarrow \mathbb{Z}_+$  be the cost and the value functions respectively,  $D$  be the specified demand and  $\mathcal{P}$  be a partial order. Then the goal is to find a subset of elements  $S \subseteq U$  that minimizes the sum  $\sum_{e \in S} c(e)$  and the two following constraints are satisfied. The first constraint is to satisfy the demand, namely,  $\sum_{e \in S} u(e) \geq D$  and the second constraint is to respect the partial order, that is, whenever we choose an element  $e$ , all the elements below of  $e$  with respect to  $\mathcal{P}$  must also be part of the solution.

This problem which is a generalization of the minimum knapsack can also be written as an integer program. The additional constraint related to the partially order can be described in the following way. For every elements  $i, j$  such that  $i \preceq j$  with respect to  $\mathcal{P}$ , it must be true that  $x_i \geq x_j$ . Thus the natural way to write the 0-1 PCKP problem as an integer program is the following one:

$$\begin{aligned} & \text{minimize} && \sum_{i \in U} c_i \cdot x_i \\ & \text{subject to} && \sum_{i \in U} u_i \cdot x_i \geq D \\ & && x_i - x_j \geq 0, \forall i \preceq j \\ & && x \in \{0, 1\}^{|U|} \end{aligned}$$

In the case that  $\mathcal{P}$  is an antichain, the integer program represents the classical minimum knapsack problem, and from Lemma 2.1 the integrality gap will be unbounded. However we are allowed to add the valid knapsack cover inequalities and end up with this strengthened integer program:

$$\begin{aligned} & \text{minimize} && \sum_{i \in U} c_i \cdot x_i \\ & \text{subject to} && \sum_{i \in U \setminus A} u_i(A) \cdot x_i \geq D(A), \forall A \subseteq U \\ & && x_i - x_j \geq 0, \forall i \preceq j \\ & && x \in \{0, 1\}^{|U|} \quad (P_{PCKP}) \end{aligned}$$

Nevertheless as stated in Proposition 1, even this strengthened formulation has an unbounded integrality gap which is specifically  $\Omega(|U|)$ .

**Proposition 1.** [12] Formulation  $(P_{PCKP})$  has an unbounded integrality gap.

*Proof.* Consider an instance of PCKP with  $n$  items and demand  $D = 1$ . All the values  $u_i$  of the elements are equal to 1. The cost of the first element  $c_1$  is equal to  $n$  and the costs of the other elements are equal to 1. The partial order is a chain of length  $n$  with  $1 \preceq 2 \preceq \dots \preceq n$ .

Then the optimal integer solution has to buy the first element at cost  $n$  due to the partial order constraints. However, the linear solution selects the solution vector  $(\frac{1}{n}, \dots, \frac{1}{n})$  with solution cost  $\frac{n}{n} + \frac{n-1}{n} < 2$ .  $\square$

McCormick et al. [12] in order to handle this issue introduced the so-called precedence constrained knapsack cover inequalities which behave almost in the same way as the knapsack cover inequalities but they are now focused only on minimal items. That is because the solutions that we are interested in, should be closed under the partial order. The problem with the knapsack cover inequalities is that they reduce the gap between the linear and the integer solutions which are not necessarily closed under the partial order, and so the distance between the closed linear and integer solutions can still be arbitrary large. To eliminate the very cheap linear solutions, we should somehow force the solution vector  $x$  to set as high values as possible to the elements that should be selected in order to get a closed solution under the partial order and this is exactly the idea behind the precedence constrained knapsack cover inequalities. Assuming that an ideal  $A \in \mathcal{L}(\mathcal{P})$  is not yet a feasible solution, these constraints force us to pick a minimal element because only in this case the produced solution will remain closed. As regards the approximation factor, intuitively as the inequalities are built upon the partial order, the integrality gap now will be bounded in terms of  $\mathcal{P}$  and specifically as we will see later, from the value  $w(\mathcal{P})$ .

Let us explain now more formally how these inequalities look like. Consider an ideal  $A \in \mathcal{L}(\mathcal{P})$  and let  $D(A) = \max\{D - \sum_{i \in A} u_i, 0\}$  be the residual demand that we must cover. If  $D(A) > 0$ , then to update  $A$  towards a feasible solution, we must necessarily pick an element from the set  $\min \mathcal{P}(A)$  and this idea could be described by the valid inequality:

$$\sum_{i \in \min \mathcal{P}(A)} x_i \geq 1 \quad (3.1)$$

However as they write in [12], these inequalities do not distinguish between minimal elements which enable us to select a large amount of value and minimal elements which do not have any successors, and this is why they introduce a slightly different set of inequalities. Nevertheless, even if the inequalities that they suggest in [12] appear as more powerful, we will show later that in the worst case they actually have the same guarantee with the inequalities (3.1). The suggested precedence knapsack cover inequalities work alike, but in the right hand side instead of 1, we have the real residual demand  $D(A)$ . However a problem that arises here is that in order to keep these inequalities valid, we should also make use of the values of the elements, but as we sum only over the minimal items  $\min \mathcal{P}(A)$ , we should somehow include the values of the other elements in our sum as well. Therefore, for an ideal  $A \in \mathcal{L}(\mathcal{P})$  and an element  $i \in \min \mathcal{P}(A)$  they define in [12]:

$$\bar{u}_i(A) = \min \left\{ u_i(A) + \sum_{j: i \prec j} \frac{u_j(A)}{|X_j(A)|}, D(A) \right\}$$

to be the value of an element plus all the uniformly shared values of the elements above it assuming that we project the values of each element  $j \in (U \setminus A) \setminus \min \mathcal{P}(A)$  uniformly in the set of elements  $X_j(A)$ . Again, as we are interested only in integer solutions, the coefficients that exceed the right hand side can be cropped and become equal to  $D(A)$ . The resulting strengthened integer program that occurs by adding these inequalities is

the following one:

$$\begin{aligned}
 & \text{minimize} && \sum_{i \in U} c_i \cdot x_i \\
 & \text{subject to} && \sum_{i \in \min \mathcal{P}(A)} \bar{u}_i(A) \cdot x_i \geq D(A), \forall A \in \mathcal{L}(\mathcal{P}) \\
 & && x \in \{0, 1\}^{|U|} \quad (P_{PCKP1})
 \end{aligned}$$

and the dual of the linear relaxation is:

$$\begin{aligned}
 & \text{maximize} && \sum_{A \in \mathcal{L}(\mathcal{P})} y(A) \cdot D(A) \\
 & \text{subject to} && \sum_{A \in \mathcal{L}(\mathcal{P}): i \in \min \mathcal{P}(A)} \bar{u}_i(A) \cdot y(A) \leq c_i, \forall i \in U \\
 & && y(A) \geq 0, A \in \mathcal{L}(\mathcal{P}) \quad (D_{PCKP1})
 \end{aligned}$$

Based on this  $(P_{PCKP1})$  formulation, in [12] they develop a  $w(\mathcal{P})$ -approximation algorithm for the 0-1 PCKP. Moreover in Lemma 5 of [12], they give a family of instances with the property that the cost of the produced dual linear solution is  $w(\mathcal{P})$  times smaller than the cost of the produced primal integer solution, and so the bound of the algorithm is tight. However someone could argue that maybe it is possible to develop another algorithm with better approximation factor based on this formulation. In the two next results though we prove something stronger for the formulation itself, and we show that this is not possible.

**Theorem 3.1.** There is an infinite family of instances of PCKP, such that the gap between the optimal integer solution of  $(P_{PCKP})$  with the optimal linear solution of the linear relaxation of  $(P_{PCKP1})$  is  $\Omega(w(\mathcal{P}))$ .

*Proof.* Consider a partial order with  $2n$  elements consisting of  $n$  parallel chains each of length two. The values and the costs of the bottom layer are defined as  $(\forall i)(1 \leq i \leq n) : u_i^1 = c_i^1 = 1$ . The values and the costs of the top layer are defined as  $(\forall i)(1 \leq i \leq n) : u_i^2 = c_i^2 = \frac{n}{2}$ . Let us consider a PCKP instance with  $D = \frac{n}{2}$ .

For the integer solution of  $(P_{PCKP})$  we can either choose an element  $e_i^2$  from the top layer or not. In the first case though, we must also choose the element  $e_i^1$  which is located below of  $e_i^2$ , because the solutions must be closed under the partial order. The cost of the solution in this case will be at least  $\frac{n}{2} + 1$ . However, in the second case we can choose  $\frac{n}{2}$  elements only from the bottom layer, giving us a feasible integer solution with cost  $OPT = \frac{n}{2}$ , which is also the optimal integer solution for  $(P_{PCKP})$ .

Let us now consider a solution for the linear relaxation of  $(P_{PCKP1})$ . Set  $(\forall i)(1 \leq i \leq n) : x_i^1 = \frac{2}{n}, x_i^2 = 0$ . Initially we will show that  $x$  is a feasible solution for the linear relaxation of  $(P_{PCKP1})$ .

In the matter of constraints, each subset  $A$  cannot contain any element from the top layer. This is because we are interested only in ideals, and if  $A$  contains an element  $e_i^2$  from the top layer, it must also contain the corresponding element  $e_i^1$  which is located below of  $e_i^2$ . But then  $D(A) < 0$  and the corresponding constraint will be trivially satisfied, because  $D(A)$  will be set equal to zero based on its definition. For the same reason, subset  $A$  cannot contain  $\frac{n}{2}$  or more elements from the bottom layer, because once again  $D(A)$  will be equal to zero. Therefore the only constraints that we should

check for satisfiability, are the ones whose the corresponding subset  $A$  contains less than  $\frac{n}{2}$  elements from the bottom layer.

Consider an arbitrary subset  $A$  that contains the indices of elements only from the bottom layer, with size less than  $\frac{n}{2}$ , that is,  $|A| < \frac{n}{2}$ . The corresponding constraint will be the following:

$$\sum_{1 \leq i \leq n: i \notin A} \bar{u}_i^1(A) \cdot x_i^1 + \sum_{1 \leq i \leq n: i \in A} \bar{u}_i^2(A) \cdot x_i^2 \geq D(A)$$

However from the definition of  $\bar{u}_i(A)$ , it holds that  $(\forall i)(1 \leq i \leq n) : \bar{u}_i^1(A) = \bar{u}_i^2(A) = D(A)$  and the corresponding constraint is equivalent to:

$$\begin{aligned} \sum_{1 \leq i \leq n: i \notin A} x_i^1 + \sum_{1 \leq i \leq n: i \in A} x_i^2 &\geq 1 && \Leftrightarrow \\ \sum_{1 \leq i \leq n: i \notin A} \frac{2}{n} + \sum_{1 \leq i \leq n: i \in A} 0 &\geq 1 && \Leftrightarrow \\ (n - |A|) \cdot \frac{2}{n} &\geq 1 && \Leftrightarrow \\ |A| &\leq \frac{n}{2} \end{aligned}$$

And since  $|A| \leq \frac{n}{2}$ , the corresponding constraint will be satisfied. As a result, all the constraints are satisfied and the solution  $x$  is a feasible one. The cost of the fractional solution  $x$  is equal to:

$$\begin{aligned} cost(x) &= \sum_{i=1}^n x_i^1 \cdot 1 + \sum_{i=1}^n x_i^2 \cdot \frac{n}{2} && \Rightarrow \\ cost(x) &= n \cdot \frac{2}{n} \cdot 1 + n \cdot 0 \cdot \frac{n}{2} && \Rightarrow \\ cost(x) &= 2 \end{aligned}$$

Therefore for the gap it holds that:

$$\frac{OPT}{cost(x)} = \frac{\frac{n}{2}}{2} = \frac{n}{4}$$

And equivalently we can say that the gap is  $\Omega(n)$ . As the width of the partial order is equal to  $n$ , that is,  $w(\mathcal{P}) = n$ , we can conclude that the gap between the optimal integer solution of  $(P_{PCKP})$  with the optimal linear solution of the linear relaxation of  $(P_{PCKP1})$  is  $\Omega(w(\mathcal{P}))$ .  $\square$

**Lemma 3.2.** The integrality gap of  $(P_{PCKP1})$  is  $\Omega(w(\mathcal{P}))$ .

*Proof.* Consider the same instance of PCKP as before. Based on the previous theorem, it suffices to show that the optimal integral solution of  $(P_{PCKP1})$  for that instance cannot have cost less than  $\frac{n}{2}$ .

Let assume the opposite, that there is a feasible integer solution  $x$  with cost less than  $\frac{n}{2}$ . By  $S = \{i \mid x_i = 1\}$  we denote the support vector of  $x$ . Because of the cost of  $S$ , we can conclude that  $S$  cannot contain any element from the top layer and also must

contain less than  $\frac{n}{2}$  elements from the bottom layer. Otherwise the cost of the solution would be equal to  $\frac{n}{2}$  or more, contradicting our assumption.

From the construction,  $S$  is an ideal and so the corresponding constraint for  $A = S$  must be satisfied. Since each element of  $A$  has value equal to one and also  $|A| < \frac{n}{2}$ , it is true that  $D(A) > 0$ . However in the corresponding constraint, the left side of the inequality sums up to zero as we have picked elements only from the set  $S$ . Therefore the inequality is not satisfied and so  $x$  cannot be a feasible integer solution.

Consequently the optimal integer solution of  $(P_{PCKP1})$  for that instance has cost at least  $\frac{n}{2}$  and the integrality gap of  $(P_{PCKP1})$  is  $\Omega(w(\mathcal{P}))$ .  $\square$

Therefore the two results above, tells us that we cannot use this formulation, in order to develop an algorithm with guarantee better than  $\Omega(w(\mathcal{P}))$ . Furthermore from the family of the instances that we have used, we can also conclude that the bound is  $\Omega(|U|)$  as well. This bound is stronger as it is always true that  $w(\mathcal{P}) \leq |U|$ . Hence we can arrive at the following corollary.

**Corollary 3.3.** The integrality gap of  $(P_{PCKP1})$  is  $\Omega(|U|)$ .

### 3.3 Pitch-1 inequalities

In this section we are going to investigate the simpler inequalities for the PCKP problem. These inequalities are also called pitch-1 inequalities [1]. Formally an inequality of the form  $\sum_{i \in U} u_i \cdot x_i \geq D$  is of *pitch* 1 if and only if, every  $u_i$  is at least  $D$ . In our case, the variable  $D$  and all the  $u_i$  will be equal to one.

These simpler inequalities were also mentioned in the previous section (3.1). To recap, consider an ideal  $A$  whose residual demand  $D(A)$  has not yet covered. These inequalities do not take into consideration the values of the elements, but pick any element from the set  $\min \mathcal{P}(A)$ . This idea can be written as a linear inequality by saying that the sum of the elements  $i \in \min \mathcal{P}(A)$  should be at least one. To simplify the notations below, let us define the set  $\mathcal{L}'(\mathcal{P}) = \{A \in \mathcal{L}(\mathcal{P}) \mid D(A) > 0\}$  of all ideals whose residual demand with respect to  $A$  is not yet covered.

The resulting integer program is the following:

$$\begin{aligned} & \text{minimize} && \sum_{i \in U} c_i \cdot x_i \\ & \text{subject to} && \sum_{i \in \min \mathcal{P}(A)} x_i \geq 1, \quad \forall A \in \mathcal{L}'(\mathcal{P}) \\ & && x \in \{0, 1\}^{|U|} \quad (P_{PCKP2}) \end{aligned}$$

And the dual of the linear relaxation of  $(P_{PCKP2})$  is defined as:

$$\begin{aligned} & \text{maximize} && \sum_{A \in \mathcal{L}'(\mathcal{P})} y(A) \\ & \text{subject to} && \sum_{A \in \mathcal{L}'(\mathcal{P}): i \in \min \mathcal{P}(A)} y(A) \leq c_i, \quad \forall i \in U \\ & && y(A) \geq 0, \quad \forall A \in \mathcal{L}'(\mathcal{P}) \quad (D_{PCKP2}) \end{aligned}$$

Even though we have already given the intuition that the new integer program is a valid relaxation for PCKP, the next theorem is a formal proof.

**Theorem 3.4.**  $(P_{PCKP2})$  is a valid relaxation for PCKP, that is, any integer feasible solution in  $(P_{PCKP})$  is feasible in  $(P_{PCKP2})$ .

*Proof.* Let us consider an integer feasible solution  $x$  of  $(P_{PCKP})$ . Let us also consider an arbitrary ideal  $A \in \mathcal{L}(\mathcal{P})$ , such that  $D(A) > 0$ . We will show that the corresponding constraint of  $A$  in  $(P_{PCKP2})$  will be satisfied.

Since  $x$  is feasible in  $(P_{PCKP})$ , we know that  $\sum_{i \in U \setminus A} u_i(A) \cdot x_i \geq D(A)$ . As  $x$  is an integer solution there must exist an  $i \in U \setminus A$  such that  $x_i = 1$ , otherwise the sum will be zero. Moreover, since  $x$  must be closed under  $\mathcal{P}$ , there must exist an  $i \in U \setminus A$  which is minimal in  $\mathcal{P}(A)$ , or in other words  $i \in \min \mathcal{P}(A)$ , such that  $x_i = 1$ . As a result  $x$  satisfies the corresponding constraint in  $(P_{PCKP2})$  which is the  $\sum_{i \in \min \mathcal{P}(A)} x_i \geq 1$  and thus  $x$  is a feasible integer solution in  $(P_{PCKP2})$ .  $\square$

As  $(P_{PCKP2})$  is a valid relaxation for the PCKP, every feasible dual linear solution of  $D_{PCKP2}$  gives a lower bound on the optimal integer solution of PCKP. Therefore, we could run the greedy Algorithm 1, using this new formulation and end up with a feasible  $(S, y)$  pair. The proof of feasibility follows the same logic as in Lemma 4 of [12]. In the next theorem, we prove that in the worst case the gap for this feasible pair is at most  $w(\mathcal{P})$ .

**Theorem 3.5.** The cost of a solution that is found by the greedy Algorithm 1 using the  $(P_{PCKP2})$  formulation, is at most  $w(\mathcal{P}) \cdot OPT$ .

*Proof.* Let  $(S, y)$  be a feasible pair, where  $S$  is the set of elements that the algorithm has picked and  $y$  is the dual vector that the algorithm has produced. The total cost of  $S$  can be evaluated as follows:

$$\begin{aligned}
 cost(S) &= \sum_{i \in S} c_i = \sum_{i \in S} \sum_{A \in \mathcal{L}'(\mathcal{P}): i \in \min \mathcal{P}(A)} y(A) \\
 &= \sum_{A \in \mathcal{L}'(\mathcal{P})} y(A) \sum_{i \in S \cap \min \mathcal{P}(A)} 1 \\
 &= \sum_{A \in \mathcal{L}'(\mathcal{P})} y(A) \cdot |S \cap \min \mathcal{P}(A)| \\
 &\leq \max_{A \in \mathcal{L}'(\mathcal{P})} \{|S \cap \min \mathcal{P}(A)|\} \cdot \sum_{A \in \mathcal{L}'(\mathcal{P})} y(A) \\
 &\leq w(\mathcal{P}) \cdot \sum_{A \in \mathcal{L}'(\mathcal{P})} y(A) \\
 &\leq w(\mathcal{P}) \cdot OPT
 \end{aligned}$$

$\square$

As a result, the previous theorem shows that it is not necessary to use the more complex inequalities in order to develop a  $w(\mathcal{P})$ -approximation primal-dual algorithm. Unfortunately, the simpler pitch-1 inequalities of the form  $\sum_{i \in \min \mathcal{P}(A)} x_i \geq 1, \forall A \in \mathcal{L}'(\mathcal{P})$  would be redundant in the  $P_{PCKP1}$  as they are implied from the inequalities of the form  $\sum_{i \in \min \mathcal{P}(A)} \bar{u}_i(A) \cdot x_i \geq D(A), \forall A \in \mathcal{L}(\mathcal{P})$ , and so by adding them to the  $(P_{PCKP1})$  will not strengthen it.

**Lemma 3.6.** Inequalities of the form  $\sum_{i \in \min \mathcal{P}(A)} x_i \geq 1, \forall A \in \mathcal{L}'(\mathcal{P})$  are redundant in the  $P_{PCKP1}$ .

*Proof.* Consider a feasible linear solution  $x$  and an ideal  $A$  such that  $D(A) > 0$ . Let us assume that  $x$  satisfies the corresponding inequality:

$$\sum_{i \in \min \mathcal{P}(A)} \bar{u}_i(A) \cdot x_i \geq D(A)$$

but it does not satisfy the corresponding simpler inequality:

$$\sum_{i \in \min \mathcal{P}(A)} x_i \geq 1$$

Because  $D(A)$  is positive, this would imply that  $x$  does not satisfy the inequality:

$$\sum_{i \in \min \mathcal{P}(A)} D(A) \cdot x_i \geq D(A)$$

From the definition of  $\bar{u}_i(A)$ , we know that  $\bar{u}_i(A) \leq D(A)$  and thus, we can conclude that:

$$\sum_{i \in \min \mathcal{P}(A)} \bar{u}_i(A) \cdot x_i \leq \sum_{i \in \min \mathcal{P}(A)} D(A) \cdot x_i < D(A)$$

which contradicts our initial assumption.  $\square$

From the previous lemma, we can understand that by adding the simpler pitch-1 inequalities to the  $(P_{PCKP1})$ , the integrality gap is not reduced, but in the worst case both types of inequalities seem equivalent.

However as [12] explains in Theorem 3, in the case that for each element and each ideal the total value which is projected to an element with respect to the ideal  $A$  is bounded by the real value of the element with respect to  $A$  multiplied by a constant  $\alpha$ , that is, assuming that:

$$\bar{u}_i(A) \leq \alpha \cdot u_i(A)$$

holds for every  $A \in \mathcal{L}(\mathcal{P})$  and  $i \in \min \mathcal{P}(A)$ , then the primal-dual algorithm constructs a solution of cost at most  $2\alpha \cdot OPT$ . The reason is that in the analysis of the cost, the value  $\bar{u}_i(A)$  can be replaced by the value  $\alpha \cdot u_i(A)$  and as  $u_i(A) \leq u_i$ , we can analyze the cost similarly with the case of the minimum knapsack problem without the precedence constraints. Without taking into account the values of the elements and using only the simpler pitch-1 inequalities, it is unlikely to get a similar result. That is because in the case that  $\mathcal{P}$  is an antichain, PCKP reduces to the minimum knapsack problem, and choosing elements in an arbitrary way without taking into consideration the values, is not effective in terms of the approximation factor. Therefore in this context the precedence knapsack cover inequalities seem stronger than the simpler pitch-1 inequalities.

### 3.4 Rounding algorithm for 0-1 PCKP

We have already said that for the 0-1 PCKP, McCormick et al. [12] developed a primal-dual  $w(\mathcal{P})$ -approximation algorithm. In this section though, we are going to apply a simple rounding technique that was also used by Carr et al. [3] for the 0-1 CIP problem, in order to develop a rounding algorithm with the same approximation factor  $w(\mathcal{P})$ . Even though there is no progress in terms of the approximation guarantee, maybe in the future through rounding techniques we could get better results for the PCKP problem. Initially let us present a very useful lemma for the development of the algorithm.

**Lemma 3.7.** Let  $x$  be an integer feasible solution of  $(P_{PCKP1})$  which is not closed under  $\mathcal{P}$ . Then there is an integer feasible solution  $y$  in  $(P_{PCKP1})$  which is closed under  $\mathcal{P}$  and the cost of  $y$  is less than the cost of  $x$ .

*Proof.* Let us consider an integer feasible solution  $x$  of  $(P_{PCKP1})$  with support  $S = \{i \mid x_i = 1\}$  and also an initially empty set  $A = \emptyset$ .

We will augment the set  $A$  as follows. At each step, we pick an element from the set  $\min \mathcal{P}(A)$  which is also part of the solution set  $S$ . That is, we pick an element  $i \in \min \mathcal{P}(A) \cap S$  and update  $A = A \cup \{i\}$ . As  $x$  is not closed under  $\mathcal{P}$ , there must be a step  $k$  for which the previous intersection is empty and let  $A_k$  be the current set for this step.  $A_k$  is an ideal because at any previous step until  $k$  we have augmented the set only by minimal items and so, there exist a corresponding constraint for the set  $A_k$  in the  $(P_{PCKP1})$ . However, the left side of the corresponding constraint sums to zero because none of the minimal elements of  $\mathcal{P}(A_k)$  belong to  $S$ . Therefore, as  $x$  is an integer feasible solution of  $(P_{PCKP1})$ , it must be true that  $D(A_k) \leq 0$ .

We can now build our new integer feasible solution  $y$  by setting  $y_i = 1$  if and only if  $i \in A_k$ . The cost of  $y$  is indeed less than the cost of  $x$  because the support  $A_k$  of  $y$  is a proper subset of the support  $S$  of  $x$ . Finally, as  $y$  satisfies the demand and is closed under the partial order, it is a feasible integer solution of  $(P_{PCKP})$  and hence, a feasible integer solution of  $(P_{PCKP1})$  as well.  $\square$

With the help of this lemma, we can now proceed with the algorithm. The main idea of the algorithm is to solve the linear relaxation of  $(P_{PCKP1})$  and set each variable  $x_i$  of the linear solution vector to 1 if and only if  $x_i \geq \frac{1}{w(\mathcal{P})}$ . However we must ensure that the linear solution will be closed under  $\mathcal{P}$ , as we want to be sure that the produced integer solution will be closed as well. To handle this problem, we can either add the polynomially many natural constraints of  $(P_{PCKP})$  that allow only closed solutions or use ideas from the previous lemma, and this is what we will eventually do.

Another big obstacle is that the linear relaxation of  $(P_{PCKP1})$  has exponentially many constraints. A very useful idea that was introduced by [3] and later also used by [13, 11], is to use the ellipsoid method and at each iteration to satisfy a subset of the constraints. Eventually we end up with a relaxed linear solution, which is of course a lower bound on the optimal integer solution. In our case we will separate over one inequality which can be found in polynomial time.

Consider a candidate linear solution  $\hat{x}$  and let us now describe the inequality that we will use in the separation oracle. The procedure in order to find out the proper constraint is similar with the previous lemma. In particular, we start from an empty set  $A = \emptyset$  and as long as there is an element  $i \in \min \mathcal{P}(A)$  with  $\hat{x}_i \geq \frac{1}{w(\mathcal{P})}$ , we augment the set  $A$  by inserting in it the element  $i$ , that is, the set  $A$  is updated to  $A = A \cup \{i\}$ . In the case that no such element exists with the previous properties, we stop the procedure. The constructed  $A$  is an ideal, and so the corresponding inequality:

$$\sum_{i \in \min \mathcal{P}(A)} \bar{u}_i(A) \cdot \hat{x}_i \geq D(A)$$

is part of the  $(P_{PCKP1})$  formulation. This is the inequality that we give to the separation oracle for the candidate solution  $\hat{x}$ . Clearly this procedure of finding such a constraint runs in polynomial time. This is an iteration of the algorithm and as long as  $\hat{x}$  does not satisfy the inequality, the ellipsoid method continues by suggesting a new linear solution and repeating the whole procedure of constructing the corresponding constraint. In the case though that  $\hat{x}$  satisfies the constructed inequality, the algorithm stops. Let us denote



by  $\bar{x}$  the final candidate solution  $\hat{x}$  which was used by the algorithm in the last iteration. Also let  $\bar{A}$  be the final constructed set which was constructed for the final candidate solution  $\hat{x}$ . The next lemma shows a property for  $\bar{x}$  and  $\bar{A}$  respectively.

**Lemma 3.8.** The solution  $\bar{x}$  is a lower bound on the optimal integer solution of the PCKP instance. Also it holds that  $D(\bar{A}) \leq 0$ .

*Proof.* The solution  $\bar{x}$  is a lower bound on the optimal integer solution of the PCKP instance because it is minimized over a subset of constraints of a linear programming relaxation of the PCKP.

Let us suppose on the contrary that  $D(\bar{A}) > 0$ . Because the algorithm stopped, the corresponding constraint for the ideal  $\bar{A}$  must be satisfied from the final solution  $\bar{x}$ . Thus it must hold that:

$$\sum_{i \in \min \mathcal{P}(\bar{A})} \bar{u}_i(\bar{A}) \cdot \bar{x}_i \geq D(\bar{A})$$

Under the condition that the instance is feasible, the left hand side must have at least one variable, because otherwise all the elements of the instance will have been included in the solution and still the demand will not be covered. Hence it should hold that:

$$\sum_{i \in \min \mathcal{P}(\bar{A})} \bar{u}_i(\bar{A}) \cdot \bar{x}_i < \sum_{i \in \min \mathcal{P}(\bar{A})} \bar{u}_i(\bar{A}) \cdot \frac{1}{w(\mathcal{P})} \leq \sum_{i \in \min \mathcal{P}(\bar{A})} D(\bar{A}) \cdot \frac{1}{w(\mathcal{P})} \leq D(\bar{A})$$

which is a contradiction, because we have assumed that  $\bar{x}$  satisfies this constraint. As a result, the residual demand  $D(\bar{A})$  cannot be positive.  $\square$

With the previous lemma we can now proceed in the next theorem, where we will show that we can construct a feasible integer solution  $y$  with cost at most  $w(\mathcal{P})$  times the cost of  $\bar{x}$ .

**Theorem 3.9.** There exists a  $w(\mathcal{P})$ -approximation rounding algorithm for the 0-1 PCKP.

*Proof.* Run the previous algorithm and set  $y_i = 1$  if and only if  $i \in \bar{A}$ . The solution  $y$  is closed under  $\mathcal{P}$  and from Lemma 3.8 the solution  $y$  satisfies the demand. Therefore  $y$  is an integer feasible solution. Moreover the cost of  $y$  is at most  $w(\mathcal{P})$  times the cost of  $\bar{x}$  because we have set  $y_i = 1$  only if  $\bar{x}_i \geq \frac{1}{w(\mathcal{P})}$ . Therefore as  $\bar{x}$  is a lower bound on the optimal integer solution of the PCKP instance, we can conclude that the cost of the solution  $y$  is at most  $w(\mathcal{P})$  times the cost of the optimal integer solution.  $\square$

Consequently from the above theorem we can conclude that there exists a rounding algorithm for the 0-1 PCKP problem with the same approximation factor with the previous primal-dual algorithm, which is equal to  $w(\mathcal{P})$ , the size of the maximum antichain.

### 3.5 PCKP with general multiplicity constraints

The previous algorithms that we have discussed, solve the PCKP problem only for the 0-1 case. For the knapsack problem without precedence constraints we can extend the results for general multiplicity constraints as well [3], [12]. However for general multiplicity constraints with a partial order, based on the Proposition 2 of McCormick et al. [12] a modified algorithm would fail to give us a bounded approximation ratio. In their Proposition 3 for the general multiplicity constraints, they suggest a pseudo-polynomial algorithm with approximation ratio equal to  $w(\mathcal{P}) \cdot \Delta$ , where  $\Delta = \max_j d_j$  is an upper

bound on the multiplicity variables. As they write, it is an open problem to provide strongly polynomial bounds for this problem. Even though a polynomial algorithm with the same approximation ratio seems to exist by using the modified algorithm, in this section we present a polynomial algorithm with strongly polynomial bounds and specifically with approximation ratio equal to  $O(\min\{|U|, w^2(\mathcal{P})\})$ .

Formally the PCKP problem with general bounds can be described as an integer program as follows:

$$\begin{aligned} & \text{minimize} && \sum_{i \in U} c_i \cdot x_i \\ & \text{subject to} && \sum_{i \in U} u_i \cdot x_i \geq D \\ & && x_i - x_j \geq 0, \forall i \preceq j \\ & && x_i \leq d_i, \forall i \in U \\ & && x \in \mathbb{Z}_+^{|U|} \end{aligned}$$

The next lemma is an auxiliary lemma that will be used later on, in order to bound some sums.

**Lemma 3.10.** Consider a finite partially ordered set  $\mathcal{P}$  such that each element has a value  $v_i$ . Denote by  $\alpha$  the upper bound of every possible sum of elements of a chain. Then, the sum of the values of all the elements of  $\mathcal{P}$  is at most  $\alpha \cdot w(\mathcal{P})$ .

*Proof.* From Dilworth's theorem [4], there is a chain decomposition  $\{S_1, \dots, S_{w(\mathcal{P})}\}$  of size equal to  $w(\mathcal{P})$ . The total sum of elements can be bounded in the following way:

$$\sum_{i \in \mathcal{P}} v_i \leq \sum_{S_j: \text{chain of } \mathcal{P}} \sum_{i \in S_j} v_i \leq \sum_{S_j: \text{chain of } \mathcal{P}} \alpha \leq \alpha \cdot w(\mathcal{P})$$

□

Now we are going to present an  $O(w^2(\mathcal{P}))$ -approximation primal-dual algorithm for the PCKP problem with  $d \geq 1$ . In the previous formulations, we were focusing on the minimal items and each non minimal item would push its value to the minimal elements which were comparable with it. The idea now is to be able to pick at each iteration any item and not only the minimals. In order to keep the solution closed under the partial order, when an element is chosen, we force the program to pick all the other elements that are below it with respect to the partial order. With this idea in mind, let us give the following definitions:

$$\begin{aligned} \underline{c}_i &= \sum_{j \preceq i} c_j \\ D(A) &= D - \sum_{i \in A} d_i \cdot u_i \\ \underline{u}_i(A) &= \min \left\{ \sum_{j \preceq i \wedge j \notin A} u_j(A), D(A) \right\} \end{aligned}$$

The new valid relaxation that we suggest is the following one:

$$\begin{aligned}
 & \text{minimize} && \sum_{i \in U} c_i \cdot x_i \\
 & \text{subject to} && \sum_{i \in U \setminus A} u_i(A) \cdot x_i \geq D(A), \forall A \in \mathcal{L}(\mathcal{P}) \\
 & && x_i \geq 0, \forall i \in U \quad (P_{PCKP3})
 \end{aligned}$$

and the dual of the linear relaxation is:

$$\begin{aligned}
 & \text{maximize} && \sum_{A \in \mathcal{L}(\mathcal{P})} y(A) \cdot D(A) \\
 & \text{subject to} && \sum_{A \in \mathcal{L}(\mathcal{P}): i \in U \setminus A} u_i(A) \cdot y(A) \leq c_i, \forall i \in U \\
 & && y(A) \geq 0, A \in \mathcal{L}(\mathcal{P}) \quad (D_{PCKP3})
 \end{aligned}$$

An interesting observation here is that there are no multiplicity constraints in the formulation. However the algorithm that we develop respects the multiplicity constraints and as the formulation is a valid relaxation for the problem, we can apply the standard analysis of the cost. Moreover the cost function of the new formulation could be problematic because in the cost of each element  $i$ , the cost of every element which is below of  $i$  in the partial order, is also included. Nevertheless, the following lemma shows that for each feasible solution of the PCKP, we can find a feasible solution of the new formulation with a cost that is larger only by a factor of  $w(\mathcal{P})$ .

**Lemma 3.11.** Let  $x^2$  be a feasible solution for PCKP. Then there exists a feasible solution  $x^1$  of  $(P_{PCKP3})$  such that  $\text{cost}(x^1) \leq w(\mathcal{P}) \cdot \text{cost}(x^2)$ .

*Proof.* Set  $x_i^1 = x_i^2 - \max_{i \prec_j} x_j^2$ . The solution  $x^1$  is feasible in the  $(P_{PCKP3})$  because for each element  $i$ , the corresponding value  $u_i$  has been counted at least  $x_i^2$  times. The reason is the following. Inductively with base case the maximal elements, consider for an element  $i$ , the element  $k = \arg \max_{i \prec_j} x_j^2$ . From the inductive hypothesis,  $u_k$  has been counted at least  $x_k^2$  times, and from the transitive relation of the partial order and the definition of  $u$ , the  $u_i$  has been counted at least  $x_k^2$  times as well. Hence by setting  $x_i^1$  as the difference  $x_i^2 - x_k^2$ , the  $u_i$  is counted in total at least  $x_i^2$  times, and this proves the feasibility of  $x^1$  in the  $(P_{PCKP3})$ . For the cost of  $x^1$  it holds that:

$$\begin{aligned}
 \text{cost}(x^1) &= \sum_{i \in U} x_i^1 \cdot c_i &= \\
 & \sum_{i \in U} x_i^1 \cdot \sum_{j: j \preceq i} c_j &= \\
 & \sum_{j \in U} c_j \cdot \sum_{i: j \preceq i} x_i^1 &\leq \\
 & \sum_{j \in U} c_j \cdot x_j^2 \cdot w(\mathcal{P}) &= \\
 & w(\mathcal{P}) \cdot \text{cost}(x^2)
 \end{aligned}$$

To explain the reason that the inequality holds, consider for a fixed element  $j$  the restricted partial order that contains only the elements  $i$  such that  $j \preceq i$ . Assuming that

the value of each element  $i$  in this partial order is equal to  $x_i^1$ , then from the construction the sum of each chain is at most  $x_j^2$ . To prove this, let us follow the same inductive argument as we did before with the feasibility of  $x^1$ . Consider for an element  $j$ , the element  $k = \arg \max_{j \prec i} x_i^2$  and assume that until the element  $k$ , the maximum sum from  $k$  until a maximal element, is at most  $x_k^2$ . By setting  $x_j^1$  as the difference  $x_j^2 - x_k^2$ , the maximum sum of a chain starting from  $j$  to a maximal element, is at most  $x_j^2$ , and this concludes our argument. Therefore from Lemma 3.10 it holds that  $\sum_{i:j \preceq i} x_i^1 \leq x_j^2 \cdot w(\mathcal{P})$ .  $\square$

Therefore we can conclude that for an instance, the cost of the optimal solution of the  $(P_{PCKP3})$  is at most  $w(\mathcal{P})$  times larger than the cost of the optimal solution of the PCKP.

Consider the following primal-dual algorithm which is running over the  $(D_{PCKP3})$  dual linear program. The idea is to keep track of two sets of elements, the set  $S$  contains all the elements that we will pick and the set  $S'$  contains all the elements that their corresponding inequality has become tight. From the solution  $x$  that is produced, we retrieve another feasible solution  $x^1$  with the same logic that we applied in Lemma 3.11. This is because the cost of the final solution  $x^2$  that we are interested in, can be bounded from the cost of  $x^1$  which can be bounded by the produced dual linear solution  $y$ .

---

$S = \emptyset, y = 0$

```

while (D(S) > 0 && |S| < n) {
  Increase y(S) until a dual constraint becomes tight for item i
  S = S ∪ {j | j ≼ i}
  S' = S' ∪ {i}
  x_i = d_i
}

```

Let  $A$  be equal to  $S$  without the elements which were inserted in  $S$  in the last iteration. Update  $x_l$  to be equal to the minimum value such that  $u_l(A)$  covers the residual demand, where  $l$  is the index of the last inserted element in the set  $S'$ .

$x_i^1 = \max\{x_i - \max_{i \prec j} x_j, 0\}$   
 $x_i^2 = \max_{i \preceq j} x_j$

---

**Theorem 3.12.** There exists a  $w^2(\mathcal{P})$ -approximation primal-dual algorithm for the PCKP with  $d \geq 1$ .

*Proof.* After the execution of the algorithm we end up with an integer feasible solution  $x^2$ , as it is related to  $S$ . The cost of  $x^2$  can be bounded by the cost of  $x^1$  in terms of the  $(P_{PCKP3})$ , because we can use the proof of Lemma 3.11, to show that the corresponding costs have been counted at least the same times. However note that with the same argument we can show that  $x^1$  is feasible in the  $(P_{PCKP3})$ , even if it is not necessary in the analysis of the cost. Let  $S_1 = \{i \mid x_i^1 > 0\}$  be the support vector of  $x^1$ . The cost of  $x^2$  can be evaluated as follows:

$$\text{cost}(x^2) \leq \underline{\text{cost}}(x^1) = \sum_{i \in S_1} x_i^1 \cdot c_i =$$

$$\begin{aligned}
 & \sum_{i \in S_1} x_i^1 \cdot \sum_{A \in \mathcal{L}(\mathcal{P}): i \notin A} \underline{u}_i(A) \cdot y(A) = \\
 & \sum_{A \in \mathcal{L}(\mathcal{P})} y(A) \cdot \sum_{i \in S_1 \setminus A} x_i^1 \cdot \underline{u}_i(A) = \\
 & \sum_{A \in \mathcal{L}(\mathcal{P})} y(A) \cdot \left( \sum_{i \in S_1 \setminus A \wedge i \neq l} x_i^1 \cdot \underline{u}_i(A) + x_l^1 \cdot \underline{u}_l(A) \right) = \\
 & \sum_{A \in \mathcal{L}(\mathcal{P})} y(A) \cdot \left( \left( \sum_{i \in S_1 \setminus A \wedge i \neq l} x_i^1 \cdot \sum_{j \preceq i \wedge j \notin A} u_j(A) \right) + x_l^1 \cdot \underline{u}_l(A) \right) = \\
 & \sum_{A \in \mathcal{L}(\mathcal{P})} y(A) \cdot \left( \left( \sum_{j \notin A \wedge \exists i \in S_1 \setminus A \setminus \{l\}: j \preceq i} u_j(A) \cdot \sum_{i \in S_1 \setminus A \setminus \{l\}: j \preceq i} x_i^1 \right) + x_l^1 \cdot \underline{u}_l(A) \right) \leq \quad (1) \\
 & \sum_{A \in \mathcal{L}(\mathcal{P})} y(A) \cdot \left( \left( \sum_{j \notin A \wedge \exists i \in S_1 \setminus A \setminus \{l\}: j \preceq i} u_j(A) \cdot d_j \cdot w(\mathcal{P}) \right) + x_l^1 \cdot \underline{u}_l(A) \right) \leq \quad (2) \\
 & \sum_{A \in \mathcal{L}(\mathcal{P})} y(A) \cdot (w(\mathcal{P}) \cdot D(A) + 2 \cdot D(A)) = \\
 & (w(\mathcal{P}) + 2) \cdot \sum_{A \in \mathcal{L}(\mathcal{P})} y(A) \cdot D(A) \leq \\
 & (w(\mathcal{P}) + 2) \cdot \underline{OPT} \leq \quad (3) \\
 & O(w^2(\mathcal{P})) \cdot \underline{OPT}
 \end{aligned}$$

In inequality (1), consider the restricted  $\mathcal{P}$  for a fixed element  $j$ . Then the sum of the values of elements of a chain is at most  $d_j$ , and so we apply Lemma 3.10. From the last step for the  $x_l$  variable, we can conclude that  $(x_l^1 - 1) \cdot \underline{u}_l(A) < D(A)$ , and so inequality (2) follows. Also inequality (3) holds, because of Lemma 3.11.  $\square$

Let us now present the  $O(|U|)$ -approximation primal-dual algorithm for the PCKP problem with general multiplicity constraints. Consider the following valid formulation of the problem, which is the same with  $(P_{PCKP3})$ , but now instead we are making use of the real values of the elements:

$$\begin{aligned}
 & \text{minimize} \quad \sum_{i \in U} \underline{c}_i \cdot x_i \\
 & \text{subject to} \quad \sum_{i \in U \setminus A} u_i(A) \cdot x_i \geq D(A), \quad \forall A \in \mathcal{L}(\mathcal{P}) \\
 & \quad \quad \quad x_i \geq 0, \quad \forall i \in U \quad (P_{PCKP4})
 \end{aligned}$$

and the dual of the linear relaxation is:

$$\begin{aligned}
 & \text{maximize} \quad \sum_{A \in \mathcal{L}(\mathcal{P})} y(A) \cdot D(A) \\
 & \text{subject to} \quad \sum_{A \in \mathcal{L}(\mathcal{P}): i \in U \setminus A} u_i(A) \cdot y(A) \leq \underline{c}_i, \quad \forall i \in U \\
 & \quad \quad \quad y(A) \geq 0, \quad A \in \mathcal{L}(\mathcal{P}) \quad (D_{PCKP4})
 \end{aligned}$$

**Theorem 3.13.** There exists an  $O(|U|)$ -approximation primal-dual algorithm for the PCKP with  $d \geq 1$ .

*Proof.* We can run the previous algorithm with or without the last pruning step. The produced solution  $x^2$  is feasible to the PCKP with  $d \geq 1$  and its cost is bounded by the cost of  $x^1$  in terms of the  $(P_{PCKP4})$ . However the cost of  $x^1$  is at most two times larger than the cost of the produced dual linear solution because the analysis is similar to the one in Lemma 2.4, for the knapsack without precedence constraints. Therefore it holds that:

$$\text{cost}(x^2) \leq \text{cost}(x^1) \leq 2 \cdot \text{OPT}$$

Unfortunately Lemma 3.11 is not applicable here because the new formulation uses the real values of the elements. In this situation, a feasible solution of PCKP is feasible in the  $(P_{PCKP4})$  with cost at most  $|U|$  times larger than before. The reason is that the cost of each element  $i$  will be included in at most  $|U|$  elements above it, and also every element above  $i$  can be selected at most  $d_i$  times. Thus we can conclude that  $\text{OPT} \leq |U| \cdot \text{OPT}$ . Consequently for the cost of  $x^2$  it is true that:

$$\text{cost}(x^2) \leq O(|U|) \cdot \text{OPT}$$

□

An interesting observation is that the produced  $x^1$  solution is not necessarily a feasible solution of the  $(P_{PCKP4})$ , but this does not cause us any problem. This is because we are interested in the  $x^2$  solution and as the produced dual linear solution remains feasible in the  $D_{PCKP4}$ , we are able to apply the usual analysis of the linear programming theory.

Eventually combining the two theorems above, we can get the main following result which gives us strongly polynomial bounds for the general PCKP problem.

**Theorem 3.14.** There exists an  $O(\min\{|U|, w^2(\mathcal{P})\})$ -approximation algorithm for the PCKP problem with general multiplicity constraints.

## 3.6 Inapproximability of PCKP

McCormick et al. [12] also proved an inapproximability result for the 0-1 PCKP which can be seen as a lower bound for the problem. Apparently as 0-1 PCKP is a special case of the PCKP, this lower bound is inherited to the general case as well. In their proof, they present a reduction from the densest  $k$ -subgraph to the  $l$ -EIS and from  $l$ -EIS to the PCKP, and because Khot [10] showed that there is no PTAS for the densest  $k$ -subgraph unless  $\text{NP} \subseteq \cap_{\epsilon > 0} \text{BPTIME}(2^{n^\epsilon})$ , they come to the conclusion that PCKP does not admit a PTAS unless  $\text{NP} \subseteq \cap_{\epsilon > 0} \text{BPTIME}(2^{n^\epsilon})$ . However in their proof of Theorem 6, some details are missing. In general the proof can help us to build an intuition about the differences of the PCKP with the simple knapsack problem that makes the first one harder than the second one. For these two reasons, we decided to present the proof once again, with minor changes that will be pointed out. First of all, let us define the two new problems that will be used in the proof.

**Definition 3.15.** *Densest  $k$ -subgraph* is a maximization problem, in which we are given a graph  $G = (V, E)$  and a parameter  $k$  and we are asked to select a subset of vertices  $S \subseteq V$ , such that  $|S| \leq k$  and the number of edges in the induced subgraph  $G[S]$  is maximized.

**Definition 3.16.**  *$l$ -EIS* is a minimization problem, in which we are given a graph  $G = (V, E)$  and a parameter  $l$  and we are asked to select a subset of edges  $S \subseteq E$ , such that  $|S| \geq l$  and the number of vertices in the induced subgraph  $G[S]$  is minimized.

The aim of the proof is to reduce an instance of the densest  $k$ -subgraph problem to a PCKP instance. An issue here though, is that the densest  $k$ -subgraph is a maximization problem, while the PCKP is a minimization problem. However, instead of trying to select vertices in order to maximize the number of induced edges, we could try to select edges in order to minimize the number of induced vertices and the new problem that arises is exactly the  $l$ -EIS. In Theorem 5 of [12] they give a simple reduction from an  $l$ -EIS instance to a PCKP instance with the same cost, which means that these two problems can be seen as equivalent. Therefore, it is sufficient to prove that if we had a PTAS for  $l$ -EIS, then we would also have a PTAS for the densest  $k$ -subgraph.

**Theorem 3.17.** (Theorem 6 [12]) For any  $\epsilon > 0$ , an  $(1+\epsilon)$ -approximation algorithm for  $l$ -EIS can be used as an  $(1 - 2 \cdot \epsilon)$ -approximation algorithm for the densest  $k$ -subgraph.

*Proof.* Let a constant  $\epsilon > 0$  and a  $(1 + \epsilon)$ -approximation algorithm  $M_B$  for the  $l$ -EIS problem. For an instance of densest  $k$ -subgraph, which is a graph  $G = (V, E)$  and a parameter  $k$ , we execute the algorithm  $M_A$  as appears in the pseudocode, and we call as a subroutine the algorithm  $M_B$ .

---

```

// result stores the selected edges
result =  $\emptyset$ 

for ( $l = 0$  |  $l \leq |E|$  |  $l += 1$ ) {
  run  $M_B$  with graph  $G$  and parameter  $l$ 
  let  $S$  be the induced vertices and
   $G[S]$  be the induced subgraph

  while ( $|S| > k$ ) {
    pick vertex  $u$  with the minimum degree
    with respect to  $G[S]$ 

    update  $S = S \setminus u$ 
  }

  let  $E'$  be the edges of the final  $G[S]$ 

  if ( $|result| < |E'|$ ) {
    result =  $E'$ 
  }
}

return result

```

---

Let  $l^*$  be the optimal solution cost for the input instance  $(G, k)$  of the densest  $k$ -subgraph. Then for  $l = l^*$ , the for-loop of the pseudocode will call  $M_B$  and it will return a subset of vertices  $S$ . If  $k^*$  is the optimal solution cost of the instance  $(G, l^*)$  for the  $l$ -EIS, then as  $M_B$  is an  $(1 + \epsilon)$ -approximation algorithm, the size of  $S$  will be bounded as:

$$|S| \leq (1 + \epsilon) \cdot k^* \leq (1 + \epsilon) \cdot k$$

Consequently, the while-loop of the pseudocode will erase at most  $\epsilon \cdot k$  vertices, in order to produce in the end a subset of  $k$  vertices. Therefore, we should prove that the final number of edges after the deletion of few vertices, is very close to the optimal  $l^*$ .

To proceed with this idea, let us initially mention some important observations that are useful for the analysis. Firstly, after the execution of  $M_B$  with parameter  $l$ , the

produced solution can contain exactly  $l$  edges, as it is not helpful to select more than  $l$ . Secondly, for a vertex  $u$  with the minimum degree in a graph  $G = (V, E)$ , it is true that:

$$\deg(u) \leq \sum_{v \in V} \deg(v) \cdot \frac{1}{|V|} = \frac{2 \cdot |E|}{|V|}$$

Hence the degree of all the erased vertices will be smaller than the average degree of the current induced subgraph, and each induced subgraph will contain at most  $l^*$  edges and at least  $k$  vertices. Thus the degree of each deleted vertex will be bounded by the value  $\frac{2 \cdot l^*}{k}$ . Therefore when  $M_A$  will run with  $l = l^*$ , the number of the erased edges will be bounded as:

$$\begin{aligned} l^* - |E'| &\leq \frac{2 \cdot l^*}{k} \cdot \epsilon \cdot k = 2 \cdot l^* \cdot \epsilon \Rightarrow \\ |E'| &\geq l^* \cdot (1 - 2 \cdot \epsilon) \end{aligned}$$

The size  $|result|$  of the final edges will be the maximum over all the iterations, and so the cost of the produced solution of the algorithm will be at least  $(1 - 2 \cdot \epsilon)$  times the cost of the optimal solution. Finally as  $M_A$  runs in polynomial time in the input size, it is an  $(1 - 2 \cdot \epsilon)$ -approximation algorithm for the maximization problem densest  $k$ -subgraph.  $\square$

From the previous theorem and the discussion above we can conclude that PCPK does not admit a PTAS unless  $\text{NP} \subseteq \cap_{\epsilon > 0} \text{BPTIME}(2^{n^\epsilon})$  and so by extending the minimum knapsack problem to the precedence constrained knapsack problem, under the previous assumption we end up with a harder problem.

The intuition that PCKP is harder than the minimum knapsack, comes from Theorem 5 of [12] with the reduction that they give from an instance of  $l$ -EIS to an instance of PCKP. In particular, PCKP has the advantage that through the partial order we can manipulate elements that represent two different entities, in our case vertices and edges. Thus we can force our program to pick the induced vertices for every selected edge, using a constraint of type  $u \preceq e \Leftrightarrow u \in e$ .





# CHAPTER 4

## CAPACITATED COVERING INTEGER PROGRAMS

### 4.1 Introduction

This chapter is devoted to the *capacitated covering integer programs (CIP)* and especially to the case that we are allowed to pick each element at most once. Let us recall that a CIP has the following form:

$$\begin{aligned} & \text{minimize } c^T x \\ & \text{subject to } Ax \geq D \\ & \quad x \leq d \\ & \quad x \in \mathbb{Z}_+^n \end{aligned}$$

By  $f$  we will denote the maximum number of non-zero coefficients in a row of  $A$  and by  $\alpha$  the *dilation*, which is the maximum number of constraints that any variable occurs in. By 0-1 CIP we will denote the special case of the CIP, where the upper bounds  $d_j$  are equal to one. Let  $N$  be the set of elements,  $M$  be the set of constraints and  $n, m$  their sizes respectively. Then a 0-1 CIP instance can be written as follows:

$$\begin{aligned} & \text{minimize } \sum_{j \in N} c_j \cdot x_j \\ & \text{subject to } \sum_{j \in N} u_{ij} \cdot x_j \geq D_i, \forall i \in M \\ & \quad x_j \in \{0, 1\}, \forall j \in N \end{aligned}$$

Many interesting optimization problems can be formulated as a CIP. Therefore the results and the techniques that have been developed for the CIP, can also be used for these special cases. It is worth mentioning though, that this is not trivial for the PCKP problem as the natural integer program that formulates it, contains constraints of the form  $x_i - x_j \geq 0$  with negative coefficients, and this not allowed to the CIP.

Furthermore CIP inherits all the lower bounds of these special cases and as a result CIP is not only NP-complete, but also there is no  $(f - 1 - \epsilon)$ -approximation algorithm for any fixed  $\epsilon > 0$  [5] and no  $o(\ln m)$ -approximation algorithm [17], unless P = NP.

These lower bounds arise from the hardness of the Set Cover problem which is a special case of 0-1 CIP, where also the values  $u_{ij}$  are either zero or one.

In the matter of approximation factors based on rows of  $A$ , for the 0-1 CIP case, Carr et al. [3] by adding the knapsack cover inequalities for each constraint, developed an  $f$ -approximation rounding algorithm and Fujito [6] an  $f$ -approximation primal-dual algorithm, using also the knapsack cover inequalities. Also for the 0-1 CIP case, Takazawa et al. at [16] and [15], slightly improve the approximation factor by presenting an  $(f - \frac{f-1}{m})$ -approximation primal-dual algorithm and an  $f_2$ -approximation primal-dual algorithm respectively, where  $f_2$  is the second largest number of non-zero coefficients in a row of  $A$ . In the case of the general multiplicity constraints ( $d > 1$ ), we can achieve an  $(f + 1)$ -approximation rounding algorithm, using the bucketing algorithm from [3]. According to [13], both papers [3, 6] claim an  $f$ -approximation algorithm for the general CIP with  $d > 1$  without a proof, and there is not any straightforward method of extending their techniques to the general case. Hence at [13], they introduce the so-called  $\rho$ -roundable constraints, with the help of which they present an  $f$ -approximation rounding algorithm for the general CIP. For approximation factors based on columns of  $A$ , Kolliopoulos and Young [11] also using the knapsack cover inequalities, developed an  $O(\ln \alpha)$ -approximation algorithm for the general CIP case, which is best possible unless  $P = NP$ .

## 4.2 Results for the 0-1 CIP

In the rest of this chapter, motivated by the ideas from Takazawa et al. at [16], we will present related results. The improved primal-dual  $(f - \frac{f-1}{m})$ -approximation algorithm at [16] for the 0-1 CIP, solves  $O(n^2)$  subproblems of 0-1 CIP instances with the primal-dual algorithm of Fujito [6] (PD). Following a very similar logic with them in the analysis of the cost, we prove in the next theorem that the primal-dual algorithm by Fujito is actually an  $(f - \frac{f-2}{m})$ -approximation algorithm. Hence the approximation factor of the algorithm is slightly better than  $f$ , without the need of the extra  $O(n^2)$  factor in the time complexity.

To be able to present this result, let us first introduce some useful notations and the relaxation that Fujito used for his primal-dual algorithm. Following the notations that we used before for a 0-1 CIP formulation, let  $A \subseteq N$  be a subset of elements. Then by  $M(A) = \{i \in M \mid D_i(A) > 0\}$ , we denote the set of constraints that are not yet satisfied after selecting all the elements of  $A$ . Also for every  $j \in N \setminus A$ , we denote by  $U_j(A) = \sum_{i \in M(A)} \frac{u_{ij}(A)}{D_i(A)}$ , the sum of the effective values of the element  $j$  from all the active constraints, divided by the residual demand of each constraint. The division is for simplification reasons, as we would prefer to have the value 1 at the right hand side of the inequality constraints.

The linear relaxation of Fujito which is a valid relaxation for the 0-1 CIP can be written as follows:

$$\begin{aligned} & \text{minimize} && \sum_{j \in N} c_j \cdot x_j \\ & \text{subject to} && \sum_{j \in N \setminus A} U_j(A) \cdot x_j \geq |M(A)|, \forall A \subseteq N \\ & && x_j \geq 0, \forall j \in N \end{aligned}$$

and the dual linear program of it, is the following one:

$$\begin{aligned} & \text{maximize} && \sum_{A \subseteq N} |M(A)| \cdot y(A) \\ & \text{subject to} && \sum_{A \subseteq N: j \notin A} U_j(A) \cdot y(A) \leq c_j, \quad \forall j \in N \\ & && y(A) \geq 0, \quad \forall A \subseteq N \end{aligned}$$

Therefore now, we can proceed with the next theorem for the approximation factor of PD.

**Theorem 4.1.** Let an instance of 0-1 CIP with  $f \geq 2$ . The cost of the produced solution  $x$  by PD is at most  $a = f - \frac{f-2}{m}$  times the cost of the optimal integer solution.

*Proof.* Let  $S = \{j \in N \mid x_j = 1\}$  be the support vector of  $x$ , and  $y$  be the dual solution which is produced by PD. For the cost of  $x$  it holds that:

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

To prove that  $\text{cost}(x) \leq a \cdot \text{OPT}$ , it suffices to show that for any  $A \subseteq N$  such that  $y(A) > 0$ , it holds that:

$$\sum_{j \in S \setminus A} U_j(A) \leq a \cdot |M(A)|$$

Because then we would conclude that:

$$\text{cost}(x) = \sum_{A \subseteq N} y(A) \sum_{j \in S \setminus A} U_j(A) \leq \sum_{A \subseteq N} y(A) \cdot |M(A)| \cdot a \leq a \cdot \text{OPT}$$

By the definition of  $U_j(A)$  it follows that:

$$\sum_{j \in S \setminus A} U_j(A) = \sum_{j \in S \setminus A} \sum_{i \in M(A)} \frac{u_{ij}(A)}{D_i(A)} = \sum_{i \in M(A)} \sum_{j \in S \setminus A} \frac{u_{ij}(A)}{D_i(A)}$$

Let  $l$  be the last element that the algorithm picks. Then because of the way PD builds the dual solution,  $y(A) > 0$  implies that  $A \subseteq S \setminus \{l\}$  and from the definition of  $M(A)$ , for these  $A$  we know that:

$$M(S \setminus \{l\}) \subseteq M(A)$$

Therefore for such a fixed  $A$ , we can split  $M(A)$  in two parts  $M(S \setminus \{l\})$  and  $M(A) \setminus M(S \setminus \{l\})$ , and as a consequence:

$$\sum_{j \in S \setminus A} U_j(A) = \sum_{i \in M(S \setminus \{l\})} \sum_{j \in S \setminus A} \frac{u_{ij}(A)}{D_i(A)} + \sum_{i \in M(A) \setminus M(S \setminus \{l\})} \sum_{j \in S \setminus A} \frac{u_{ij}(A)}{D_i(A)}$$

For the second part of the sum, because  $u_{ij}(A) \leq D_i(A)$ , it is true that for any  $i \in M(A) \setminus M(S \setminus \{l\})$ :

$$\sum_{j \in S \setminus A} u_{ij}(A) \leq f \cdot D_i(A)$$

and as  $D_i(A) > 0$ , it holds that:

$$\sum_{j \in S \setminus A} \frac{u_{ij}(A)}{D_i(A)} \leq f \quad (4.1)$$

For the first part of the sum, we can notice that taking all the elements except the last one is not enough to satisfy the constraints which is part of  $M(S \setminus \{l\})$ . Hence, it must be true that for any  $i \in M(S \setminus \{l\})$ :

$$\begin{aligned} \sum_{j \in S \setminus A} u_{ij}(A) &= \sum_{j \in (S \setminus \{l\}) \setminus A} u_{ij}(A) + u_{il}(A) \\ &\leq \sum_{j \in (S \setminus \{l\}) \setminus A} u_{ij} + u_{il}(A) \\ &= \sum_{j \in S \setminus \{l\}} u_{ij} - \sum_{j \in A} u_{ij} + u_{il}(A) \\ &< D_i - \sum_{j \in A} u_{ij} + D_i(A) \\ &= D_i(A) + D_i(A) = 2 \cdot D_i(A) \end{aligned}$$

and as  $D_i(A) > 0$ , it holds that:

$$\sum_{j \in S \setminus A} \frac{u_{ij}(A)}{D_i(A)} \leq 2 \quad (4.2)$$

From (4.1) and (4.2) we obtain that:

$$\begin{aligned} \sum_{j \in S \setminus A} U_j(A) &\leq \sum_{i \in M(S \setminus \{l\})} 2 + \sum_{i \in M(A) \setminus M(S \setminus \{l\})} f \\ &= 2 \cdot |M(S \setminus \{l\})| + f \cdot (|M(A)| - |M(S \setminus \{l\})|) \\ &= |M(S \setminus \{l\})| \cdot (2 - f) + f \cdot |M(A)| \end{aligned}$$

Since the set  $S \setminus \{l\}$  is an infeasible solution, it is true that  $|M(S \setminus \{l\})| \geq 1$ . Also for any  $A$  such that  $y(A) > 0$ , it is true that  $1 \leq |M(A)| \leq m$ . Therefore for  $f \geq 2$ , we finally obtain that:

$$\begin{aligned} \sum_{j \in S \setminus A} U_j(A) &\leq 2 - f + f \cdot |M(A)| \\ &= \left(f - \frac{f-2}{|M(A)|}\right) \cdot |M(A)| \\ &\leq \left(f - \frac{f-2}{m}\right) \cdot |M(A)| \\ &= a \cdot |M(A)| \end{aligned}$$

And as a result we can conclude that PD is an  $(f - \frac{f-2}{m})$ -approximation algorithm.  $\square$

Someone could argue that we can apply a more complicated analysis in the PD algorithm and get a better approximation guarantee. In the next lemma though, we show that it is possible for the PD to return an integer solution with cost of exactly  $f - \frac{f-1}{m}$

times the cost of the dual solution  $y$  that is produced by the algorithm. Thus with the standard analysis, that we compare the integer solution cost with the dual solution cost, we should not expect to get better approximation factor than  $f - \frac{f-1}{m}$ , with this formulation and the PD algorithm.

**Lemma 4.2.** An integer solution found by PD can have cost of exactly  $f - \frac{f-1}{m}$  times the cost of the dual solution  $y$  which is produced by the algorithm.

*Proof.* Let the following infinite family of instances, for which all the constraints are similar except one.

$$\begin{aligned} & \text{minimize } \sum_{j \in N} x_j \\ & \text{subject to } \sum_{j \in N} n \cdot x_j \geq n, \quad \forall i \in (M \setminus \{m\}) \\ & \quad \sum_{j \in N} x_j \geq n \\ & \quad x_j \in \{0, 1\}, \quad \forall j \in N \end{aligned}$$

The corresponding dual of the linear relaxation of Fujito is the following one:

$$\begin{aligned} & \text{maximize } \sum_{A \subseteq N} |M(A)| \cdot y(A) \\ & \text{subject to } \sum_{A \subseteq N: j \notin A} U_j(A) \cdot y(A) \leq 1, \quad \forall j \in N \\ & \quad y(A) \geq 0, \quad \forall A \subseteq N \end{aligned}$$

The algorithm starts by increasing  $y(\emptyset)$ . For every  $j \in N$  it holds that  $U_j(\emptyset) = m - 1 + \frac{1}{n}$  and hence the value of  $y(\emptyset)$  will be set equal to  $\frac{1}{m-1+\frac{1}{n}}$ . All the dual constraints will become tight. Even though the algorithm will not terminate until it gathers all the elements,  $y(\emptyset)$  will be the only dual variable which will have a non-zero value. Therefore the cost of the dual solution  $y$  which will be produced, will be equal to:

$$\sum_{A \subseteq N} |M(A)| \cdot y(A) = m \cdot y(\emptyset) = \frac{m}{m-1+\frac{1}{n}}$$

The cost of the integer solution which is also the optimal one will be equal to  $n$  and thus the ratio will be equal to:

$$\frac{n}{\frac{m}{m-1+\frac{1}{n}}} = \frac{n \cdot (m-1+\frac{1}{n})}{m} = \frac{n \cdot m - n + 1}{m} = n - \frac{n}{m} + \frac{1}{m}$$

Observing that  $f = n$ , we can conclude that the difference between the cost of the integer solution and the dual solution which are produced by the PD algorithm is exactly equal to:

$$f - \frac{f}{m} + \frac{1}{m} = f - \frac{f-1}{m}$$

□

The integer program of Fujito [6] is a relaxation of the integer program of Carr et al. not only in the sense that it does not cut off the integer feasible solutions, but also it does not eliminate any feasible solution at all. That is, the convex polytope of the feasible set of the linear program of Carr et al. is a subset of the convex polytope of the feasible set of the linear program of Fujito. Therefore, from Theorem 4.1 we can conclude that the integrality gap of the formulation of Carr et al. is at most  $f - \frac{f-2}{m}$ . Moreover, as we have already mentioned, Carr et al. [3] developed an  $f$ -approximation rounding algorithm for the 0-1 CIP. In the next theorem though, we apply the same ideas we used for the algorithm of Fujito, but this time we develop an  $(f - \frac{f-2}{m})$ -approximation primal-dual algorithm using the formulation of Carr et al.

**Theorem 4.3.** Let  $a = f - \frac{f-2}{m}$  and an instance of 0-1 CIP. Then there exists an  $a$ -approximation primal-dual algorithm using the formulation of Carr et al.

*Proof.* For an instance of 0-1 CIP, the formulation of Carr et al. is the following one:

$$\begin{aligned} & \text{minimize } \sum_{j \in N} c_j \cdot x_j \\ & \text{subject to } \sum_{j \in N \setminus A} \frac{u_{ij}(A)}{D_i(A)} \cdot x_j \geq 1, \quad \forall A \subseteq N \quad \forall i \in M(A) \\ & \quad \quad \quad x_j \in \{0, 1\}, \quad \forall j \in N \end{aligned}$$

The dual of the linear relaxation of the above integer program is:

$$\begin{aligned} & \text{maximize } \sum_{A \subseteq N} \sum_{i \in M(A)} y_i(A) \\ & \text{subject to } \sum_{A \subseteq N: j \notin A} \sum_{i \in M(A)} \frac{u_{ij}(A)}{D_i(A)} \cdot y_i(A) \leq c_j, \quad \forall j \in N \\ & \quad \quad \quad y_i(A) \geq 0, \quad \forall A \subseteq N \quad \forall i \in M(A) \end{aligned}$$

The algorithm will produce a  $(x, y)$  pair solution, where  $x$  is a primal integer solution and  $y$  is a dual linear solution such that the cost of  $x$  is at most  $a$  times the cost of  $y$ . The algorithm will follow the usual procedure of primal-dual algorithms, starting with  $S = \emptyset$  and repeatedly increasing the solution set  $S$ . While the algorithm is still running and the set  $S$  is not yet a solution, we increase uniformly all the  $y_i(S)$  for all  $i \in M(S)$ , until a dual constraint becomes tight. Therefore for a specific  $A \subseteq N$ , it will hold that for all  $i \in M(S)$ , all the  $y_i(A)$  will be equal, and this value will be denoted as  $y(A)$ . Let  $S = \{j \in N \mid x_j = 1\}$  be the support vector. For the cost of  $x$  it holds that:

$$\begin{aligned} \text{cost}(x) &= \sum_{j \in N} c_j \cdot x_j = \sum_{j \in S} c_j = \sum_{j \in S} \sum_{A \subseteq N: j \notin A} \sum_{i \in M(A)} \frac{u_{ij}(A)}{D_i(A)} \cdot y_i(A) \\ &= \sum_{A \subseteq N} \sum_{i \in M(A)} y_i(A) \sum_{j \in S \setminus A} \frac{u_{ij}(A)}{D_i(A)} = \sum_{A \subseteq N} \sum_{i \in M(A)} y(A) \sum_{j \in S \setminus A} \frac{u_{ij}(A)}{D_i(A)} \\ &= \sum_{A \subseteq N} y(A) \sum_{i \in M(A)} \sum_{j \in S \setminus A} \frac{u_{ij}(A)}{D_i(A)} \end{aligned}$$

From Theorem 4.1, we know that:

$$\sum_{j \in S \setminus A} U_j(A) = \sum_{j \in S \setminus A} \sum_{i \in M(A)} \frac{u_{ij}(A)}{D_i(A)} = \sum_{i \in M(A)} \sum_{j \in S \setminus A} \frac{u_{ij}(A)}{D_i(A)}$$

and this value is bounded by:

$$\sum_{j \in S \setminus A} U_j(A) \leq a \cdot |M(A)|$$

Therefore we can bound the cost of  $x$  by:

$$\text{cost}(x) \leq \sum_{A \subseteq N} y(A) \cdot |M(A)| \cdot a \leq a \cdot \text{cost}(y)$$

The last inequality holds because for each  $i \in M(A)$ , we increase uniformly the corresponding  $y_i(A)$ , and so:

$$\text{cost}(y) = \sum_{A \subseteq N} \sum_{i \in M(A)} y_i(A) = \sum_{A \subseteq N} \sum_{i \in M(A)} y(A) = y(A) \cdot |M(A)|$$

□

Consequently this result implies that the formulation of Fujito is not necessary in order to achieve the previous results for the 0-1 CIP.





## CHAPTER 5

### CONCLUSION AND OPEN PROBLEMS

In this thesis, our main goal was to investigate the PCKP problem together with the formulation that McCormick et al. [12] suggested. In the third chapter, initially we investigated the integrality gap of the formulation and we proved with Lemma 3.2, that it is  $\Omega(w(\mathcal{P}))$ . Hence, in order to improve the guarantee for the PCKP problem, we should use some new ideas. A different formulation was also studied using the simpler pitch-1 inequalities and in Theorem 3.5 we proved that in the worst case the two formulations are equivalently strong. Also a new rounding  $w(\mathcal{P})$ -approximation algorithm for the 0-1 PCKP problem was developed. In Section 3.5, we studied the PCKP problem with general multiplicity constraints and with Theorem 3.14 we prove that there exists an algorithm with strongly polynomial bounds, something which was an open question at [12]. In the end of the third chapter, a more detailed proof of the already known inapproximability result for the PCKP, is also given. Finally in the fourth chapter, we studied the CIP problem and specifically the 0-1 case and we presented some results related to the Fujito and Carr et al. linear programming relaxations.

In the second chapter we described the minimum knapsack problem for the case that the elements can be selected at most once and we presented results from the literature [3], [2]. Interestingly, the minimum knapsack problem can be easily extended to the general case with multiplicity constraints [3], [12], and the approximation algorithms maintain the same approximation factor, which is in our case equal to 2. However this generalization from the 0-1 case to the general multiplicity constraints is not a trivial procedure for the PCKP problem. This problem was studied by McCormick et al. [12] and they state that it is an open problem to provide an algorithm with strongly polynomial bounds. In the third chapter, we described a way to achieve strongly polynomial bounds, but the question remains whether we can achieve something better, for example to replace  $w^2(\mathcal{P})$  with  $w(\mathcal{P})$  in Theorem 3.14. Furthermore the gap between the lower and upper bounds for the PCKP problem is large, even for the 0-1 case, and so it would be interesting either to provide a stronger lower bound than the one that says that there is no PTAS or to develop an algorithm with approximation factor better than  $w(\mathcal{P})$ .

---

## BIBLIOGRAPHY

- [1] Daniel Bienstock, Yuri Faenza, Igor Malinović, Monaldo Mastrolilli, Ola Svensson, and Mark Zuckerberg. "On inequalities with bounded coefficients and pitch for the min knapsack polytope". In: *Discrete Optimization* (2020), p. 100567.
- [2] Tim Carnes and David Shmoys. "Primal-dual schema for capacitated covering problems". In: *International Conference on Integer Programming and Combinatorial Optimization*. Springer. 2008, pp. 288–302.
- [3] 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*. 2000, pp. 106–115.
- [4] Robert P. Dilworth. "A decomposition theorem for partially ordered sets". In: *Classic Papers in Combinatorics*. Springer, 2009, pp. 139–144.
- [5] Irit Dinur, Venkatesan Guruswami, Subhash Khot, and Oded Regev. "A new multilayered PCP and the hardness of hypergraph vertex cover". In: *SIAM Journal on Computing* 34.5 (2005), pp. 1129–1146.
- [6] Toshihiro Fujito. "On combinatorial approximation of covering 0-1 integer programs and partial set cover". In: *Journal of combinatorial optimization* 8.4 (2004), pp. 439–452.
- [7] Michael R. Garey and David S. Johnson. *Computers and intractability*. W. H. Freeman and Company, 1979.
- [8] David S. Johnson and K. A. Niemi. "On knapsacks, partitions, and a new dynamic programming technique for trees". In: *Mathematics of Operations Research* 8.1 (1983), pp. 1–14.
- [9] Richard M. Karp. "Reducibility among combinatorial problems". In: *Complexity of computer computations*. Springer, 1972, pp. 85–103.
- [10] Subhash Khot. "Ruling out PTAS for graph min-bisection, dense k-subgraph, and bipartite clique". In: *SIAM Journal on Computing* 36.4 (2006), pp. 1025–1071.
- [11] Stavros G. Kolliopoulos and Neal E. Young. "Approximation algorithms for covering/packing integer programs". In: *Journal of Computer and System Sciences* 71.4 (2005), pp. 495–505.
- [12] S. Thomas McCormick, Britta Peis, José Verschae, and Andreas Wierz. "Primal-dual algorithms for precedence constrained covering problems". In: *Algorithmica* 78.3 (2017), pp. 771–787.

- [13] David Pritchard and Deeparnab Chakrabarty. "Approximability of sparse integer programs". In: *Algorithmica* 61.1 (2011), pp. 75–93.
- [14] Natthawut Samphaiboon and Y. Yamada. "Heuristic and exact algorithms for the precedence-constrained knapsack problem". In: *Journal of optimization theory and applications* 105.3 (2000), pp. 659–676.
- [15] Yotaro Takazawa and Shinji Mizuno. "A 2-approximation algorithm for the minimum knapsack problem with a forcing graph". In: *Journal of the Operations Research Society of Japan* 60.1 (2017), pp. 15–23.
- [16] Yotaro Takazawa, Shinji Mizuno, and Tomonari Kitahara. *An improved approximation algorithm for the covering 0-1 integer program*. Working Paper 2017-6. Department of Industrial Engineering and Economics, 2017.
- [17] Luca Trevisan. "Non-approximability results for optimization problems on bounded degree instances". In: *Proceedings of the thirty-third annual ACM symposium on Theory of computing*. 2001, pp. 453–461.
- [18] Vijay V. Vazirani. *Approximation algorithms*. Springer Science & Business Media, 2013.
- [19] David P. Williamson and David B. Shmoys. *The design of approximation algorithms*. Cambridge University Press, 2011.