Bin Packing Games

Master thesis

Xian Qiu
xian-qiu@hotmail.com

Discrete Mathematics and Mathematical Programming Chiar
Department of Applied Mathematics
University of Twente

January 2010

Responsible	**Supervisor**
Prof. Marc Uetz	Walter Kern

Abstract

The bin packing game is a cooperative N-person game, where the set of players consists of k bins, each has capacity 1 and n items of sizes $a_1, a_2, \cdots, a_n$, w.l.o.g, we assume $0 \leq a_i \leq 1$ for all $1 \leq i \leq n$. The value function of a coalition of bins and items is the maximum total size of items in the coalition that can be packed into the bins of the coalition. A typical question of the bin packing game is to study the existence of the core, i.e. given an instance of a bin packing game v, is the core $C(v) \neq \emptyset$? If the answer is 'yes', then how to find the core allocation of the grand coalition?

Instead of directly analyzing the existence of the core, we study by look at the ϵ-core, which can be viewed as the generalization of the core because it is the core when $\epsilon = 0$. For any instance of the bin packing game, there exists a minimal ϵ_{min} such that for all $\epsilon \geq \epsilon_{min}$, the ϵ-core is not empty. The ϵ is also called the tax rate, hence the problem becomes to find the minimal tax rate such that the associated ϵ-core is nonempty.

In chapter 1, we briefly introduce the background of game theory and some concepts from the cooperative game theory. In chapter 2, by studying the fractional bin packing game, we give a sufficient and necessary condition for the existence of the ϵ-core and successively summarize some results about the bound of the minimal tax rate. In chapter 3, we study the computational complexity of bin packing games and fractional bin packing games. In chapter 4 and chapter 5, we discuss exact algorithms and approximation algorithms for computing the value function of bin packing games and the corresponding fractional bin packing games, as well as the approximation algorithm for computing the minimal tax rate. Finally, in chapter 6, we summarize the conclusions of previous chapters and further discuss the related unsolved problems we have met. In the end, we present some simple applications of the bin packing game, which are useful in practice.

Notation

$\mathbb{R}^n$	n-dimensional real number space
N	player set or the grand coalition
$\|N\|$	number of players in player set N
2^N	collection of all subsets of N
$S \subset N$	coalition
G^N	set of characteristic functions
$v : 2^N \backslash \{\emptyset\} \to \mathbb{R}$	characteristic function (value function)
$\langle N, v \rangle$	the game with player set N and value function v
$x = (x_1, x_2, \cdots, x_{\|N\|})^T$	payoff vector of the grand coalition N
$C(v)$	core of the game v
$C_\epsilon(v)$	ϵ-core of the game v
$I = \{1, 2, \cdots, n\}$	item set
$I_B = \{n+1, n+2, \cdots, n+k\}$	bin set
$a = (a_1, a_2, \cdots, a_n)^T$	size vector, where a_i is the size of item i
v_{INT}	value function of the bin packing game
v_{FRA}	value function of the fractional bin packing game
$\mathcal{F}$	collection of all feasible sets
f_j	feasible set
$\sigma(f_j)$	value of the feasible set f_j
$\sigma = (\sigma(f_j)) \in \mathbb{R}^{\mathcal{F}}$	total size vector
b_j	feasible vector
$B = (b_1\ b_2\ \cdots\ b_{\|\mathcal{F}\|})$	feasible matrix

ϵ_{min}	the minimal tax rate
I	an instance of the optimization problem
$\|I\|$	encoding length of the instance I
$A(I)$	output of the algorithm A when applying to the instance I
$OPT(I)$	optimal value of the instance I

Contents

Chapter 1

Introduction

1.1 Report Outline

Our study on the bin packing game is focused on the problem of finding the core (ϵ-core) allocation of the grand coalition, which in this thesis, principally consists of four parts: study the existence of the core (ϵ-core), the complexity results about computations of the core (ϵ-core), the exact algorithms as well as the approximation algorithms of computing the core (ϵ-core).

Since the bin packing game is a cooperative game, in this chapter we briefly introduce the history of the game theory and some basic definitions from the cooperative game theory, after this we define the bin packing game and further show an example of it.

In chapter 2, we study the emptiness of the ϵ-core. For any instance of the bin packing game, we first define the corresponding fractional bin packing game, then using this definition we give a sufficient and necessary condition for the existence of the ϵ-core. Based on this condition, we successively derive some results about bound of the minimal tax rate.

In chapter 3, we study the computational complexity of bin packing games and fractional bin packing games. Moreover, the problem of testing whether an allocation vector lies in the ϵ-core and the problem of deciding whether an instance of the bin packing game has a nonempty ϵ-core also draw our attention in this chapter.

By introducing the matrix form of the linear program for bin packing games and the corresponding fractional bin packing games, we present an algorithm for computing the corresponding value function in chapter 4. In particular for large instance of the fractional bin packing game, we apply the column generation approach, and in the sequel give an example to account for how this approach works.

In chapter 5, instead of researching on exact algorithms, we investigate approximation algorithms for computing value functions of the bin packing game and the

fractional bin packing game. After that, we further propose an approximation algorithm for calculating the minimal tax rate.

Finally, in chapter 6, we summarize the conclusions of previous chapters and discuss the related unsolved problems we have met. In the end, we present some simple applications of the bin packing game, which are useful in practice.

1.2 Games and Applications

Game theory is a branch of applied mathematics that is used in the social sciences, most notably in economics, as well as in biology, engineering, political science, international relations, computer science, and philosophy. Game theory attempts to mathematically capture behavior in strategic situations, in which an individual's success in making choices depends on the choices of others. While initially developed to analyze competitions in which one individual does better at another's expense (zero sum games), it has been expanded to treat a wide class of interactions, which are classified according to several criteria.

Traditional applications of game theory attempt to find equilibria in these games. In an equilibrium, each player of the game has adopted a strategy that they are unlikely to change. Many equilibrium concepts have been developed (most famously the Nash equilibrium) in an attempt to capture this idea. These equilibrium concepts are motivated differently depending on the field of application, although they often overlap or coincide. In this section, we give a short introduction about the background of the game theory and some basic definitions from the cooperative game theory.

1.2.1 History

The earliest example of a formal game-theoretic analysis is the study of a duopoly by Antoine Cournot in 1838. The mathematician Émile Borel suggested a formal theory of games in 1921, which was furthered by the mathematician John von Neumann in 1928 in a paper *theory of parlor games*. Game theory was established as a field in its own right after the 1944 publication of the monumental volume *Theory of Games and Economic Behavior* by von Neumann and the economist Oskar Morgenstern. This book provided much of the basic terminology and problem setup that is still in use today.

In 1950, John Nash introduced the concept of a *nash equilibrium*, which became a focal point of analysis in *noncooperative game theory* since then. Immediately after this, game theory was broadened theoretically and applied to problems of war and politics. In 1951, Nash followed this up with the concept of a *nash bargaining*

solution for *cooperative games*. Lloyd Shapley (1953) introduced the concept of a *shapley value* and the *core* as solutions to cooperative games. Throughout the early 1960s, Robert J. Aumann and Martin Shubik began to apply cooperative game theory extensively throughout economics, and, in the process, went on to invent several solution concepts for cooperative games (e.g. bargaining set, strong equilibrium). Since the 1970s, game theory has driven a revolution in economic theory. Additionally, it has found applications in sociology and psychology, and established links with evolution and biology. Game theory received special attention in 1994 with the awarding of the Nobel prize in economics to Nash, John Harsanyi, and Reinhard Selten.

At the end of the 1990s, a high-profile application of game theory has been the design of auctions. Prominent game theorists have been involved in the design of auctions for allocating rights to the use of bands of the electromagnetic spectrum to the mobile telecommunications industry. Most of these auctions were designed with the goal of allocating these resources more efficiently than traditional governmental practices, and additionally raised billions of dollars in the United States and Europe.

1.2.2 What is the Game?

The object of study in game theory is the *game*, which is characterized by a number of *players* or *decision makers* who interact, possibly threaten each other and form coalitions, take actions under uncertain conditions, and finally receive some benefit or reward or possibly some punishment or monetary loss.

The commonly known games, for instance, the entertaining games, such as chess, poker, tic-tac-toe, bridge, baseball, computer games etc. In some cases, we want to understand what is happening in order to make better predictions about the future and furthermore to be able to suggest what courses of an action should be taken by the players. Fortunately, various mathematical models of games have already been established, although may not be perfect.

Game theory can be roughly divided into two broad areas: non-cooperative (or strategic) games and cooperative (or coalitional) games . The meaning of these terms are self evident. Players in the non-cooperative games work independently, so for each player, he is only interested in making the best decision to maximize his proceeds. The non-cooperative game theory is concerned with the analysis of strategic choices, namely, the details of players' choices are crucial to determine the outcome of a game; while the cooperative game theory investigates coalitional games with respect to the formation of coalitions of players, and a fair allocation of the payoff to each player. This is most naturally applied to situations arising in political science or international relations, where concepts like coalitions are most important.

In following sections, we shall present the mathematical model of cooperative games.

1.2.3 Cooperative Games

A *cooperative game* is concerned primarily with groups of players-who coordinate their actions and pool their winnings. Consequently, one of the problems here is how to fairly divide the extra earnings among the members of the formed groups, so that every player is still willing to cooperate. Let N be a non-empty finite set of players, $S \subset N$ is referred to as a *coalition*, which represents the group of players. The set N is called the *grand coalition* and $\emptyset$ is called the *empty coalition*. We denote the collection of coalitions, i.e. the set of all subsets of N by 2^N. Commonly the player set $N = \{1, 2, \cdots, n\}$, and for each $S \in 2^N$ we denote by $|S|$ the number of elements of S, and by e^S the characteristic vector of S with i-th component $(e^S)^i = 1$ if $i \in S$, and $(e^S)^i = 0$ if $i \in N \backslash S$.

Definition 1.2.1. *A **cooperative game** in characteristic function form is an ordered pair $\langle N, v\rangle$ consisting of the player set N and the characteristic function $v : 2^N \to \mathbb{R}$ with $v(\emptyset) = 0$.*

The characteristic function v is also often called the *value function*. Given a coalition S, the real number $v(S)$ can be interpreted as the maximal worth or cost savings that the members of S can obtain when they cooperate. Often we identify the game $\langle N, v\rangle$ with its characteristic function v.

Example 1.2.2. *Unanimity games $\langle N, u_T\rangle$, $T \in 2^N \backslash \{\emptyset\}$, are defined by*

$$u_T(S) = \begin{cases} 1 & \text{if } T \subset S, \\ 0 & \text{otherwise.} \end{cases}$$

The set G^N of characteristic functions of coalitional games with player set N forms with the usual operations of addition and scalar multiplication of functions a $\left(2^{|N|} - 1\right)$-dimensional linear space; a basis of this space is supplied by the unanimity games u_T, $T \in 2^N \backslash \{\emptyset\}$. One can easily check that for each $v \in G^N$ we have

$$v = \sum_{T \in 2^N \backslash \{\emptyset\}} c_T u_T, \text{with } c_T = \sum_{S : S \subset T} (-1)^{(|T| - |S|)} v(S).$$

The interpretation of the unanimity game u_T is that a gain (or cost savings) of 1 can be obtained if and only if all players in coalition S are involved in cooperation.

Definition 1.2.3. *A game $v \in G^N$ is **additive** if $v(S \cup T) = v(S) + v(T)$ for all S, $T \in 2^N$ with $S \cap T = \emptyset$.*

For an additive game $v \in G^N$, we have $v(S) = \sum_{i \in S} v(i)$ [1] for all $S \in 2^N$, so it forms an n-dimensional linear subspace of G^N. A game $v \in G^N$ is called *inessential* if it is an additive game. For an inessential game there is no problem how to allocate total gain $v(N)$ because $v(S) = \sum_{i \in S} v(i)$, which is to say, in this case coalition S gets no extra profit if players cooperate, compared to working individually.

Most of cooperative games arising from real life situations are superadditive games.

Definition 1.2.4. *A game $v \in G^N$ is superadditive if $v(S \cup T) \geq v(S) + v(T)$ for all S, $T \in 2^N$ with $S \cap T = \emptyset$.*

Of course, in a superadditive game we have $v(\cup_{i=1}^k S_i) \geq \sum_{i=1}^k v(S_i)$ if S_1,S_2,$\cdots$,S_k are pairwise disjoint coalitions. In particular, $v(N) \geq \sum_{i=1}^k v(S_i)$ for each partition $(S_1, S_2, \cdots, S_k)$ of N and $v(N) \geq \sum_{i=1}^n v(i)$. Therefore, in a superadditive game it is advantageous for the players to cooperate. The set of (characteristic function of) superadditive games form a *cone* in G^N, i.e. for all v and w that are superadditive we have that $\alpha v + \beta w$ is also a superadditive game, where $\alpha, \beta \in \mathbb{R}_+$.

Definition 1.2.5. *A map $\lambda : 2^N \backslash \{\emptyset\} \to \mathbb{R}_+$ is called a **balanced map** if*

$$\sum_{S \in 2^N \backslash \{\emptyset\}} \lambda(S) e^S = e^N. \tag{1.1}$$

By the definition we see $\sum_{S \ni i, S \in 2^N \backslash \{\emptyset\}} \lambda(S) = 1$, for all $i \in N$. We can interpret the balanced map λ as follows. For all player $i \in N$, and coalition $S \subset N$, $\lambda(S)$ indicates the energy of player i paid in coalition S. No matter how many coalitions he engaged in, the total energy of the player is equal to 1.

Definition 1.2.6. *A game $v \in G^N$ is **balanced** if for each balanced map $\lambda : 2^N \backslash \{\emptyset\} \to \mathbb{R}_+$ we have*

$$\sum_{S \in 2^N \backslash \{\emptyset\}} \lambda(S) v(S) \leq v(N). \tag{1.2}$$

The above inequality says that the grand coalition gains most among other possible coalitions. Thus, intuitively, we may say the players are very glad to cooperate in the grand coalition N in a balanced game. Further discussions about this will be carried on in the next section.

1.2.4 The Core and Related Concepts

Following up on the preceding description of the cooperative games and characteristic functions, now we take a look at the *payoff vectors* $x = (x_i)_{i \in N} \in \mathbb{R}^n$, with x_i

[1] To simplify notation, we often write $v(1, \cdots, n)$ instead of $v(\{1, \cdots, n\})$.

being the payoff to be given to player $i \in N$, under the condition that cooperation in the grand coalition is reached. Clearly, the actual formation of the grand coalition is based on the agreement of all players upon a proposed payoff in the game. Such an agreement is, or should be, based on all other cooperation possibilities for the players and their corresponding payoffs.

We note first that only payoff vectors $x \in \mathbb{R}^n$ satisfying $\sum_{i\in N} x_i \leq v(N)$ are reachable in the game $v \in G^N$. However, to have any chance of being agreed upon, a payoff vector should be *efficiency*, i.e.

$$\sum_{i\in N} x_i = v(N). \tag{1.3}$$

To motivate the efficiency condition we argue that $\sum_{i\in N} x_i \geq v(N)$ should also hold.

Suppose that $\sum_{i\in N} x_i < v(N)$. In this case we would have

$$a = v(N) - \sum_{i\in N} x_i > 0.$$

Then the players can still form the grand coalition and receive the better payoff $y = (y_1, y_2, \cdots, y_n)$ with $y_i = x_i + a/n$ for all $i \in N$.

Now, note that if there is a player $i \in N$ whose payoff x_i satisfies $x_i < v(i)$, the grand coalition would never form. The reason is that such a player would prefer not to cooperate since acting on his own hen can obtain more. Hence, the *individual rationality* condition

$$x_i \geq v(i) \text{ for all } i \in N \tag{1.4}$$

should hold in order that a payoff vector has a real chance to be realized in the game.

Definition 1.2.7. *A payoff vector $x \in \mathbb{R}^n$ is an **imputation** for the game $v \in G^N$ if it is efficient and individual rational, i.e.*

1. $\sum_{i\in N} x_i = v(N)$;
2. $x_i \geq v(i)$ *for all* $i \in N$.

Nevertheless, imputation can not always grantee every player of N is satisfied. Since there may exists some players $S \subset N$, that their payoff is less than their earnings, which forces these players to work in coalition S, rather than in N. Therefore, to make sure all players of N are satisfied, the payoff vectors should be in the core.

Definition 1.2.8. *The **core** $C(v)$ of a game $v \in G^N$ is the polytope of all vectors $x \in \mathbb{R}^N$ satisfying*

1. $\sum_{i\in N} x_i = v(N)$;
2. $\sum_{i\in S} x_i \geq v(S)$, *for all* $S \subset N$.

If $x \in C(v)$ is the proposed reward allocation in N, then no coalition S has an incentive to split off from N, because the total amount $\sum_{i\in S} x_i$ allocated to S is not smaller than the amount $v(S)$ which the players can obtain by forming the subcoalition.

Reviewing the definition of the balanced game, it tells the players that cooperation in grand coalition N earns maximum profit among other coalitions, which is to say, the core of a balanced game is nonempty. Namely we have the following theorem.

Theorem 1.2.9. *[1] The game $v \in G^N$ is balanced if and only if $C(v) \neq \emptyset$.*

Proof. Consider the linear program (LP)

$$\begin{aligned} &\min \sum_{i\in N} x_i \\ &\text{s.t.} \sum_{i\in S} x_i \geq v(S) \text{ for all } S \in 2^N \backslash \{\emptyset\}. \end{aligned}$$

Note that $C(v) \neq \emptyset$ if and only if the optimal objective value of (LP) equals to $v(N)$. Its dual problem (DP) is

$$\begin{aligned} &\max \sum_{S\in 2^N\backslash\{\emptyset\}} \lambda(S)v(S) \\ &\text{s.t.} \sum_{S\in 2^N\backslash\{\emptyset\}} \lambda(S)e^S = e^N, \\ &\qquad \lambda(S) \geq 0. \end{aligned}$$

The constraints of (DP) implies λ is a balanced map (1.1), so $C(v) \neq \emptyset$ if and only if the optimal objective value of (DP) equals to $v(N)$ and this holds if and only if v is balanced (1.2). □

1.3 Bin Packing Games

A *binpacking game* is defined by a set of items $I = \{1, 2, \cdots, n\}$ of sizes $a_1, a_2, \cdots, a_n$, and k bins, denoted by $I_B = (n+1, n+2, \cdots, n+k)$, each of capacity 1, where we assume, w.l.o.g, $0 \leq a_i \leq 1$. The player set N consists of all bins and all items, i.e. $N = I \cup I_B$, so we have $|N| = n + k$.

For a coalition S containing $k' \leq k$ bins and items $i_1, i_2, \cdots, i_s$, the sizes of the items are $a_{i_1}, a_{i_2}, \cdots, a_{i_s}$. The characteristic function (value function) v relative to S is defined as below

$$v(S) := \max \sum_{j=1}^{k'} \sum_{i \in I_j} a_i, \tag{1.5}$$

where the maximum is taken over all collections of pairwise disjoint subsets $I_1, \cdots, I_{k'} \subseteq \{i_1, \cdots, i_s\}$ such that

$$\sum_{i \in I_j} a_i \leq 1.$$

We set $v(S) = 0$ if $k' = 0$ or S only consists of bins.

In fact, if we assign the items $i_1, i_2, \cdots, i_s$ to the k' bins on condition that the total weight of assigned items in each bin does not exceed the capacity 1, then $v(S)$ is the maximum weight of the assigned items of coalition S . We refer to the assigned items as the *packed items*, while the unassigned items are called *unpacked items*. A *feasible packing* of an item set $I' \subset I$ into bin set $I'_B \subset I_B$ is an assignment of some (or all) elements in I' to the bins in I'_B such that the overall size of items assigned to any bin does not exceed the bin capacity 1. The *value* of a feasible packing is the overall size of all packed items. An *optimal packing* of a coalition S is the feasible packing which has the maximal value over all feasible packing of S.

Example 1.3.1. *Consider a bin packing game of* 2 *bins, and* 4 *items of sizes* $\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2} + \epsilon$*, where* ϵ *is a very small positive real number.*

In this example $N = \{1, 2, 3, 4, 5, 6\}$, besides, if we pack item $1, 2$ into the first bin (player 5) and item 4 into the second bin (player 6), then we get an optimal packing of N, with $v(N) = 3/2 + \epsilon$. Moreover, it is easy to observe that the bin packing game is superadditive since more bins and items the coalition has, the larger value it may get. However, in this example the core of this game does not exist. To show this, we first prove the following lemma.

Lemma 1.3.2. *Let* v *be a bin packing game and suppose* $C(v) \neq \emptyset$*. We have*

(i) if an item is not packed into any bin in the optimal packing of the grand coalition N*, then its payoff is* 0 *for all* $x \in C(v)$*.*

(ii) there exists a payoff vector $x \in C(v)$ *satisfying each bin gets the same payoff and the payoffs to the items which have the same size are equal, i.e. if* $a_i = a_j$*, then* $x_i = x_j$*.*

Proof. Let $x \in C(v)$ be the core allocation vector of the bin packing game $\langle N, v \rangle$, where $x_1, \cdots, x_n$ are payoffs to players $1, \cdots, n$ and $x_{n+1}, \cdots, x_{n+k}$ are payoffs to bins $1, \cdots, k$.

(i) Suppose $j \in N$ is not packed into any bin in the optimal packing of N, and its payoff $x_j > 0$, then

$$v(N\backslash\{j\}) = v(N).$$

By definition of the core (definition 1.2.8) we know

$$v(N) = \sum_{i=1}^{n+k} x_i,$$

and

$$v(N\backslash\{j\}) \leq \sum_{\substack{i=1 \\ i\neq j}}^{n+k} x_i < \sum_{i=1}^{n+k} x_i = v(N).$$

This gives a contradiction.

(ii) Let x_{l_1}, x_{l_2}, $n+1 \leq l_1, l_2 \leq n+k$ be the payoffs to bins $l_1 - k, l_2 - k$ and we assume $x_{l_1} < x_{l_2}$. Then the new allocation vector x', where

$$x'_{l_1} = x'_{l_2} = \frac{x_{l_1} + x_{l_2}}{2}, \text{ and } x'_i = x_i \text{ for } 1 \leq i \leq n+k, i \neq l_1, l_2$$

is also a core allocation. In fact, $x'_{l_1} + x'_{l_2} = x_{l_1} + x_{l_2}$, so any coalition S involving both x_{l_1} and x_{l_2} naturally meets the expressions $v(S) \leq \sum_{i\in S} x_i$ and $v(N) = \sum_{i\in N} x_i$. Assume S only includes either bin l_1 or bin l_2, then

$$v(S) \leq \sum_{i\in S} x_i = \sum_{\substack{i\in S \\ i\neq l_1}} x_i + x_{l_1} \leq \sum_{\substack{i\in S \\ i\neq l_1}} x_i + \frac{x_{l_1} + x_{l_2}}{2},$$

namely we have

$$v(S) \leq \sum_{i\in S} x'_i.$$

So $x' \in C(v)$, in this way we can find an allocation vector that each bin has the same payoff. Similarly, in the case of 2 items which have the same size, the same argument can be applied. □

What the lemma says is quite reasonable, because the unpacked items contributes 0 value to $v(N)$; also, each bin has the same contribution as well as those items that have equal size should not be paid distinctly.

Now let us return to example 1.3.1, and we assume $C(v) \neq \emptyset$. Since either item 1, 2 or 3 is not packed in the optimal packing, applying Lemma 1.3.2 gives 0 payment to each item. However, $v(\{1, 2, 5\}) = 1$, this implies the payment to each bin should be at least 1. So we have

$$v(N) = \frac{3}{2} + \epsilon = \sum_{i\in N} x_i \geq x_5 + x_6 = 2,$$

where x_5, x_6 are the payoffs to the 2 bins and here we get a contradiction.

Chapter 2

ϵ-Core of Bin Packing Games

2.1 Introduction and Definitions

Example 1.3.1 told us that not all instances of the bin packing game are balanced, in other words, the bin packing game which has an empty core can not grantee a formation of the grand coalition N, because there exists some coalition S such that the players in S earn more than their current payoff. One may naturally arise the question that how to assert the emptiness of the core of a bin packing game?

To answer this question, we first introduce a more general definition with respect to the core, which not only involves all core allocation vectors but also the allocation vectors close to the core. Then, by this generalized definition, we try to find an allocation vector as close to the core as possible, and even if we failed to find the core allocation vector, while at least, we know how far it is from the core.

As an extension of the core, Faigle and Kern (1993) [5] introduced the ϵ-core.

Definition 2.1.1. *Given a bin packing game $\langle N, v\rangle$ and $0 \leq \epsilon \leq 1$, the ϵ-**core** $C_\epsilon(v)$ is defined as the polytope of all vectors $x \in \mathbb{R}^N$ satisfying conditions*

1. $\sum_{i\in N} x_i = v(N)$;
2. $\sum_{i\in S} x_i \geq (1-\epsilon)v(S)$, *for all* $s \subset N$.

The first condition is known as the "efficiency" condition, while the second condition, instead of $\sum_{i\in S} x_i \geq v(S)$, can be interpreted as that the government tax players by rate ϵ, so the ϵ is also called the *tax rate*. Evidently, 1-core of any bin packing game is not empty and 0-core is the core. Hence, in order to approach the core as close as we can, we want to know the minimal tax rate $\epsilon_{\min}$ which grantees a nonempty ϵ-core.

As a further remark of the ϵ-core, one may have noticed that if the players work individually, then the second condition says, $x_i \geq (1-\epsilon)v(i)$ for all $i \in N$, so the

"individual rationality"(1.4) may not hold in some cases. However, in the case of bin packing games, $v(i) = 0$ for all $i \in N$, all ϵ-core allocations of bin packing games are indeed individual rational. Namely we have, for all $x \in C_\epsilon(v)$, x is a imputation (definition 1.2.7).

Parallel to the ϵ-core, Faigle and Kern (1993) [5] also extended the balanced games to the ϵ-balanced games.

Definition 2.1.2. *A game $v \in G^N$ is **ϵ-balanced** if for each balanced map $\lambda : 2^N \backslash \{\emptyset\} \to \mathbb{R}^+$, we have*

$$(1-\epsilon) \sum_{S \in 2^N \backslash \{\emptyset\}} \lambda(S) v(S) \leq v(N). \tag{2.1}$$

As stated in Theorem 1.2.9, a game is balanced if and only if the core is not empty. As such, we may ask whether ϵ-balanced games have the similar property? The answer is stated as below.

Theorem 2.1.3. *[5] The game $v \in G^N$ is ϵ-balanced if and only if $C_\epsilon(v) \neq \emptyset$.*

Proof. Consider the linear program (LP)

$$\begin{aligned} \min & \sum_{i \in N} x_i \\ \text{s.t.} & \sum_{i \in S} x_i \geq (1-\epsilon) v(S) \text{ for all } S \in 2^N \backslash \{\emptyset, N\}. \end{aligned}$$

Note that $C_\epsilon(v) \neq \emptyset$ if and only if the optimal objective value of (LP) is less than or equal to $v(N)$. Its dual problem (DP) is

$$\begin{aligned} \max & \sum_{S \in 2^N \backslash \{\emptyset, N\}} (1-\epsilon) \lambda(S) v(S) \\ \text{s.t.} & \sum_{S \in 2^N \backslash \{\emptyset, N\}} \lambda(S) e^S = e^N, \\ & \lambda(S) \geq 0. \end{aligned}$$

The constraints of (DP) implies λ is a balanced map (1.1). Then $C_\epsilon(v) \neq \emptyset$ if and only if the optimal objective value of (DP) does not exceed $v(N)$ and this holds if and only if v is ϵ-balanced (2.1.2). □

So far so good, the preparations for analysis of bin packing games have been well done. Next we look at the linear program of bin packing games and based up on this to define the fractional bin packing game. Then, by studying the fractional bin packing game, we will show a sufficient and necessary condition for the existence of the ϵ-core. At the end of this chapter, we show some results about the bound of the minimal tax rate ϵ_{min} for any instance of bin packing games.

2.2 Fractional Bin Packing Games

Consider the bin packing game v. As we have shown in the proof of Theorem 1.2.9, the core $C(v) \neq \emptyset$ if and only if the linear program (LP)

$$(LP) \qquad \min \sum_{i \in N} x_i$$
$$\text{s.t.} \sum_{i \in S} x_i \geq v(S) \text{ for all } S \in 2^N \backslash \{\emptyset\}$$

has an optimal objective value $v(N)$.

By Lemma 1.3.2, if $C(v) \neq \emptyset$, then there exists an optimal solution of (LP) allocating the same payoff x_0 to each bin. Furthermore, instead of considering all coalitions $S \in 2^N \backslash \{\emptyset\}$, we restrict S by consisting of only one bin and some subset $f_j \subset I = \{1, 2, \cdots, n\}$ of items with total size

$$\sigma(f_j) = \sum_{i \in f_j} a_i < 1, \quad j = 1, 2, \cdots$$

Let us call such f_j the *feasible set* and denote by $\mathcal{F}$ the collection of all different feasible subsets and $\sigma = (\sigma(f_j)) \in \mathbb{R}^{\mathcal{F}}$ the *total size vector*. Now our allocation problem can be written in the form

$$(AP) \qquad \min\ kx_0 + \sum_{i=1}^{n} x_i$$
$$\text{s.t. } x_0 + \sum_{i \in f_j} x_i \geq \sigma(f_j) \text{ for all } f_j \in \mathcal{F},$$
$$x_0, x_i \geq 0.$$

It is easy to see that (AP) is equivalent to (LP), and we give a strict proof in the following lemma.

Lemma 2.2.1. *Problems (LP) and (AP) are equivalent.*

Proof. By Lemma 1.3.2 we see the objective functions of (LP) and (AP) are equivalent, so (AP) is obtained from (LP) by relaxing S to be one bin and the feasible set of all possibilities. Then the feasible solution of (LP) is also feasible for (AP). On the other side, consider the feasible solution x_0, x_i, $1 \leq i \leq n$ of (AP), and for any $S \in 2^N \backslash \{\emptyset\}$, suppose S consists of $k' \leq k$ bins and items $I' \subset I$, and $f_1, f_2, \cdots, f_{k'}$

be the optimal packing of coalition S, then we have

$$\begin{aligned} x_0 + \sum_{i \in f_1} x_i &\geq \sigma(f_1) \\ x_0 + \sum_{i \in f_2} x_i &\geq \sigma(f_2) \\ &\vdots \\ x_0 + \sum_{i \in f_{k'}} x_i &\geq \sigma(f_{k'}) \end{aligned}$$

Summing up these inequalities yields

$$k'x_0 + \sum_{i \in S} x_i = k'x_0 + \sum_{j=1}^{k'} \sum_{i \in f_j} x_i \geq \sum_{j=1} k'\sigma(f_j) = v(S).$$

This implies x_0, x_i, $1 \leq i \leq n$ are also feasible for (LP), and we are done. □

Let $\mathcal{F} = \{f_1, f_2, \cdots, f_{|\mathcal{F}|}\}$, the dual of (AP) is formulated as the *fractional bin packing game*, where the value function $v_{FRA}(N)$ is defined by solving

$$\begin{aligned} (INT) \qquad & \max \ \sigma^T y \\ & \text{s.t.} \ \sum_{j=1}^{|\mathcal{F}|} y_{f_j} \leq k, \\ & \sum_{\substack{j=1 \\ f_j \ni i}}^{|\mathcal{F}|} y_{f_j} \leq 1 \quad (i = 1, 2, \cdots, n), \\ & y \geq 0. \end{aligned} \tag{2.2}$$

Note that the corresponding integer programming of (FRA) is the value function of the bin packing game, namely

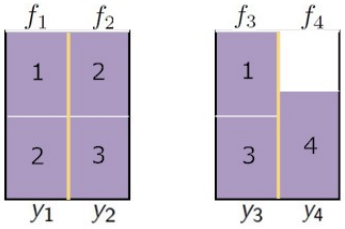
$$\begin{aligned} (FRA) \qquad & \max \ \sigma^T y \\ & \text{s.t.} \ \sum_{j=1}^{|\mathcal{F}|} y_{f_j} \leq k, \\ & \sum_{\substack{j=1 \\ f_j \ni i}}^{|\mathcal{F}|} y_{f_j} \leq 1 \quad (i = 1, 2, \cdots, n), \\ & y \in \{0, 1\}^{\mathcal{F}}. \end{aligned} \tag{2.3}$$

A feasible solution y of (FRA) is called the *feasible packing vector* of the fractional bin packing game. The feasible packing vector as well as the associated feasible sets construct the *feasible packing* of the fractional bin packing game. So for a feasible packing, each feasible set f_j has a corresponding *fractional index* y_j. In the following context, we refer to v_{INT} and v_{FRA} as the value function of the bin packing game and the corresponding fractional bin packing game respectively. Thus, by our above analysis, we have

Theorem 2.2.2. *Consider the bin packing game $\langle N, v_{INT}\rangle$ and the fractional bin packing game $\langle N, v_{FRA}\rangle$, then $C(v_{INT}) \neq \emptyset$ if and only if $v_{FRA}(N) = v_{INT}(N)$.*

Proof. By Lemma 2.2.1, we know (AP) is equivalent to (LP), and (FRA) is the dual of (AP), so (FRA) and (LP) have the same optimal value, which means $C(v_{INT}) \neq \emptyset$ if and only if $v_{FRA}(N) = v_{INT}(N)$. □

To further understand the fractional bin packing game, we review the example 1.3.1. Given 2 bins and 4 items with sizes $a_1 = a_2 = a_3 = 1/2$, $a_4 = 1/2 + \epsilon$. By solving the linear program (FRA), we get $v_{FRA}(N) = 7/4 + \epsilon/2$, and the optimal packing is depicted in the picture.



As is shown above, f_1, f_2, f_3, f_4 are the *optimal feasible sets*, i.e. $f_1 = \{1, 2\}$, $f_2 = \{2, 3\}$, $f_3 = \{1, 3\}$, $f_4 = \{4\}$. Besides, y_1, y_2, y_3, y_4 are the corresponding *optimal fractional indexes*. Note that, a feasible packing should always meet the constraints of (FRA), which, in this example is

$$\begin{aligned} &\text{(a)} \quad y_1 + y_2 + y_3 + y_4 \leq 2; \\ &\text{(b)} \quad \text{item 1: } y_1 + y_3 \leq 1; \\ &\qquad\quad \text{item 2: } y_1 + y_2 \leq 1; \\ &\qquad\quad \text{item 3: } y_2 + y_3 \leq 1. \end{aligned}$$

The first constraint says the sum of the components of y should be less than or equal to the number of bins, while the second condition indicates that for each item, no matter how many feasible sets it appeared, the total sum of these parts can not exceed 1. This can be interpreted as that every item can be split into many fractional parts so as to form large feasible sets, as in our example, item 1 appears in both

f_1 and f_3, with $y_1 = y_3 = 1/2$. However, this can not happen in the 'integer' bin packing game because only integer values the y can take. This advantage explains why v_{FRA} has larger possibilities to be greater than v_{INT} for the same player set N.

2.3 Sufficient and Necessary Condition

Based on former analysis, now we recognize that the fractional bin packing game plays a crucial role of analyzing the existence of the core of the bin packing game. In fact, by calculations of both $v_{FRA}(N)$ and $v_{INT}(N)$, we immediately know the distance $v_{FRA}(N) - v_{INT}(N)$, if it is vanished, then we claim $C(v_{INT})$ exists; however, what does it imply if the distance is nonzero? it is clear that $C(v_{FRA})$ is always nonempty, we may speculate that the distance $v_{FRA}(N) - v_{INT}(N)$ indicates how far is the bin packing game from having a core allocation, which is to say

Theorem 2.3.1. *For a bin packing game $\langle N, v_{INT}\rangle$, ϵ-core $C_\epsilon(v_{INT}) \neq \emptyset$ if and only if*

$$\epsilon \geq \frac{v_{FRA}(N) - v_{INT}(N)}{v_{FRA}(N)}. \tag{2.4}$$

Proof. ($\Rightarrow$) Suppose $x \in C_\epsilon(v_{INT})$, by definition of the ϵ-core we have

$$\sum_{i\in N} x_i = v_{INT}(N) \text{ and } \sum_{i\in S} x_i \geq (1-\epsilon)v(S), \text{ for all } S \in 2^N \backslash \{\emptyset\}.$$

Let $x' = (1-\epsilon)^{-1}x$, then

$$\sum_{i\in N} x'_i = \frac{1}{1-\epsilon}\sum_{i\in N} x_i = \frac{1}{1-\epsilon} v_{INT}(N)$$

and

$$\sum_{i\in S} x'_i = \frac{1}{1-\epsilon}\sum_{i\in S} x_i \geq v(S), \text{ for all } S \in 2^N \backslash \{\emptyset\}.$$

So x' is a feasible solution of (LP) (see section 2.2). From lemma 2.2.1 we know (LP) has the same optimal objective with (AP), so we have

$$v_{FRA}(N) \leq \sum_{i\in N} x'_i = \frac{1}{1-\epsilon} v_{INT}(N),$$

namely,

$$\epsilon \geq \frac{v_{FRA}(N) - v_{INT}(N)}{v_{FRA}(N)}.$$

($\Leftarrow$) If inequality (2.4) holds, then we have

$$v_{FRA}(N) \leq \frac{1}{1-\epsilon} v_{INT}(N).$$

Suppose x' is the optimal solution of (LP), then

$$\sum_{i \in N} x'_i = v_{FRA}(N) \leq \frac{1}{1-\epsilon} v_{INT}(N) \text{ and } \sum_{i \in S} x'_i \geq v(S), \text{ for all } S \in 2^N \backslash \{\emptyset\}.$$

Let

$$x = (1-\epsilon)x' + \frac{v_{INT}(N) - (1-\epsilon)\sum_{i \in N} x'_i}{|N|},$$

we get

$$\sum_{i \in N} x_i = (1-\epsilon)\sum_{i \in N} x'_i + v_{INT}(N) - (1-\epsilon)\sum_{i \in N} x'_i = v_{INT}(N)$$

and

$$\sum_{i \in S} x_i \geq (1-\epsilon)\sum_{i \in S} x'_i \geq (1-\epsilon)v(S) \text{ for all } S \in 2^N \backslash \{\emptyset\}.$$

This implies $x \in C_\epsilon(v_{INT})$. □

Let $\epsilon_{\min}(N) = 1 - v_{INT}(N)/v_{FRA}(N)$, then the $\epsilon_{\min}(N)$ is the minimal ϵ such that $C_\epsilon(v_{INT}) \neq \emptyset$ and $\epsilon_{\min}$ are referred to as the *minimal tax rate*. To further study the emptiness of ϵ-core, now we are concerned about the value of $v_{INT}(N)/v_{FRA}(N)$. However, before calculating $\epsilon_{\min}(N)$, we first look at its upper and lower bound over all instances of bin packing games.

For example, we consider an arbitrary bin packing game $\langle N, v_{INT} \rangle$ and the associated fractional bin packing game $\langle N, v_{FRA} \rangle$, where N consists of n items and k bins. If all items are packed in the optimal packing of N, then $v_{INT}(N) = v_{FRA}(N)$, which results in a nonempty core ($\epsilon_{\min} = 0$); otherwise half capacity of bins should be filled, i.e. $v_{INT}(N) \geq k/2$, besides, it is clear that $v_{FRA}(N) \leq k$. We get

$$0 \leq \epsilon_{\min} \leq 1 - \frac{k/2}{k} = \frac{1}{2}.$$

Now we can claim that 1/2-core of the bin packing game always exists. Obviously, zero is a tight lower bound of $\epsilon_{\min}$, so one may ask is 1/2 also a tight upper bound? If it is not, can we find a sharper bound? More detail is discussed in the following section.

2.4 Bound of Tax Rate

We use the Example 1.3.1 to show that if $\epsilon < 1/7$, there always exists an instance of the bin packing game which has an empty core. Given 2 bins and 4 items with sizes $\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}+\epsilon$, it is easy to compute that $v_{INT}(N) = 3/2+\epsilon$ and $v_{FRA}(N) = 7/4+\epsilon/2$ (we will discuss how to compute them in Chapter 4). So we have

$$\epsilon_{min}(N) = 1 - \frac{v_{INT}(N)}{v_{FRA}(N)} = 1 - \frac{3/2+\epsilon}{7/4+\epsilon/2} = \frac{1-2\epsilon}{7+2\epsilon} < \frac{1}{7}.$$

It is easy to see that $\epsilon_{min}(N)$ converges to $1/7$ as ϵ goes to 0. Moreover, Woeginger (1995) [16] proved that $\epsilon_{min} \leq 1/3$ for all instances of bin packing games. In the following we will show another proof of this result.

Theorem 2.4.1. *[16] For any instance of the bin packing game $\langle N, v_{INT}\rangle$, the minimal tax rate $\epsilon_{min}(N) \leq 1/3$.*

Proof. Refer to [16]. □

Let UB be the upper bound of the minimal tax rate ϵ_{min} for any instance of the bin packing game, then based on above discussions we have

$$\frac{1}{7} \leq UB \leq \frac{1}{3}.$$

Now we are concerned about the exact value of UB, or ask that can we shrink the bound of UB? We have 2 ways of performing this task: one is to find an instance N of the bin packing game, such that the $\epsilon_{min}(N) = \alpha \in (1/7, 1/3]$, then we can claim that $\alpha \leq UB \leq 1/3$; another way is to theoretically prove a smaller bound of the ϵ_{min} for any instance of bin packing games.

Faigle and Kern [6] mentioned a conjecture which is proposed by Woeginger.

Conjecture 2.4.2. *[6] There exists a universal constant $C > 0$ such that each bin packing game $\langle N, v_{INT}\rangle$ admits an allocation vector $x \in \mathbb{R}^n$ with the properties*

1. $\sum_{i\in N} x_i \leq v(N) + C$;
2. $\sum_{i\in S} x_i \geq v(S)$ *for all subsets* $S \subset N$.

Based on our former analysis, the above conjecture can be expressed in relation of the bin packing game and the corresponding fractional bin packing game, i.e.

Conjecture 2.4.3. *For any instance of the bin packing game $\langle N, v_{INT}\rangle$, and the associated fractional bin packing game $\langle N, v_{FRA}\rangle$, there exists a universal constant $C > 0$ such that*

$$v_{FRA}(N) - v_{INT}(N) \leq C. \tag{2.5}$$

Divided by $v_{FRA}(N)$, the inequality (2.5) becomes

$$\epsilon_{min}(N) = \frac{v_{FRA}(N) - v_{INT}(N)}{v_{FRA}(N)} \leq \frac{C}{v_{FRA}(N)}.$$

Let N^* be the instance of the bin packing game such that the $\epsilon_{min}(N^*)$ reaches the upper bound UB, and we assume $v_{INT}(N^*) \geq k/2$, otherwise all items are packed in the optimal packing of N, and we have $v_{FRA}(N^*) = v_{INT}(N^*)$. If C is a tight bound, then $v_{FRA}(N) \geq k/2 + C$ and this gives

$$UB = \epsilon_{min}(N^*) \leq \frac{C}{v_{FRA}(N^*)} \leq \frac{C}{k/2 + C} \leq \frac{C}{1 + C}.$$

Therefore, if we can show this conjecture by $C < 1/2$, then a smaller bound of UB would hold, i.e. $UB \leq C/(1 + C) < 1/3$. Faigle and Kern (1998) [6] proved that if item sizes are all strictly greater than $1/3$, then the bound $C \leq 1/4$. Kuipers [10] (1998) in the same year showed in this case $UB = 1/7$.

Intuitively thinking, the small items will not enlarge the gap between v_{INT} and v_{FRA} because we can always 'greedily' improve the value of the bins by adding small items. Besides, we know that $v_{FRA} - v_{INT} \leq 1/4$ for any instance of the bin packing game, with all item sizes strictly greater than $1/3$. So we have sufficient reasons to believe that its bound C is not a large number. Moreover, we have verified this by computational experiments and found that the maximum value of C is $1/4$. Although this is not a proof of the bound, yet it is useful for computing a bound of v_{INT}, since in most cases v_{FRA} is very close to v_{INT}.

Chapter 3

Complexity Results

3.1 Introduction

As a summary of previous chapters, we have introduced the basic conceptions from cooperative games, and further studied a specific case of the cooperative game, which is referred to as the bin packing game. Based on the theory of cooperative games, a central question of the bin packing game is how to fairly allocate the total profits of the grand coalition N to each player. Then the problem turns out to be seeking out the core of the bin packing game, however, not all instances of bin packing games are balanced, which is to say the problem occurs when the core is empty.

An alternative choice is to find a nonempty ϵ-core, obviously, 1-core of any bin packing game is not empty. Thus, one may be eager to know the minimal ϵ that ensures a nonempty ϵ-core. Motivated by this idea, we further defined the fractional bin packing game v_{FRA} and found that the existence of the ϵ-core lies on the quotient v_{INT}/v_{FRA}.

Following up this clue, it seems that the next step would be working on the algorithms for computing v_{INT} and v_{FRA}, yet one question should be answered before doing so. Does there exist a polynomial algorithm for finding v_{FRA} or v_{INT}? If the polynomial algorithm for them does not exist at all, then instead, another idea is to find good approximation algorithms (non-exact but in polynomial time) for them.

In this chapter we focus on analyzing the time complexity of calculating v_{INT} and v_{FRA}, rather than solving them. Moreover, we will show that to assert the existence of ϵ-core or even to test whether an allocation vector is in the core or not is $\mathcal{NP}$-hard.

3.2 $\mathcal{NP}$ and $\mathcal{NP}$-complete

Probably most readers have some intuitive idea about what a problem is and what an algorithm is, and what is meant by the running time of an algorithm. Although this intuition has been sufficient to understand the substance of the matter, in some cases it is important to formalize this intuition. This is particularly the case when we deal with concepts like $\mathcal{NP}$ and *$\mathcal{NP}$-complete*. In the following context, we introduce some concepts and methodologies from the computational complexity theory.

3.2.1 $\mathcal{P}$ and $\mathcal{NP}$

An optimization problem is the problem of finding the best solution from all feasible solutions. More specifically, given an instance $I = (S, f)$ of optimization problem P, where S is the set of all feasible solutions and $f : x \to f(x) \in \mathbb{R}$, $x \in S$ is called the objective function. The problem P is to find a $x^* \in S$ maximizing or minimizing $f(x^*)$.

In order to classify optimization problems into the several classes, like $\mathcal{P}$ and $\mathcal{NP}$, we first introduce the decision problem, which is only required to answer 'yes' or 'no', and later we will see every optimization problem has its decisional version. Then by studying the hardness of solving these decision problems, we are able to classify the optimization problems.

Definition 3.2.1. *A **decision problem** P is a set of instance I that can be partitioned into 'yes' and 'no' instances I_Y, I_N such that $I = I_Y \cup I_N$ and $I_Y \cap I_N = \emptyset$.*

We take an example to account for what the decision problem is. Given a bin packing game $\langle N, v\rangle$, where N consists of k bins and n items of sizes $a_1, \cdots, a_n$. For a real number $0 < s \leq k$, the question is

For the bin packing game $\langle N, v\rangle$, is $v(N)$ greater than or equal to s?

Simply speaking, the answers to decision problems only consists of 'yes' or 'no' so for this problem, if we answer 'yes' , then we need to be able to give a feasible packing such that $v(N) \geq s$; otherwise, an optimal packing is needed to show $v(N) < s$. It is easy to see that if the optimization problem is solvable, so is the corresponding decision problem, namely to say, the decision problem seems to be easier than the optimization problem. Reversely, if the decision problem is polynomially solvable, can we solve the optimization problems in poly-time? We would say for quite many discrete optimization problems the answer is yes. If the space of the output only includes finite many elements, by binary search we can compute the optimal value via solving the corresponding decision problem.

Nevertheless, in many cases decision problems are so difficult that we can not solve them within an acceptable time. Therefore, the rest of the possibility is to 'guess' a right answer, which in above example is to guess a feasible packing. Then we can certify the value of this packing and further conclude if it is no less than s the answer to the decision problem is 'yes'. Such problems are in the class $\mathcal{NP}$, which will be described more precisely below, is a class of problems that might be very difficult to solve.

In particular, $\mathcal{NP}$ does not mean "not polynomial time", while it stands for "nondeterministic polynomial time". The class $\mathcal{NP}$ consists, roughly speaking, of all those questions with the property that for any input that has a 'yes' answer, there is a "certificate" from which the correctness of this answer can be derived in polynomial time. Accordingly, the class of problems solvable in polynomial time is usually denoted by $\mathcal{P}$. Clearly, $\mathcal{P} \subset \mathcal{NP}$.

Given decision problem P and instance $I \in P$. We know that the data in computers are essentially restored in binary form, and before solving the instance I, it is encoded and restored in the memory, we denote $|I|$ as the binary encoding length of the instance.

Definition 3.2.2. *The decision problem $P \in \mathcal{P}$ if P has an algorithm A and polynomial p such that*

1. *$A(I)$ is 'yes' $\Leftrightarrow$ I is 'yes' instance;*
2. *$t_A(I) \leq p(|I|)$, for all instances $I \in P$,*

where $A(I)$ is the output of algorithm A when solving instance I, and $t_A(I)$ is the corresponding computation time.

By the first condition, the algorithm A identifies the 'yes' instance and the 'no' instance of I, that is to say A solves I; the second condition indicates A is a polynomial algorithm. Accordingly, the definition of $\mathcal{NP}$ is given by

Definition 3.2.3. *The decision problem $P \in \mathcal{NP}$ if P has an algorithm A and polynomial p such that*

1. *there exists a certificate $z(I)$ for all 'yes' instances $I \in P$, $|z(I)| \leq p(|I|)$;*
2. *$A(I, z(I))$ is 'yes' $\Leftrightarrow$ I is 'yes' instance;*
3. *$t_A(I, z(I)) \leq p(|I|)$, for all instances $I \in P$ and any $z(I)$,*

where $A(I, z(I))$ is the output of algorithm A for verifying certificate $z(I)$ and $t_A(I, z(I))$ is the corresponding computation time.

Condition 1 requires the encoding length of the certificate $z(I)$ be polynomially bounded and in condition 2, the algorithm A is used for verifying the 'yes' instance of I. Besides, as formulated in the last condition, the computation time of the algorithm A should be polynomial.

3.2.2 Reductions and $\mathcal{NP}$-complete

Within the class $\mathcal{NP}$ there are $\mathcal{NP}$-*complete* problems, which are defined to be the hardest problems in the class $\mathcal{NP}$. However, the word 'hardest' is not able to convince us whether a problem is really the hardest, the concepts like polynomial reductions and transformations between problems are needed.

Definition 3.2.4. *(Turning Reduction) Given two problems P and Q, say P* ***polynomially reduces*** *to Q ($P \propto Q$), if the following is true: assuming a poly-time algorithm for Q, there is a poly-time algorithm for P.*

Intuitively speaking, problem Q is at least as hard as P, and we say P and Q are *polynomially equivalent* if $P \propto Q$ and $Q \propto P$.

Definition 3.2.5. *(Karp Reduction) Given decision problems P and Q, say P* ***polynomially transforms*** *to Q if the following is true: there is a polynomial time algorithm (the transformation) $T : P \to Q$, such that instance $I \in P =' yes' \Leftrightarrow$ instance $T(I) \in Q =' yes'$.*

As the 'hardest' problem, the $\mathcal{NP}$ class is characterized as below

Definition 3.2.6. *Given decision problem Q, then Q is* $\mathcal{NP}$***-complete*** *if*

1. *$Q \in \mathcal{NP}$;*
2. *for all $P \in \mathcal{NP}$, P polynomially transforms to Q.*

A decision problem Q in $\mathcal{NP}$ is $\mathcal{NP}$-complete if every problem in $\mathcal{NP}$ can be polynomially transformed to Q. Cook (1971) [3] proved that the problem SATISFIABILITY (SAT) is $\mathcal{NP}$-complete. Since 1971, showing that problem Q is $\mathcal{NP}$-complete is 'easy' , because we just need to show $Q \in \mathcal{NP}$ and to find a polynomial transformation $T : SAT \to Q$. Surprisingly, there are a great many prominent combinatorial optimization problems that are $\mathcal{NP}$-complete, like 3-SATISFIABILITY, 3–DIMENSIONAL MATCHING, VERTEX COVER, PARTITION, HAMILTONIAN CYCLE, CLIQUE etc. The list of $\mathcal{NP}$-complete problems is endless.

At present, we generally distinguish between the polynomially solvable problems and the $\mathcal{NP}$-complete problems, although there is no proof that these two concepts are really distinct. For a large number of combinatorial optimization problems, one

has been able to prove either a problem is solvable in polynomial time, or that is $\mathcal{NP}$-complete. In addition, we call the optimization problem *$\mathcal{NP}$-hard* if the corresponding decision problem is $\mathcal{NP}$-complete.

Moreover, among $\mathcal{NP}$-complete problems, we consider a more 'difficult' class of problems, which is referred to as the *strongly $\mathcal{NP}$-complete*. Let P be a problem, $|I|$ the (binary) encoding length of any instance $I \in P$, and $f : \mathbb{R}^+ \to \mathbb{R}^+$ be some nonnegative function, denote by P_f *the restricted version of problem* P, where

$$P_f = \{I \in P \mid number(I) \leq f(|I|)\}.$$

Definition 3.2.7. *Problem P is **strongly/unary** $\mathcal{NP}$-complete ($\mathcal{NP}$-hard) if there exists a polynomial function p such that P_p is $\mathcal{NP}$ complete ($\mathcal{NP}$-hard).*

The problem remains $\mathcal{NP}$-hard even if 'all numbers' are restricted to be small, i.e. polynomial in unary encoding. For instance, 3-PARTITION is strongly $\mathcal{NP}$-complete while PARTITION is not.

3.3 Characteristic Functions

By using reductions, now we are able to prove the complexity results of the bin packing game $\langle N, v_{INT}\rangle$ and the corresponding fractional bin packing game $\langle N, v_{FRA}\rangle$, where N consists of k bins and n items. In this section, we particular concern ourselves with characteristic functions of them.

First we consider the number of bins is a fixed constant, and we have

Theorem 3.3.1. *For fixed k , to find $v_{INT}(N)$ is $\mathcal{NP}$-hard.*

Proof. It is not difficult to see the corresponding decision problem is in $\mathcal{NP}$. In fact, for any feasible packing of the bin packing game there are at most k feasible sets, and the value of these feasible sets can be computed in $O(n^2)$. To prove its NP-hardness, we reduce from SUBSET SUM: given a set S of n positive integers, $S = \{s_1, s_2, \cdots, s_n\}$, and an integer $X \geq s_i$, $i = 1, \cdots, n$, does there exist $S' \subset S$ such that $\sum_{s_i \in S'} s_i = X$?

We construct a bin packing game with $k \geq 1$ bins, and item set $I = \{1, 2, \cdots, n+k-1\}$, with sizes $a_1, a_2, \cdots, a_{n+k-1}$, where

$a_i = \frac{s_i}{X}$, for $1 \leq i \leq n$,

$a_i = 1$, for $n+1 \leq i \leq n+k-1$.

The question is: does there exist a packing with total size no less than k?

($\Rightarrow$) If SUBSET SUM has a solution, then there exists $S' \subset S$ such that $\sum_{s_i \in S'} s_i = X$. Let $I' = \{a_i | s_i \in S'\}$, then $\sum_{a_i \in I'} a_i = \sum_{s_i \in S'} \frac{s_i}{X} = 1$. We put

I' into the first bin and the remaining $k-1$ items $n+1, \cdots, n+k-1$ into the remaining $k-1$ bins, thereby having $v_{INT}(N) \geq k$.

($\Leftarrow$) Assume $v_{INT}(N) \geq k$, then each bin is fully filled by items. Let $I_0 = \{1, 2, \cdots, n\}$, then the rest of items excluding the set I_0 have total size $k-1$. We can always find an optimal feasible set I_1 such that $I_1 \subset I_0$ and $\sum_{a_i \in I_1} a_i = 1$, which also indicates a solution of SUBSET SUM. □

Next we consider the time complexity of computing v_{INT} for arbitrary number of bins.

Theorem 3.3.2. *For arbitrary k, to find $v_{INT}(N)$ is strongly $\mathcal{NP}$-hard.*

Proof. We prove by reduction from 3-PARTITION, which is already known as strongly $\mathcal{NP}$-hard. Given a set S of $3t$ positive integers, denote by $S = \{b_1, b_2, \cdots, b_{3t}\}$, and $\sum_{i=1}^{3t} b_i = t \cdot B$ ($B/4 < b_i < B/2$), does there exist a partition $S_1, S_2, \cdots, S_t$ such that the sum of the numbers in each subset is equal?

W.l.o.g, we assume $k \leq n$, otherwise we can conclude $v_{INT}(N) = \sum_{i=1}^{n} a_i$ by packing each item in each bin. We construct a bin packing game of $k = t$ bins, and item set $I = \{1, 2, \cdots, 3t\}$, with sizes $a_1, a_2, \cdots, a_{3t}$ where $a_i = b_i/B$. The question is: does there exist a packing with total size greater than or equal to k? It is easy to verify that the answer is "yes" if and only if 3-PARTITION has a solution. □

From above theorems, we see computing v_{INT} is $\mathcal{NP}$-hard, although frustrating, it at least gives us an information that the polynomial algorithm does not exist unless $\mathcal{P} = \mathcal{NP}$. By the same approach, we further study the complexity of computing v_{FRA}, is it as difficult as solving v_{INT}? We answer this question in the theorem below.

Theorem 3.3.3. *For fixed k ($k \geq 2$), to find $v_{FRA}(N)$ is $\mathcal{NP}$-hard.*

Proof. First we show the corresponding decision problem is in $\mathcal{NP}$. Consider the linear program of the fractional bin packing game (2.2), we know from the theory of the linear programming that the optimal solution y^* lies in the basic solution set, so y^* has at most $n+1$ nonzero entries, and the objective function $\sigma^T y$ can be computed in $O(n^2)$, which implies the decision problem is in the class $\mathcal{NP}$. Next we show this problem is $\mathcal{NP}$-hard by reducing from PARTITION.

Given a set S of n positive integers, $S = \{b_1, \cdots, b_n\}$, and $\sum_{i=1}^{n} b_i = 2B$ ($b_i \leq B$), does there exist S_1, S_2 such that $S_1 \bigcup S_2 = S$, $S_1 \bigcap S_2 = \emptyset$ and $\sum_{b_i \in S_1} b_i = \sum_{b_i \in S_2} b_i = B$?

Now we construct a bin packing game of k bins, and item set $I = \{1, 2, \cdots, n+k-1\}$, with item sizes $a_1, a_2, \cdots, a_{n+k-1}$, where

$a_i = \frac{b_i}{B}$, for $i \in \{1, \cdots, n\}$,

$a_i = 1$, for $i \in \{n+1, \cdots, n+k-1\}$.

The question is: does there exist a packing with total size no less than k?

($\Rightarrow$) If PARTITION has a solution, then there exist sets S_1, S_2 satisfying $S_1 \bigcup S_2 = S$, $S_1 \bigcap S_2 = \emptyset$ and $\sum_{b_i \in S_1} b_i = \sum_{b_i \in S_2} b_i = B$. Let $I_1 = \{a_i | b_i \in S_1\}$, $I_2 = \{a_i | b_i \in S_2\}$, so $\sum_{a_i \in I_1} a_i = \sum_{a_i \in I_2} a_i = 1$. We put I_1, I_2 into the first 2 bins, and the remaining $k-2$ items $n+1, n+2, \cdots, n+k-2$ into the remaining $k-2$ bins, thereby having $v_{FRA}(N) \geq k$.

($\Leftarrow$) Assume $v_{FRA}(N) \geq k$, then each bin is full. Let $I_0 = \{1, 2, \cdots, n\}$, then the rest of items excluding the set I_0 have total size $k-1$. We can always find an optimal feasible set I_1 such that $I_1 \subset I_0$ and $\sum_{a_i \in I_1} a_i = 1$. Let $I_2 = I_0 \backslash I_1$, then we have $\sum_{a_i \in I_1} a_i = \sum_{a_i \in I_2} a_i = 1$. So I_1, I_2 are the partition of I_0. By multiplying B on each element of I_1, I_2 we also get a partition of S. □

Now one may ask: can we further conclude the strong $\mathcal{NP}$-hardness for arbitrary number of bins? The answer is probably 'no' . Since computing v_{FRA} is in $\mathcal{P}$ when items sizes greater than a given ϵ (even for arbitrary k).

3.4 Core Membership

Compared to assert the emptiness of the core, an alternative question is: given an instance of a game $\langle N, v \rangle$ and the allocation vector x, is x in the core? In the case of bin packing game $\langle N, v_{INT} \rangle$, now it becomes clear to answer this question, by using similar arguments as we have shown in the previous section.

Theorem 3.4.1. *Given any instance of the bin packing game of fixed number of bins, to assert whether an allocation vector lies in the core is $\mathcal{NP}$-hard and is strongly $\mathcal{NP}$-hard for arbitrary number of bins.*

Proof. We first show the strong $\mathcal{NP}$-hardness for arbitrary number of bins. We prove this by using the same reduction as we have shown in the proof of Theorem 3.3.2: we construct $k = t$ bins and $3k$ items of sizes $a_1, a_2, \cdots, a_{3k}$, where $a_i = b_i/B$ for $1 \leq i \leq 3k$. Clearly, $\sum_{i=1}^{3k} a_i = k$. Let $x = (a_1, a_2, \cdots, a_{3k}, 0, \cdots, 0)^T$ be a payoff vector, where a_i is the payoff to player i and 0 is the payoff to each bin. It is easy to check that $x \in C(v_{INT})$ if and only if 3-PARTITION has a solution. If k is fixed, then a similar reduction from PARTITION can be applied here. □

As a direct corollary of Theorem 3.4.1, to identify the ϵ-core membership of an allocation vector is also $\mathcal{NP}$-hard. Without proof we claim

Corollary 3.4.2. *Given $\epsilon \in [0, 1]$ and any instance of the bin packing game of fixed number of bins, to assert whether an allocation vector lies in the ϵ-core is $\mathcal{NP}$-hard and is strongly $\mathcal{NP}$-hard for arbitrary number of bins.*

3.5 Core Emptiness

Moreover, Liu (2009) [11] proved the $\mathcal{NP}$-hardness of identifying the core emptiness of a given instance of the bin packing game. First we consider the case for fixed number of bins.

Theorem 3.5.1. *[11] The problem of deciding whether an instance of the bin packing game with a fixed number of k bins ($k \geq 2$) has an empty core is $\mathcal{NP}$-hard.*

Proof. We prove this by reduction from PARTITION: Given a set of $2t$ integers $S = \{b_1, b_2, \cdots, b_{2t}\}$, and $\sum_{j=1}^{2t} b_j = 2B$, does there exist subsets S_1, S_2 of S with $S_1 \cap S_2 = \emptyset$ and $S_1 \cup S_2 = S$ satisfying $\sum_{b_j \in S_1} b_j = \sum_{b_j \in S_2} b_j$?

We construct a bin packing game of $k \geq 2$ bins, and item set $I = \{1, 2, \cdots, 2t + 2k\}$ of sizes $a_1, a_2 \cdots, a_{2t+k+1}$, where

$a_i = b_i(1 - 2\epsilon)/2B$, for all $1 \leq i \leq 2t$;

$a_{2t+1} = \epsilon$,

$a_i = 1/2$, for all $2t + 2 \leq i \leq 2t + 2k$.

Next we show that the PARTITION has a solution if and only if the bin packing game has a nonempty core.

($\Rightarrow$) Suppose the PARTITION problem has a solution, i.e. there exist subsets S_1, S_2 of S with $S_1 \cap S_2 = \emptyset$ and $S_1 \cup S_2 = S$ satisfying $\sum_{b_j \in S_1} b_j = \sum_{b_j \in S_2} b_j = B$. Let $I_1 = \{i \mid b_i \in S_1\}$, $I_2 = \{i \mid b_i \in S_2\}$ then we put items of I_1 together with item $2t+1, 2t+2$ into the first bin and items from $2t+3$ to $2t+2k$ into the remaining $k-1$ bins, so we get $v_{INT}(N) = k = v_{FRA}(N)$. By Theorem 2.2.2 we know $C(v_{INT}) \neq \emptyset$.

($\Leftarrow$) If the PARTITION has no solution, then we get $v_{INT}(N) = k - \epsilon$ by putting items from 1 to $2t + 1$ into the first bin and items from $2t + 2$ to $2t + 2k - 1$ into the remaining $k - 1$ bins. In fact, we see

$$\min_{I_1, I_2} \left(\sum_{i \in I_1} a_i - \sum_{i \in I_2} a_i \right) = \min_{S_1, S_2} \left(\sum_{i \in S_1} b_i - \sum_{i \in S_2} b_i \right) \cdot \frac{1 - 2\epsilon}{2B} \geq \frac{1 - 2\epsilon}{2B},$$

where S_1, S_2 are the partition of S, and I_1, I_2 are the associated item set with respect to S_1, S_2, i.e. $I_{1,2} = \{i \mid b_i \in S_{1,2}\}$. Let $\epsilon < \frac{1}{2(1+B)}$, then above inequality becomes

$$\min_{I_1, I_2} \left(\sum_{i \in I_1} a_i - \sum_{i \in I_2} a_i \right) \geq \frac{1 - 2\epsilon}{2B} > \epsilon.$$

Thus we can not expect a subset I' of items $\{1, 2, \cdots, 2t\}$ satisfying $\sum_{i \in I'} a_i \in [1/2 - \epsilon, 1/2]$.

We denote the core allocation vector by $x = (x_1, \cdots, x_{2t+2k}, \cdots, x_{2t+3k})^T$, where $x_{2t+2k+1}, \cdots, x_{2t+3k}$ are the payoffs to each bin. By Lemma 1.3.2, we further assume

$x_i = \alpha$ be the payoff to the item that has size 1/2, $2t + 2 \leq i \leq 2t + 2k$;
$x_i = \beta$ be the payoff to each bin, $2t + 2k + 1 \leq i \leq 2t + 3k$.

Since item $2t+2k$ is not packed into any bin in the optimal packing, we have $\alpha = 0$. Besides, if we consider the coalition consisting of 1 bin and items $2t+2k-1, 2t+2k$, by definition of the core we have

$$\beta = \alpha + \beta \geq 1.$$

Namely $\beta \geq 1$ and $\sum_{i=1}^{2t+2k+3k} x_i \geq k > v_{INT}(N)$, so we get a contradiction and this implies $C(v_{INT}) = \emptyset$. □

Next we continue our problem, but to consider the case when the number of bins is arbitrary.

Theorem 3.5.2. *[11] The problem of deciding whether an instance of the bin packing game with arbitrary number of bins has an empty core is strongly $\mathcal{NP}$-hard.*

Proof. We prove the result by reduction from 3-PARTITION, which is already known as strongly $\mathcal{NP}$-hard. Given a set S of $3t$ positive integers, denote by $S = \{b_1, b_2, \cdots, b_{3t}\}$, and $\sum_{i=1}^{3t} b_i = t \cdot B$ $(B/4 < b_i < B/2)$, does there exist a partition $S_1, S_2, \cdots, S_t$ such that the sum of the numbers in each subset is equal?

Now we construct a bin packing game of $k = t + 1$ bins and item set $I = \{1, 2, \cdots, 4t + 3\}$ of sizes $a_1, a_2, \cdots, a_{4t+3}$, where
$a_i = b_i/B$, $i = 1, 2, \cdots, 3t$;
$a_i = 1 - \epsilon$, $i = 3t + 1, \cdots, 4t$;
$a_i = 1/2$, $i = 4t + 1, 4t + 2, 4t + 3$.

Next we show that the 3-PARTITION has a solution if and only if the bin packing game has a nonempty core.

($\Rightarrow$) Suppose 3-PARTITION has a solution, i.e. there exist subset $S_1, \cdots, S_t$ with $S_i \cap S_j = \emptyset$, $\forall i, j$ and $\cup_{i=1}^{t} S_i = S$ satisfying $\sum_{b_j \in S_i} b_j = B$ for all $1 \leq i \leq 3t$. Let $I_j = \{i \mid b_i \in S_j\}$ for all $1 \leq j \leq t$. By putting $I_1, I_2, \cdots, I_t$ into the first t bins, and items $4t+2, 4t+2$ into the $(t+1)$-th bin, then we get $v_{INT}(N) = k = v_{FRA}(N)$. By Theorem 2.2.2, the core $C(v_{INT}) \neq \emptyset$.

($\Leftarrow$) If the 3-PARTITION does not have a solution, then it is easy to see $v_{INT}(N) < k$. We next show that in any optimal packing of the grand coalition N, at least one of items $4t + 1, 4t + 2, 4t + 3$ is not packed into any bin. Suppose this condition is violated, i.e. all items are packed, then there exists a single item, denoted by $4t + 1$, is not together with item $4t + 2$ or $4t + 3$ in any bin, and we further assume the item $4t + 1$ is packed into the first bin. Since $B/4 < b_i < B/2$, we also have $B/4 < b_i \leq (B - 1)/2$, which yields,

$$\frac{1}{4} < a_i \leq \frac{1}{2} - \frac{1}{2B}.$$

So except for item $4t+3$, the first bin contains exact one another item from the set $\{1, 2, \cdots, 3t\}$. Then we have

$$v_{INT}(N) \leq k - \frac{1}{2B}.$$

Let $\epsilon < 1/(2tB)$, we have $v_{INT}(N) < k - t\epsilon$. However, if we put items from $3t+1$ to $4t$ into the first t bins and items $4t+1, 4t+2$ into the $(t+1)$-th bin, then we get the contradiction

$$v_{INT}(N) \geq t(1-\epsilon) + 1 = k - t\epsilon > v_{INT}(N).$$

From Lemma 1.3.2, in an core allocation the payoff to each of items $4t+1$, $4t+2$ and $4t+3$ is 0. Since a coalition of a single bin and any two of items $4t+1$, $4t+2$ and $4t+3$ has a value of 1, in a core allocation the payoff to each bin is at least 1. As a result, the total payoff of any core allocation is required to be no less than k, which is greater than $v_{INT}(N)$. □

Chapter 4

Exact Algorithms

4.1 Introduction

Based on previous discussions, now we set about our task of identifying the emptiness of the ϵ-core. Theorem 2.3.1 says that the minimal tax rate of a bin packing game is

$$\epsilon_{min}(N) = \frac{v_{FRA}(N) - v_{INT}(N)}{v_{FRA}(N)}.$$

Thus in this chapter, we primarily discuss the algorithms of computing v_{INT} and v_{FRA}.

Working along the route of the linear programming, first we introduce the matrix form of the bin packing game (2.3) and the fractional bin packing game (2.2), which will be easier for computation. The problems in the matrix form requires a input of the *feasible matrix*, and we will present an algorithm that can generate this feasible matrix systematically, according to the item sizes. However, when the number of the different feasible sets grows exponentially large, we are not able to enumerate all feasible vectors, to overcome this difficulty, we still want to generate the feasible matrix automatically, but instead, only the feasible vectors which have the potential to improve the value function are needed. This approach will be later discussed as the column generation method (refer to Gilmore and Gomory [7, 8]).

4.2 Linear Programming

The most straightforward idea of computing v_{INT} and v_{FRA} is to solve the linear program (INT) (2.2) and (FRA) (2.3). However, the formulations of (INT) and (FRA) are not suitable for writing codes or figure out by hand. In this section we introduce the corresponding matrix form of them, which perfectly suits our purpose of calculation.

Consider a bin packing game with $k \geq 1$ bins and item set $I = \{1, 2, \cdots, n\}$, which has sizes $a_1, a_2, \cdots, a_n$. We refer to $a = (a_1, a_2, \cdots, a_n)^T$ as the *size vector*. A vector $b_j \in \{0,1\}^n$ is called a *feasible vector* if $a^T b_j \leq 1$, so the *feasible set* $f_j = \{i \mid b_{ij} = 1,\ b_j \text{ is a feasible vector}\}$. Moreover, let $F = |\mathcal{F}|$ be the number of different feasible vectors, where $\mathcal{F}$ is the set of all feasible sets and $B \in \{0,1\}^{n \times F}$ be the *feasible matrix*

$$B = (b_1\ b_2\ \cdots\ b_F).$$

Then the linear program (INT) and (FRA) can be written in the following *matrix form*.

$$\begin{aligned} &\max\ a^T B y, \\ &\text{s.t. } By \leq e_n, \\ &\quad e_F^T y \leq k, \\ &\quad y \in \{0,1\}^F, \end{aligned} \tag{4.1}$$

where $e_n = (1, 1, \cdots, 1)^T \in \mathbb{R}^n$ and $e_F = (1, 1, \cdots, 1)^T \in \mathbb{R}^F$.

$$\begin{aligned} &\max\ a^T B y, \\ &\text{s.t. } By \leq e_n, \\ &\quad e_F^T y \leq k, \\ &\quad y \in [0,1]^F. \end{aligned} \tag{4.2}$$

Besides, if we let

$$c^T = \begin{pmatrix} a^T & 0 \end{pmatrix}, \quad H = \begin{bmatrix} B \\ e_F^T \end{bmatrix} = [h_1\ h_2\ \cdots\ h_F], \quad g = \begin{bmatrix} e_n \\ k \end{bmatrix},$$

then (4.1) and (4.2) can be equivalently written as

$$\begin{aligned} &\max\ c^T H y \\ &\text{s.t. } Hy \leq g, \\ &\quad y \in \{0,1\}^F, \end{aligned} \tag{4.3}$$

$$\begin{aligned} &\max\ c^T H y \\ &\text{s.t. } Hy \leq g, \\ &\quad y \geq 0. \end{aligned} \tag{4.4}$$

The two problems are *full rank*, which is to say

Lemma 4.2.1. *B and H are full rank, namely,*

$$rank(B) = n,\ rank(H) = \begin{cases} n, & F = n \\ n+1, & F > n \end{cases}.$$

Proof. Every item itself is a feasible set, then B and H has the form

$$B = \begin{bmatrix} 1 & & & \cdots \\ & 1 & & \cdots \\ & & \ddots & \\ & & & 1 \quad \cdots \end{bmatrix}, \; H = \begin{bmatrix} 1 & & \cdots & & * & \cdots \\ & 1 & \cdots & & * & \cdots \\ & & \ddots & & * & \cdots \\ & & & 1 & * & \cdots \\ 1 & 1 & \cdots & 1 & 1 & \cdots \end{bmatrix}.$$

By elementary column transformations, we can see B and H are full rank. □

Under matrix formulations of the bin packing game and the fractional bin packing game, now we can explicitly solve them, although not in polynomial time. Indeed, in the linear program (4.1) and (4.2), the input involves the number of bins k, the size vector a and the corresponding feasible matrix B. However, a general bin packing game does not give the feasible matrix, we need an algorithm to generate it automatically, according to the size vector a. Please refer to appendix A for more detail.

4.3 Column Generation Approach

4.3.1 Simplex Method

Given $A \in \mathbb{R}^{m\times n}, c, b, x \in \mathbb{R}^n, x, b \geq 0$, consider the *standard form* of the linear programming problem of the form

$$\begin{aligned} \max \quad & c^T x \\ \text{s.t.} \quad & Ax = b, \\ & x \geq 0. \end{aligned}$$

$c^T x$ is called the *objective function*, and any nonstandard form of LP problems can be transformed into the standard form. For example, if x is free, let $x^+, x^- \geq 0$, and replace x by $x^+ - x^-$ then we get all nonnegative variables.

Consider $Ax = b$ and $rank(A) = m < n$. Let B be the matrix obtained by choosing m linear independent columns of A, and let N denote the matrix formed by the other columns of A. Then $x_B = B^{-1}b$ is called the *basic solution* and the components of x_B are called *basic variables*. Correspondingly, other variables x_N are called *nonbasic variables* and we call the vector formed by these nonbasic variables *nonbasic solution*. Hence the equation $Ax = b$ can be written in terms of x_B and x_N.

$$Bx_B + Nx_N = b \Rightarrow x_N = B^{-1}b - B^{-1}Nx_N.$$

The objective function

$$c^T x = c_B^T x_B + c_N^T x_N = c_B^T B^{-1} b + (c_N^T - c_B^T B^{-1} N) x_N.$$

Taking derivative of the objective function with respect to b, the *shadow price* of the associated constraint is defined by $\lambda^T = c_B^T B^{-1}$. Similarly, taking derivative with respect to x_N defines the *reduced cost* of a variable $\mu^T = c_N^T - \lambda^T N$. To apply the column generation to the fractional bin packing game, we will use the expression of μ^T to generate possible feasible vectors.

4.3.2 Apply Column Generation to (FRA)

As an application of the simplex method, the column generation method is started by creating a submodel of the fractional bin packing game (*master problem*) which contains a set of feasible vectors. Obviously, the initial set is not necessarily optimal. Then after solving this submodel and using its byproduct shadow price, we are able to formulate an auxiliary model (feasible vector generation model) (*subproblem*). Solving this auxiliary model identifies a feasible vector which is then added into the fractional bin packing game submodel to improve its objective. The fractional bin packing game with this extra feasible vector, is then solved. By using the updated shadow price, the process is then repeated until the submodel contains the optimal set of feasible vectors. By the proof of Theorem 3.3.3 we know the optimal set of feasible vectors consists of at most $n + 1$ feasible vectors. So in practice, the total number of new feasible vectors generated by the auxiliary model is quite small and so the overall algorithm is quite efficient.

We know from the simplex method that, the reduced cost of a variable gives the changing rate of the objective function for one unit increase in the bound of the variable. During a simplex iteration, one of nonbasic variables is introduced into the basis, and consequently a basic variable leaves the basis to become nonbasic. For the case, as in our problem (4.2), where all variables are nonnegative and nonbasic variables are vanished, such an exchange is only of interest (for a maximization problem) when the corresponding component of the reduced cost $c_N^T - \lambda^T N$ is positive. In this particular case, the objective function value (v_{obj}) of the master problem will increase when the value of the corresponding component of x_N is increased. As soon as all components of the reduced cost vector become negative, no improvement in the v_{obj} can be made, and the current basic solution $x_B = B^{-1} b$ is optimal. The iterative solving of the fractional bin packing game submodel and the feasible vector generation program is summarized below.

Algorithm-Column Generation (CG)

```
Initialize master problem
WHILE reduced cost>0
  Solve master problem
  Solve subproblem
  Add the feasible vector to master problem
ENDWHILE
```

The master problem is defined by the LP model of the fractional bin packing game and we restate it as below

$$\begin{aligned} \text{(master problem)} \quad \max\ & a^T B y, \\ \text{s.t.}\ & Bx \le e_n^T, \\ & e_F^T x \le k, \\ & x \in [0,1]^F. \end{aligned}$$

Note that the matrix B only consists of a initial set of feasible vectors, rather than all feasible vectors.

Let $\lambda \in \mathbb{R}^n$ be the shadow price of the first inequality and $\rho \in \mathbb{R}$ be the shadow price of the second inequality. Let z be the newly generated feasible vector, then the value of z should be less than or equal to 1, i.e. $a^T z \le 1$. The corresponding reduced cost $\mu = c_N^T - \lambda^T N = (a^T - \lambda^T)z - \rho$. Our feasible vector generation model is to maximize the reduced cost with respect to z, i.e.,

$$\begin{aligned} \text{(subproblem)} \quad \max\ & (a^T - \lambda^T)z \\ \text{s.t.}\ & a^T z \le 1, \\ & z \in \{0,1\}^n. \end{aligned}$$

Let $p_i = a_i - \lambda_i$, $w_i = a_i$, $i = 1, 2, \cdots, n$ and $W = 1$, then to solve the subproblem is indeed to solve a KNAPSACK problem: given n items $1, 2, \cdots, n$ and each item has a value p_i and weight w_i, and a knapsack with total capacity W, to maximize the total value of the knapsack on condition that the total weight of items do not exceed the capacity W. We know the KNAPSACK problem is $\mathcal{NP}$-hard, thus in each iteration, we solve a $\mathcal{NP}$-hard problem, more specifically, we often use the dynamic programming to solve the KNAPSACK problem, which is known as pseudo-polynomial.

In practical, when the number of feasible sets can not be enumerated, the column generation algorithm can be used to compute the value function of the fractional bin packing game, since it only provides the vector that has the potential to improve the objective value, and without the redundant feasible vectors, the master problem can be efficiently solved. Besides, under some rules of rounding the fractional solution, this also gives a way of approaching the value function of the corresponding bin packing game. More specifically, we give an example to show how it works.

4.3.3 Example

Given 2 bins and 4 items with size vector $a = (1/2, 1/2, 1/2, 2/3)^T$. *Compute* $v_{FRA}(N)$ *by column generation.*

Each item itself is obviously a feasible set, the initial set of feasible vectors can be naturally chosen as below

$$B = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

1st iteration: solve master problem and we get

$$y = (1/3, 1/3, 1/3, 1)^T, \quad v_{FRA}(N) = 7/6, \quad \lambda = (0, 0, 0, 1/6)^T, \quad \rho = 1/2,$$

then solve subproblem we get

$$z = (1, 1, 0, 0)^T, \quad \mu = 1/2.$$

Since reduced cost $\mu > 0$, we add z to the master problem, which gives

$$B = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

2nd iteration: solve master problem and we get

$$y = (0, 0, 0, 1, 1)^T, \quad v_{FRA}(N) = 5/3, \quad \lambda = (0.2342, 0.2342, 0, 0.1351)^T, \quad \rho = 0.5315,$$

then solve subproblem we get

$$z = (0, 1, 1, 0)^T, \quad \mu = 0.2342.$$

Since $\mu > 0$, we add z to the master problem, which gives

$$B = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}.$$

3rd iteration: solve master problem and we get

$$y = (0, 0, 0, 1, 1/2, 1/2)^T, \quad v_{FRA}(N) = 5/3, \quad \lambda = (0, 0.4398, 0, 0.1064)^T, \quad \rho = 0.5602,$$

then solve subproblem we get

$$z = (1, 0, 1, 0)^T, \quad \mu = 0.4398.$$

Since $\mu > 0$, we add z to the master problem, which gives

$$B = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}.$$

4th iteration: solve master problem and we get

$$y = (0, 0, 0, 1/2, 1/2, 1/2, 1/2)^T, \quad v_{FRA}(N) = 11/6, \quad \lambda = (1/6, 1/6, 1/6, 0)^T, \quad \rho = 2/3,$$

then solve subproblem we get

$$z = (1, 1, 0, 0)^T, \quad \mu = 0.$$

Since $\mu = 0$, the algorithm terminates and we get the optimal solution

$$y = (0, 0, 0, 1/2, 1/2, 1/2, 1/2)^T, \text{ and } v_{FRA}(N) = 11/6.$$

Chapter 5

Approximation Algorithms

5.1 Introduction and Terminologies

In chapter 4, we studied the exact but non-polynomial algorithm for bin packing games and fractional bin packing games, however, in most practical cases, we just want to find a nice but not necessarily optimal solution for them, with the requirements that the computation time is in poly-time and the output is as close to the optimum value as possible. This is the motivation of why we study the approximation algorithm. For a given approximation algorithm, how do we evaluate its performance? In other words, given approximation algorithms A and B, what is the reason of saying A performs better or worse than B? In the following we introduce some terminologies about performance measures of approximation algorithms, and later on we go into detail to analyze the performance of the approximation algorithms of computing the value functions of the bin packing game and the fractional bin packing game.

Before start, we first look at the classical bin packing problem (see [2]) which is already know as a $\mathcal{NP}$-hard problem.

(*Classical bin packing problem*) In the classical one-dimensional bin packing problem, we are given a sequence $I = \{1, 2, \cdots, n\}$ items, each with a size $a_i \in (0, 1]$ and are asked to pack them into a minimum number m of subsets (bins) $B_1, B_2, \cdots, B_m$ such that $\sum_{a_i \in B_j} a_i \leq 1$, $1 \leq j \leq m$.

In the case of classical bin packing problem, the standard metric for worst-case performance is the *asymptotic worst-case performance ratio*. For a given instance I and algorithm A, let $A(I)$ be the output of the algorithm, i.e. the number of bins used when A is applied to list I, and denote by $OPT(I)$ the optimal number of bins for a packing of I. The *absolute* worst-case performance ratio is defined to be

$$R_A = \inf\{r \geq 1 : \forall\ I, A(I) \leq r \cdot OPT(I)\}.$$

The *asymptotic* worst-case performance ratio R_A^∞ is defined to be

$$R_A^\infty = \inf\{r \geq 1 : \exists\ C > 0, A(I) \leq r \cdot OPT(I) \text{ for all } I, \text{ with } OPT(I) \geq C\}.$$

Given an instance of the classical bin packing problem, there are some well known simple algorithms: Next Fit (NF), First Fit (FF), and First Fit-Decreasing (FFD). Their absolute worst-case performance ratios are $R_{NF} = 2, R_{FF} = 17/10$ and $R_{FFD} = 11/9$ (refer to [9] for more detail).

So one may ask whether it is possible to apply these algorithms to bin packing games. If possible, what is the worst-case performance ratio for each algorithm? To answer these questions, we first need similar definitions of worst-case performance ratio for bin packing games.

Consider an instance N of the bin packing game and a given approximation algorithm A. The value function of the bin packing game is denoted by $v_{INT}(N)$, and the output value of A is denoted by $A(N)$ when it is applied to N . Similar to the classical bin packing problem, to evaluate the performance of A, the *absolute* worst-case performance ratio R_A is defined to be

$$R_A = \inf\{r \leq 1 : \forall\ N, A(N) \geq r \cdot v_{INT}(N)\}.$$

The *asymptotic* worst-case performance ratio R_A^∞ is defined to be

$$R_A^\infty = \inf\{r \leq 1 : \exists\ C > 0, A(N) \geq r \cdot v_{INT}(N) \text{ for all } N, \text{ with } v_{INT}(N) \geq C\}.$$

Moreover, algorithm A is called an *α-approximation algorithm* if $A(N) \geq \alpha \cdot v_{INT}(N)$, and α is the *approximation ratio*. If $\alpha = R_A$ we say α is a *tight bound*.

Next, we present some approximation algorithms of bin packing games and to discuss their approximation ratios.

5.2 Approximate v_{INT}

To compute v_{INT}, one may naturally think about the greedy algorithm, i.e., compute the largest feasible set in each iteration and pack it into a bin. However, as we have shown in Theorem 3.3.1, the bin packing game is $\mathcal{NP}$-hard even when $k = 1$, thus computing the largest feasible set is also $\mathcal{NP}$-hard. Besides, even if we do not care the time cost of the greedy algorithm, it is also not a reliable algorithm for the bin packing game. For instance, consider 5 items with sizes $a_1 = a_2 = 1/2 - 2\epsilon$, $a_3 = 4\epsilon$, $a_4 = a_5 = 1/2 + \epsilon$. The greedy algorithm gets value $3/2 + \epsilon$ while $v_{INT}(N) = 2 - 2\epsilon$. Therefore, the greed algorithm is neither efficient nor exact for bin packing games.

Look back the algorithms of solving the classical bin packing problem, we want to apply those ideas to compute v_{INT}. To do this, the next we show another version of

the algorithm NF, FF and FFD, which are designed for solving the value function of bin packing games. After that, to see how well do they work, we further prove their worst-case performance ratios. Additionally, we will consider a special case of the bin packing game, i.e. given a constant $B > 0$ and the number of bins less than B, then we propose a $(1-\epsilon)$-approximation algorithm for computing v_{INT}.

5.2.1 Algorithm NF,FF and FFD

Given an instance of the bin packing game, the input includes the number of bins k and the number of items n as well as the item sizes $a_1, a_2, \cdots, a_n$. We will introduce the algorithm Next Fit (NF), First Fit (FF) and First Fit-Decreasing (FFD) for bin packing games. For easy of description, we denote by $b_1, b_2, \cdots, b_k$ the feasible packing which is generated by the algorithm, i.e. the items of the feasible set b_j are packed into the j-th bin. NF, FF, FFD are proceeded as follows.

Next Fit (NF)

Initialize $i_0 = 0$
FOR all bins $1 \leq p \leq k$ DO
  IF $\exists j \in \{i_{p-1}+1, \cdots, n-1\}$, s.t. $\sum_{i=i_{p-1}+1}^{j+1} a_i > 1$
    THEN let $i_p := \min\{j | \sum_{i=i_{p-1}+1}^{j+1} a_i > 1\}$
  ELSE
    THEN let $i_p := n$
  ENDIF
  Pack items into the p-th bin , which is $b_p = \{i_{p-1}+1, i_{p-1}+2, \cdots, i_p\}$
  IF all items are used, i.e. $i_p = n$
    Break
  ENDIF
ENDFOR
Output total size of packed items of all bins

First Fit (FF)

Initialize $b_1 = b_2 = \cdots = b_k = \emptyset$
FOR all items i $(1 \leq i \leq n)$ DO
  Compute the smallest index j, s.t. $\sum_{i \in b_j} a_i \leq 1$
  $b_j := b_j \cup \{i\}$
ENDFOR
Output total size of packed items of all bins

First Fit-Decreasing (FFD)

Sort item sizes by decreasing order $a_1 \geq a_2 \geq \cdots \geq a_n$

```
Apply First Fit algorithm
```

Slightly different from algorithms for the classical bin packing problem, the output of these algorithms is the total size of the packed items, instead of the number of bins. However, the crucial ideas of the algorithms for bin packing games are almost the same with those of for the classical bin packing problem. Having introduced the above approximation algorithm NF, FF and FFD, now we are curious about their performance for the bin packing games.

5.2.2 Performance of NF,FF and FFD

Having been aware of performance measures of the approximation algorithm, in this section we show the approximation ratios of the algorithm NF, FF and FFD.

Theorem 5.2.1. *$R_{NF}^{\infty} = 1/2$.*

Proof. If all items are packed into the k bins by NF, then $NF(N) = v_{INT}(N)$. Suppose NF ends after the k-th iteration and there exists some items not packed into any bin. Let $b_1, b_2, \cdots, b_k$ be the feasible packing obtained by NF, where b_j is the set of items in the j-th bin and let $\sigma(b_j) = \sum_{i \in b_j} a_i$ be the value of the j-th bin, $1 \leq j \leq k$. For all j, if $\sigma(b_j) \geq \frac{1}{2}$, then $A(N) \geq k/2 \geq v_{INT}(N)/2$. If not, we first assume $k \geq 2$ and consider the following 2 cases.

Case1: there exists a $b_j (1 \leq j < k)$, such that $\sigma(b_j) < 1/2$. We have

$$\begin{aligned} NF(N) &= \sum_{i=1}^{j-1} \sigma(b_i) + \sigma(b_j) + \sum_{i=j+1}^{k} \sigma(b_i) \\ &= \sum_{i=1}^{j-1} \sigma(b_i) + \sigma(b_j) + \sigma(b_{j+1}) + \sum_{i=j+2}^{k} \sigma(b_i) \\ &> \frac{j-1}{2} + 1 + \frac{k-j-1}{2} \\ &= \frac{k}{2} \geq \frac{v_{INT}(N)}{2}. \end{aligned}$$

The second last inequality holds since b_{j+1} is not empty, then exists an item $i \in b_{j+1}$, and we know from the process of algorithm NF that i is packed into the $(j+1)$-th bin if $\sigma(b_j) + a_i > 1$, so we have $\sigma(b_j) + \sigma(b_{j+1}) > 1$.

Case2: $\sigma(b_k) < 1/2$, $\sigma(b_j) \geq 1/2$ $(j = 1, 2, \cdots, k)$. We have

$$\begin{aligned} NF(I) &= \sum_{i=1}^{k-1} \sigma(b_i) + \sigma(b_k) \\ &= \sum_{i=1}^{j-2} \sigma(b_i) + \sigma(b_{k-1}) + \sigma(b_k) \\ &\geq \frac{k-2}{2} + 1 \\ &= \frac{k}{2} \geq \frac{v_{INT}(N)}{2}. \end{aligned}$$

Note the fact that, if $\sigma(b_j) < 1/2$, then $\sigma(b_{j-1}) > 1/2$ and $\sigma(b_{j+1}) > 1/2$. So if there are more than 1 bin with the total size of the packed items less than $1/2$, by similar proof we can also get the same conclusion.

To show $1/2$ is a tight bound on condition $k \geq 2$, consider an instance of 2 bins, and 4 items with sizes $\{\frac{1}{2}, \frac{1}{2}+\epsilon, 1, 1\}$. It is easy to see $NF(N) = 1+\epsilon, v_{INT}(N) = 2$. If $1/2$ is not a tight bound for NF, i.e. $\exists\ \delta > 0$, s.t. $NF(N) \geq (\frac{1}{2} + \delta)v_{INT}(N)$. Thus in this particular instance,

$$NF(I) = 1 + \epsilon \geq (\frac{1}{2} + \delta) \cdot 2 = 1 + 2\delta.$$

Let $\epsilon = \delta$, we have $\delta \geq 2\delta$ and this is a contradiction. This means no matter how small the δ is, we can always construct a counterexample such that $NF(I) < (\frac{1}{2} + \delta)v_{INT}(N)$. Hence $1/2$ is a tight bound, that is to say $R_{NF}^{\infty} = 2$. □

Remark: NF may approximate arbitrarily bad when $k = 1$. To show this, we give 2 items with sizes $\epsilon, 1$, so we have $NF(N) = \epsilon$, $v_{INT}(N) = 1$, which is to say, $NF(N) \geq \epsilon \cdot v_{INT}(N)$. The approximation ratio goes to 0 when we let ϵ be sufficiently small.

Theorem 5.2.2. $R_{FF}^{\infty} = 1/2$.

Proof. Assume $k \geq 2$, by the same proof with Theorem 5.2.1, it is easy to see $R_{FF}^{\infty} \geq 1/2$. We give an example to show $1/2$ is a tight bound of FF and the proof is also similar to that of the Next-Fit in Theorem 5.2.1. Consider a bin packing game with 2 bins, and 6 items with sizes $\{\frac{1}{2} + \epsilon, \frac{1}{2} + \epsilon, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}\}$. □

Theorem 5.2.3. $R_{FFD} = 1/2$.

Proof. We know FFD performs no worse than NF, so if $k \geq 2$, $R_{FFD} \geq 1/2$. We only need to show $FFD(N) \geq 1/2$ when $k = 1$. Since all items are sorted in decreasing order by size, if $\sum_{i=1}^{n} a_i \geq 1$, then at least $1/2$ of bins should be filled;

if $\sum_{i=1}^{n} a_i < 1$, then $FF(N) = v_{INT}(N)$, so we are done. Consider an instance of the bin packing game of 2 bins, and 6 items with sizes $\{\frac{1}{2}+\epsilon, \frac{1}{2}+\epsilon, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}\}$. By similar proof to Theorem 5.2.1, it is easy to see 1/2 is also a tight bound, thus we have $R_{FFD} = 1/2$. □

Based on above analysis, we found NF, FF and FFD perform no better than a 1/2 approximation ratio. However, for the classical bin packing problem (minimization problem), W. F. de la Vega and G. S. Lueker [4] showed the existence of a $(1+\epsilon)$-approximation algorithm, i.e., $A(I) \leq (1+\epsilon)OPT(I)$. This encourages us to think of the $(1-\epsilon)$-approximation algorithm for computing v_{INT}.

5.2.3 $(1-\epsilon)$-Approximation algorithms

The bin packing game we have mentioned in formal chapters are all in *standard form*, i.e. item sizes are in $(0, 1]$ and the capacity of each bin is 1. Now we consider the *integer form* of the bin packing game, namely, item sizes only take positive integer values, which do not exceed the capacity of each bin B. We can easily see that the two different forms of the bin packing game are indeed equivalent, by dividing the bin capacity to each item and let the capacity of the bin to be 1. Under the integer form, a natural way of reduce the hardness of this problem is to restrict the largest number to be bounded by a constant C, i.e. $B \leq C$. Let $\epsilon = 1/C$, then in the standard form, it is identical to assume item sizes to be greater than or equal to ϵ.

In this way, we are able to make our problems easier, and at first we try to solve our problem in this special case.

Lemma 5.2.4. *Given $\epsilon > 0, B > 0$ and the bin packing game of all item sizes greater than ϵ, and $k < B$, then there exists a polynomial algorithm for computing $v_{INT}(N)$.*

Proof. Since all item sizes are larger than ϵ, there are at most $\lfloor 1/\epsilon \rfloor$ number of items in each bin. Let $M = \lfloor 1/\epsilon \rfloor$, so the number of different feasible sets is bounded by

$$\binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{M} = p(n).$$

It is easy to see $p(n)$ is polynomial in n. To obtain $v_{INT}(N)$, we need to choose k feasible sets, such that the sum is maximal. The number of choices is polynomially bounded by

$$\max\left\{\binom{2B}{B}, \binom{p(n)}{B}\right\}.$$

Enumerating these feasible sets, we can get $v_{INT}(N)$ in polynomial time. □

In this lemma, we can not get rid of the requirement $k < B$. Reviewing the proof of Theorem 3.3.2 that the 3-PARTITION can be reduced to the bin packing game, which has all item sizes greater than 1/4. Therefore, the problem of computing $v_{INT}(N)$ is also strongly $\mathcal{NP}$-hard even when all item sizes greater than a given $\epsilon(\leq 1/4)$.

Next under the assumption $k < B$, we present the $(1-\epsilon)$-approximation algorithm of computing $v_{INT}(N)$.

Theorem 5.2.5. *Given $B > 0$ and the bin packing game with number of bins $k < B$, there exists a $(1-\epsilon)$-approximation algorithm for computing $v_{INT}(N)$.*

Proof. For given $\epsilon > 0$, let $I_1 = \{i \mid a_i \geq \epsilon, i \in I\}$, $I_2 = I \backslash I_1$, where we assume I be the item set. First we try to solve v_{INT} according to I_1. By Lemma 5.2.4, this is polynomially solvable. Next we want to add the items of I_2 into these bins as much as we can, on condition that the total size of the packed items in each bin does not exceed 1. Applying algorithm FF to I_2, if all items are packed into the k bins, we have $A(N) = v_{INT}(N)$; if not, we suppose $b_1, b_2, \cdots, b_k$ be the feasible packing generated by FF, where b_j is the set of the items in the j-th bin and let $\sigma(b_j) = \sum_{i \in b_j} a_i$ be the value of the j-th bin. By the process of FF we see for all j, we have $\sigma(b_j) \geq 1 - \epsilon$. This means

$$A(N) \geq (1-\epsilon)k \geq (1-\epsilon)v_{INT}(N).$$

□

5.3 Approximate v_{FRA}

To approximate v_{FRA}, we use the same way as we have shown in Section 5.2.3. First we only consider the restricted version of the bin packing game, and further to prove it is polynomially solvable; then for the general case of the bin packing game, we split the item set into 2 parts–one part is polynomially solvable and in another part, we expect an algorithm to well approximate the optimal value.

Lemma 5.3.1. *Given $0 < \epsilon \leq 1$ and the fractional bin packing game of all item sizes greater than ϵ, there exists a polynomial algorithm for computing $v_{FRA}(N)$.*

Proof. If $k \geq n$, we just need to pack each item into the first n bins. So we can always assume $k < n$. Since sizes of all items are larger than ϵ, there are at most $\lfloor 1/\epsilon \rfloor$ items in each bin. Let $M = \lfloor 1/\epsilon \rfloor$, so the number of feasible sets is bounded by

$$\binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{M} = p(n).$$

It is easy to see $p(n)$ is polynomial in n. Now we look at the linear program of the fractional bin packing game,

$$\begin{aligned} \max\ & a^T By, \\ \text{s.t. } & By \le e_n^T, \\ & e_F^T y \le k, \\ & y \in [0,1]^F, \end{aligned}$$

where F is the number of feasible sets and $B \in \{0,1\}^{n\times F}$. The input consists of the number of bins k, the size vector a and the feasible matrix B. Denote by $|L|$ the encoding length of the input, then

$$|L| \in O(n + n \times F) \in O(np(n)),$$

which is polynomial in n. We know the linear programming can be solved polynomially if the input length is polynomially bounded (eg. Karmarkar's Alogrithm, [13]), that is to say, the polynomial algorithm for computing $v_{FRA}(N)$ exists. □

Theorem 5.3.2. $\forall\ 0 < \epsilon \le 1$, *there exists a* $(1-\epsilon)$*-approximation algorithm for computing* $v_{FRA}(N)$.

Proof. Given $\epsilon > 0$, let $I_1 = \{i \mid a_i \ge \epsilon, i \in I\}$, $I_2 = I \backslash I_1$, where we assume I be the item set. First we try to solve v_{FRA} according to I_1. By Lemma 5.3.1, it is polynomially solvable. Next we want to improve the result by adding items from I_2 to the current solution as much as we can. Refer to the LP model of the fractional bin packing game (4.2), suppose $y_1, y_2, \cdots, y_r \in (0,1]$ be the optimal solution according to I_1 and $f_1, f_2, \cdots, f_r$ be the associated feasible sets. As is shown in the proof of Theorem 3.3.3, the number of feasible sets in the optimal solution is at most $n+1$, so r is polynomially bounded. Since items fractionally appear in these feasible sets, we use the following way to add items of I_2 to them (bins) as much as we can.

Let $I_2 = \{i_1, i_2, \cdots, i_t\}$ with sizes $\{a_{i_1}, a_{i_2}, \cdots, a_{i_t}\}$ and let $y_{i_j} = 1$ for all $1 \le j \le t$, which is used for recording the usable fractional parts of each item, so the initial value is 1. Let $\sigma(f_j) = \sum_{i\in f_j} a_i$ be the value of the j-th feasible set, $1 \le j \le r$.

Algorithm-Fractional First Fit (FFF)

```
FOR all items a_{i_j}, 1 ≤ j ≤ t DO
  WHILE y_{i_j} ≠ 0 DO
    Compute the smallest index l, s.t. σ(f_l) + a_{i_j} ≤ 1
    IF y_{i_j} − y_l ≥ 0
      THEN f_l := f_l ∪ {i_j}, y_{i_j} := y_{i_j} − y_l
    ELSE
```

```
      Let y_ij := 0
      Break f_l into 2 parts in following way:
        f_l1 := f_l ∪ {i_j}, y_l1 := y_ij
        f_l2 := f_l, y_l2 := y_l − y_ij
      Delete f_l
    ENDIF
  ENDWHILE
ENDFOR
```

We further give some explanations for Fractional First Fit (FFF). FFF can be viewed as an extension of First Fit (FF) algorithm, which packs fractional parts of items into bins. For instance, to improve the current solution, we can first apply FF to I_2, if item a_{i_j} can be packed into f_j, which has a fractional index y_j, then we put y_j of item i_j into bin f_j and $1 - y_j$ of item i_j into other possible bins. Since there are t iterations, and in each iteration we compute the smallest index l, which can be done in $O(n^2)$, so the running time of FFF $\in O(t \cdot O(n^2)) \in O(n^3)$.

Therefore, using items of I_2 by algorithm FFF the current solution can be improved. If FFF fully used all items, then $A(N) = v_{INT}(N)$; if not, we know from the process of FFF that, for all $1 \leq j \leq k$, $\sigma(f_j) \geq 1 - \epsilon$ holds and this implies $A(N) \geq (1 - \epsilon)k \geq (1 - \epsilon)v_{INT}(N)$. □

5.4 The Minimal Tax Rate

As far as we have discussed, we have fortunately found the $(1 - \epsilon)$-approximation algorithm for v_{INT} (only instances with $k < B$) and v_{FRA}, now we are able to approximate the minimal tax rate ϵ_{min}.

Theorem 5.4.1. *Consider a bin packing game $\langle N, v_{INT}\rangle$, where N consists of k bins and item set I and let $\epsilon_{min}(N)$ be the minimal tax rate for N. For fixed $\delta > 0$, there is a polynomial algorithm A to approximate $\epsilon_{min}(N)$, s.t. $|\epsilon_{A(N)} - \epsilon_{min}(N)| \leq \delta$.*

Proof. Given $\delta > 0$, Faigle and Kern [6] showed that $C_\epsilon(v_{INT}) \neq \emptyset$ when $k \geq 48\lceil\epsilon^{-1}\rceil^5$. If $k \geq 48\lceil\delta^{-1}\rceil^5$, we can take $\epsilon_{A(N)} = \delta$. So we may assume $k \leq 48\lceil\delta^{-1}\rceil^5$. By theorem 5.2.5 and 5.3.2, there exists a $(1 - \delta)$-approximation algorithm for computing $v_{INT}(N)$ and $v_{FRA}(N)$ and the approximation values are denoted by $A_{INT}(N)$, $A_{FRA}(N)$ respectively. Let $\epsilon_{A(N)} = 1 - A_{INT}(N)/A_{FRA}(N)$, we get

$$v_{INT}(N) \geq A_{INT}(N) \geq (1 - \delta)v_{INT}(N),$$

$$v_{FRA}(N) \geq A_{FRA}(N) \geq (1 - \delta)v_{FRA}(N).$$

We have

$$\epsilon_{A(N)} = 1 - \frac{A_{INT}(N)}{A_{FRA}(N)} \leq 1 - \frac{(1-\delta)v_{INT}(N)}{v_{FRA}(N)} = \delta + (1-\delta)\epsilon_{min}(N) \leq \delta + \epsilon_{min}(N),$$

$$\epsilon_{A(N)} \geq 1 - \frac{v_{INT}(N)}{(1-\delta)v_{FRA}(N)} \geq \epsilon_{min}(N) - \frac{\delta}{1-\delta} \cdot \frac{v_{INT}(N)}{v_{FRA}(N)} \geq \epsilon_{min}(N) - \delta.$$

Combining the above two inequalities, we get $|\epsilon_{A(N)} - \epsilon_{min}(N)| \leq \delta$. □

Corollary 5.4.2. *Consider a bin packing game $\langle N, v_{INT}\rangle$, where N consists of k bins and item set I. For fixed $\delta > 0$, there is a polynomial algorithm to compute an ϵ^* with the properties*

1. *ϵ^*-core $C_{\epsilon^*}(v_{INT})$ is nonempty;*
2. *$\epsilon^* - \epsilon_{min}(N) \leq \delta$.*

Proof. Let $\gamma = \delta/2$, by Theorem 5.4.1 we know there exists a $(1-\gamma)$-approximation algorithm A to compute $\epsilon_{min}(N)$, and we denote by $\epsilon_{A(N)}$ the approximation result. So

$$|\epsilon_{A(N)} - \epsilon_{min}(N)| \leq \gamma.$$

Let $\epsilon^* = \epsilon_{A(N)} + \gamma$, then $0 \leq \epsilon^* - \epsilon_{min}(N) \leq 2\gamma = \delta$. □

Chapter 6

Applications and Discussions

6.1 Summary of Conclusions

The bin packing game is a specific example of the N-person cooperative game, which primarily concerns the formation of coalitions among players. Centered around the question of finding an core allocation vector for a given instance of the bin packing game, we started by introducing a generalized definition–the ϵ-core, then our problem becomes how to find the nonempty ϵ-core. Obviously, 1-core always exists for any instance of the bin packing game, so the question is in fact to find the minimal ϵ such that the ϵ-core is not empty.

However, it is not easy to see whether a problem has a nonempty ϵ-core. To see this, a straightforward idea is to formulate the linear program of this allocation problem (AP), then to solve (AP) is a way of finding the ϵ-core. Indeed, if the objective value equals to the earning of the grand coalition, then the core exists; otherwise there exists some ϵ such that the ϵ-core is not empty. In order to compute the ϵ, we studied the dual problem of (AP), which is the so called fractional bin packing game, and thereby giving a sufficient and necessary condition for the existence of the ϵ-core, i.e. the ϵ-core exists if and only if $\epsilon \geq (v_{FRA}(N) - v_{INT}(N))/v_{FRA}(N)$.

The next is to compute the minimal ϵ which ensures a nonempty ϵ-core and the main results of this report are also based on above analysis. Specifically speaking, we showed computing the value function of the bin packing game (v_{INT}) and the corresponding fractional bin packing game (v_{FRA}) is $\mathcal{NP}$-hard, and further conclude that given an allocation vector, to assert whether it lies in the ϵ-core is also $\mathcal{NP}$-hard. Moreover, we showed the problem of deciding whether an instance of the bin packing game has an empty core is $\mathcal{NP}$-hard.

Therefore, polynomial algorithms for computing v_{INT} and v_{FRA} do not exist unless $\mathcal{P} = \mathcal{NP}$. Faced by this situation, we have to choices: the first is to solve these $\mathcal{NP}$-hard problems exactly but non-efficiently (not in poly-time), to do this,

we introduced a matrix form of the linear program for the bin packing game and the fractional bin packing game, which is easier for calculations. Besides, we applied the column generation approach to the fractional bin packing game, and this method successfully solves large instances of fractional bin packing games; the second choice is to find the approximation algorithms that can well approximate the optimal value as well as the running time is polynomial. For the bin packing game, we considered a special case of it: the number of bins is bounded by a constant, and we proposed the $(1-\epsilon)$-approximation algorithm for it; while for any instance of the fractional bin packing game, we found the $(1-\epsilon)$-approximation algorithm.

Based on approximation algorithms of computing v_{INT} and v_{FRA} and given $\delta > 0$, we finally proposed an approximation algorithm for computing the minimal ϵ within the error δ.

6.2 Problems to be Solved

There are still some problems to be solved. Reviewing Conjecture 2.4.2 which is proposed by Woeginger (1995) [6], it still has no answer for general case of bin packing games. To verify its possibility of the correctness, we made a lot of computational test, and surprisingly found the trend that the more items and bins we have, the gap between v_{FRA} and v_{INT} becomes less. Besides, we even failed to find an instance of the bin packing game such that $v_{FRA}(N) - v_{INT}(N) \geq 1/4$. So our problem is

problem 1: is $v_{FRA}(N) - v_{INT}(N)$ bounded by a constant for all instances of the bin packing game?

For this problem we have an interesting observation. Consider the 3-PARTITION problem of $3t$ positive integers $b_1, b_2, \cdots, b_{3t}$, where $\sum_{i=1}^{3t} b_i = t \cdot B$ and $B/4 < b_i < B/2$, the question is: does there exist a partition $S_1, S_2, \cdots, S_t$ such that the sum of the numbers in each subset is equal?

As we have shown before, we construct an instance of the bin packing game by letting $k = t$ and $a_i = b_i/B$ for all $1 \leq i \leq 3t$, then $v_{INT}(N) \geq k$ if and only if 3-PARTITION has a solution. Since $a_i > 1/4$ for all i, by Lemma 5.3.1, $v_{FRA}(N)$ can be computed in polynomial time. Therefore, if $v_{FRA}(N) < k$, we certainly have $v_{INT}(N) < k$, which also implies 3-PARTITION do not have a solution. Conversely, there must exist some instance N such that $v_{FRA}(N) \geq N$ and $v_{INT}(N) < N$, otherwise we can polynomially solve 3-PARTITION and this leads to $\mathcal{P} = \mathcal{NP}$.

We assume $\mathcal{P} \neq \mathcal{NP}$ and let N be the instance of the bin packing game satisfying $v_{INT}(N) < k$ and $v_{FRA}(N) > k$, where N consists of k bins and n items. We construct another bin packing game N' by letting $k' = k$ and $a'_i = 1/3 + \epsilon(1/3 - a_i)$ for $1 \leq i \leq n$. It is easy to see that the two instances are in fact equivalent, since N' has the same feasible sets with N. In this case, at least one item (denoted by a_i)

is not packed into any bin in the optimal packing of N, so we have

$$v_{FRA}(N) - v_{INT}(N) = v_{FRA}(N') - v_{INT}(N') \geq \frac{1}{3} + \epsilon(\frac{1}{3} - a_i) \approx \frac{1}{3}.$$

So there exists an instance N such that the gap $v_{FRA}(N) - v_{INT}(N)$ is around $1/3$, however, an example of gap larger than $1/4$ is not known yet.

Moreover, in Theorem 3.5.1 we stated that deciding whether an instance of the bin packing game has an empty core is $\mathcal{NP}$-hard and we proved this by reduction from PARTITION. However, given $\epsilon > 0$ the same reduction can not be simply applied to the case of ϵ-core. So our question is to find a proper $\mathcal{NP}$-hard problem so that it can be reduced to the problem of deciding whether an instance of the bin packing game has an empty ϵ-core, namely to answer

problem 2: let $UB = inf\{\epsilon \mid$ *exits an instance of the bin packing game, s.t. the* ϵ*-core is empty*$\}$ *and given* $0 < \epsilon < UB$*, is the problem of deciding whether an instance of the bin packing game has an empty* ϵ*-core* $\mathcal{NP}$*-hard?*

6.3 Applications

In this section we show some practical problems which can be modeled as bin packing games. In applicable aspect of view, the application of our research on bin packing games is the motivation of why we study this subject and this can also be viewed as the application of game theory.

6.3.1 Packing Problems

The literal explanation of bin packing games is just the problem of packing items. Specifically speaking, given k bins and n items with sizes $a_1, a_2, \cdots, a_n$, a basic problem is to pack these items into the k bins maximizing the total size of the packed items. For example, an ocean shipping company distributes k ships on the shipping line from China to the Netherlands, and each ship has a capacity C, i.e. the total weight of the loaded cargos can not exceed C. Every month the k ships set sail from China and transport cargos to the Netherlands. Note that each item of cargos has weight $0 < a_i \leq C$, to maximize the profit for a single trip, so our problem is to maximize the loadings of the k ships.

It is clear that to solve this problem is exactly the same with computing the value function of a bin packing game. As a further remark, the value function of the bin packing game is in fact a special case of the multiple KNAPSACK problem: given k knapsacks, each has a capacity C and n items, each item has a value v_i and weight a_i, $1 \leq i \leq n$, the problem is to find a packing maximizing the total value of

the knapsacks on condition that the total weight of each knapsack does not exceed C. If we let $v_i = a_i$, for $1 \leq i \leq n$, then the multiple KNAPSACK problem becomes a question of the bin packing game.

6.3.2 Allocation Problems

The initial incentive of our research is to find the core allocation of the total earning of the grand coalition. Thus, allocation problems in real life also have much to do with bin packing games.

Suppose the orders of a software company are completed by work teams–each work team is divided into groups, and each group has a leader and the employees of the work team are distributed to the leaders. We consider a work team of k leaders and n employees, in order to finish a project, the leaders' job is to make the blue print of their work and to guide their subordinates, while the employees are the real persons to do the actual project of the work. Since employees have different individual abilities, we use the index $0 < a_i \leq 1$ as an evaluation of the abilities of each person, which means, exists some constant C, if the leader paid a_i energy to the employee i, then he can earn $a_i \cdot C$ euros for his company, otherwise he earns nothing. We assume a employee can not be guided by 2 leaders at the same time and the total energy of a leader is 1, then the sum of the energy the leader pays to his subordinates can not exceed 1. The first problem is to distribute these employees to the leaders maximizing the total profits of the company; the second question is to find a core allocation vector to each employee (including the leaders) such that every one is satisfied.

It is easy to see that this problem is a standard characterization of the bin packing games in a actual application. However, the core allocation does not exist for many instances of the bin packing game. A secondary solution is to tax the earning of each employee by a rate ϵ, therefore the second question becomes to find an ϵ-core allocation vector for the total work team. By using the algorithms we have introduced for computing the minimal tax rate, we are able to solve this practical problem.

Appendix A

Generate Feasible Matrix

Given a size vector $a = (a_1, a_2, \cdots, a_n)^T \in (0, 1]^n$, the associated feasible vector b_j is defined by $a^T b_j \leq 1$, $b_j \in \{0, 1\}^n$. The feasible matrix B consists of all feasible vectors, i.e.

$$B = (b_1\ b_2\ \cdots\ b_F), \text{ where } F \text{ is the number of different feasible vectors.}$$

Let $b_j = (b_{1j}, b_{2j}, \cdots, b_{nj})^T$, then the corresponding feasible set $f_j = \{i | b_{ij} = 1\}$. Obviously, every item itself can be a feasible set.

Given $a = (1/4, 1/2, 2/3)^T$, we take this example to illustrate the sketch of our algorithm. First we discard the capacity constraint $a^T b_j \leq 1$ and observe that the feasible matrix has the form

$$P_3 = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 \end{bmatrix}.$$

The matrix P_3 implies all combinations of items and satisfies that the number of items in each set is most 3. Similarly, we define the matrices P_1, P_2 by

$$P_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} p_1 & p_2 & p_3 \end{bmatrix},$$

$$P_2 = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} p_1 & p_2 & p_3 & p_4 & p_5 & p_6 \end{bmatrix}.$$

We see P_1 is a submatrix of P_2, while P_2 is also a submatrix of P_3 and note that the matrices P_1, P_2, P_3 only depend on the number n and the maximal number s $(1 \leq s \leq n)$ of items we choose in a set. Therefore, a natural idea is to recursively generate the matrix P_3.

Observe that, by knowing P_2, we only need to add columns that have 3 nonzero components, and the easiest way of realizing this purpose is to make use of the vectors which have 2 nonzero entries, i.e. p_4. In fact, we compute the largest index n_p such that $p_{n_p,4} = 1$, then one of the next positions $n_p + 1, n_p + 2, \cdots, n$ can take the value 1. Nevertheless, we can not use p_5, p_6 since $p_{35} = p_{36} = 1$, otherwise we have no free position to add new items. However, in order to know which column has 2 nonzeros, we define indexes r_1, r_2 to record its *start column number* and the *end column number*.

Now we are able to generate P_3, but it may contain non-feasible vectors. So the next is to consider the capacity constraint. In every recursive iteration, before adding a new vector q to the current matrix, we compute $a^T q$ and if it does not exceed 1, then we add q to our feasible matrix, otherwise throw it away. Our algorithm is described as below.

Algorithm-Generate Feasible Matrix

To recursively genrate matrices B_1, B_2 $\cdots$ B_n
In the s-th iteration
   Use B_{s-1} (columns from r_1 to r_2) to generate
     the new vector q which has s nonzero entires
   IF $a^T q \leq 1$
      THEN add q to B_{s-1}
   ELSE
      Discard q
   ENDIF
   Update r_1, r_2
END

The MATLAB code of the above algorithm is pasted as follows.

```
function [B,r1,r2]=GetMatrix(a,n,s)
if s>1
    %get the matrix with the maximal number of items <= s-1
    [B,r1,r2]=GetMatrix(a,n,s-1);
    r3=r2;%record the position of r2
    for i=r1:r2
        p=B(:,i);%take the i-th column of B
        %compute the next position, which may enable a new feasible vector
        np=n+1;
        while p(np-1)==0
            np=np-1;
        end
        for j=np:n
```

```
            q=p;%to create a new feasible vector
            q(j)=1;%then add another item to to the position j
            %if the value of the new feasible vector does not exceed 1
            if a'*q<=1
               B=[B,q];%add this new feasible vector to B
               r2=r2+1;%set the end column number
            end
        end
    end
    %if we found the feasible vector which has s nonzero components
    if r2>r3
        r1=r3+1;%then set the start column number to r1
    else
        return;
    end
else
   B=eye(n);
   r1=1;
   r2=n;
end
end
```

Bibliography

[1] R. Branzei, D. Dimitrov, and S. Tijs. *Models in Cooperative Game Theory*. Springer-Verlag, 2005.

[2] E. G. Coffman, J. M. R. Garey, and D. S. Johnson. Approximation algorihms for bin packing: A survey. In D. Hochbaum, editor, *Approximation Algorithms for NP-Hard Problems*, pages 46–93. PWS Publishing, Boston, 1996.

[3] S. A. Cook. The complexity of theorem-proving procedures. In *Annual ACM Symposium on Theory of Computing*, pages 151–158, New York, USA, 1971. Association for Computing Machinery, ACM.

[4] W. F. de la vega and G. S. Lueker. Bin packing can be solved within $1+\epsilon$ in linear time. *Combinatorica*, 1(4):349–355, 1981.

[5] U. Faigle and W. Kern. On some approximately balanced combinatorial cooperative games. *Mathods and Models of Operation Research*, 38:141–152, 1993.

[6] U. Faigle and W. Kern. Approximate core allocation for binpacking games. *SIAM J. Discrete Math*, 11(3), August 1998.

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

[8] P. C. Gilmore and R. E. Gomory. A linear programming approach to the cutting stock problem-part ii. *Operations Research*, 11(6):863–888, 1963.

[9] H. T. Jongen, K. Meer, and E. Triesch. *Optimization Thoery*, chapter 26. Springer, London, 2004.

[10] J. Kuipers. Bin packing games. *Mathematical Methods of Operations Research*, 47:499–510, 1998.

[11] Z. Liu. Complexity of core allocation for the bin packing game. *Operations Research Letters*, 37:225–229, 2009.

[12] J. V. Neumann and O. Morgenstern. *Theory of Games and Economic Behavior*. Princeton University, Princeton, 1947.

[13] N.Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395, 1984.

[14] L. S. Shapley. On balanced sets and cores. *Naval Res. Logistics Quart*, 14:453–460, 1967.

[15] L. S. Shapley. Cores and convex games. *Internat. J. Game Theory*, 1:1–26, 1971.

[16] G. J. Woeginger. On the rate of taxation in a cooperative bin packing game. *Mathematical Methods of Operations Research*, 42:313–324, 1995.

Index

Acknowledgements

I could not have completed all my courses and written this master thesis without help from many people. I must begin by thanking all members of the admissions committee of Industrial Engineering and Operation Research track in Applied Mathematics because they offer me this precious chance of being a master student and the chance of enjoying my life in the Netherlands as well as in the university of Twente. Besides, I am grateful to the staff of our group–Discrete Mathematics and Mathematical Programming, they arranged my emigration and the settlement and provided me with perfect conditions for my study.

Working along this master thesis, it was my supervisor Walter Kern who kept guiding me, from the choices of my topic to the proof of each theorem, and even my mistakes of the English grammar. The expertise suggestions, technical advice and the motivational encouragement I received from him significantly improved my interpretation to the academic research. Besides, he is always there for me, and I appreciate him very much of answering my questions without any hesitate.

In particular, I have to mention the scene on the first day of my arrival in Enschede, I still remember the face of the very kind man Kees Hoede who picked me up at the train station. Though the departure of him brought me into a deep sorrow, his smile and kindness are remembered by me and so many thanks to him for his help on both of my study and my life, which even can not be exactly expressed by any word.

Also, I would express my thanks to Georg Still who is always prepared for helping others. The assistance I received from him as well as his friendly smile and fatherly love he showed to me will never go out of my memories. I would say I am so lucky to make acquaintance with him. Additionally, many thanks to the chair Prof. Marc Uetz, the secretary Dini Heres-Ticheler and other colleagues from DMMP for their help and especially thank them for holding so many interesting activities in our group.

I am also thankful to my friends in Enschede: Jiwu Lu, Qiang Tang & Shenglan Hu, Zheng Gong, Yanting Chen, Jichang Wu, Hongrui Zhang, Hongxi Guo, Kamman, Mingshang Yu, Xiao Ma, Lingfeng Xie & Danni Wang and Dongshuang Hou.

The unforgettable memories with all of you are my big treasure.

Most important of all, I am deeply grateful to my parents, grand parents, girlfriend and other relatives who were care about me. The selflessly support and encouragement I received from them largely reduced my pressure and the feeling of loneness and moreover promoted my study.