ON THE SELECTION AND ASSIGNMENT PROBLEM WITH MINIMUM QUANTITY COMMITMENTS

XU ZHOU

NATIONAL UNIVERSITY OF SINGAPORE

2003

ON THE SELECTION AND ASSIGNMENT PROBLEM WITH MINIMUM QUANTITY COMMITMENTS

XU ZHOU

(B.E. TSINGHUA UNIVERSITY)

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF SCIENCE

DEPARMENT OF COMPUTER SCIENCE

NATINAL UNIVERSITY OF SINGAPORE

2003

Name: XU Zhou
Degree: Master of Science
Dept: Computer Science
Thesis Title: On the Selection and Assignment Problem with Minimum Quantity Commitments

Abstract

The thesis studies a new constraint, named minimum quantity commitments, for the selection and assignment problem. It is motivated by the Royal Philips Electronics and a special stipulation of the US Federal Maritime Commission. Our work is expected to initiate the research on this novel constraint for the selection and assignment problem. We first classify all the special cases and prove their computational complexity, and then, focus on a basic problem by analyzing its mixed integer programming model and proposing two heuristics algorithms to solve it. On top of the theoretical analysis, we also implement the different methods, and provide extensive experimentations to measure their performance.

Keywords: Approximation algorithm; Computational complexity; Heuristic algorithm; Minimum quantity commitment; Mixed integer programming; Selection and assignment problem

Acknowledgements

Many people have contributed to my education and research through their guidance and support during my graduate school time. I especially want to thank my supervisor, Associate Professor Loe Kia Fock, who helped and encouraged me to overcome difficulties throughout my work. Without him, the thesis can not be completed successfully.

I acknowledge Associate Professor Andrew Lim, who helped to suggest this interesting topic of my research.

I am grateful to my friends, Dr. Wang Fan who checked the proofs carefully, Mr. Zhang Xing Wen who assisted me to run the extensive experiments, and many other friends, including Ms. Chen Ping, Mr. Huang Gaofeng, Mr. Xiao Fei, Mr. Zhu Yi, Mr. Ong Kian Win, Mr. Guo Yunsong, Mr. Zhang Kaicheng, Mr. Zhu Wenbin, Mr. Tan Jingsong, Ms. Wu Xiaotao, Mr. Li Shuaichen, Mr. Chong Ket Fah, Mr. Frances Ng Hoong Kee, Ms. Wang Fei, and etc., for their supports and encouragements.

Last but not least, I especially thank my family and my girl friend, for their loves, cares and understandings through my life.

Contents

List of Figures, Tables and Algorithms

List of Figures

List of Tables

List of Algorithms

Summary

We study a new constraint, named minimum quantity commitments, for the selection and assignment problem. It is motivated by Royal Philips Electronics, who needs to select and assign shipping agents to satisfy its shipping needs with a minimal total costs. Because the US Federal Maritime Commission stipulates that any company sending goods to US via any shipping agent must commit to a minimum quantity for all its inbound goods to the US, Philips has to assign enough containers to every shipping agent engaged to transport cargoes to US.

This thesis is expected to initiate the research on this new practical constraint for the selection and assignment problem.

various special cases with this new constraint have been classified, and their computational complexity results are all proved. For each case considered, either an efficient algorithm or a proof to show that such an algorithm unlikely exists is given. The complexity result tells that the new minimum quantity commitment constraint brings difficulties to the selection and assignment problem, because finding optimal solutions turns out to be $\mathcal{NP}$-hard for most special cases. In contrast, when this new constraint is relaxed, some cases can be efficiently solved.

To study the new constraint further, the author works on a basic special case. This basic problem can be formulated as a mixed integer programming model, where a strong inequality is proved to be its facet under mild conditions. Experiments shows that this facet improves the performance of the mixed integer programming model significantly, such that an exhaustive branch and cut algorithm solves small and medium problem instances well.

To solve large problem instances practically, two heuristic algorithms are invented. One is a linear programming rounding method, which behaves well in experiments. The other is a greedy scheme, which guarantees a non-constant approximation ratio in theory under some circumstances.

In addition, extensive experiments have been made to measure both the performance of the mixed integer programming formulation, and the performance of the two heuristic algorithms.

Chapter 1

Introduction

1.1 Overview

A brief introduction is given in this chapter. Section 1.2 explains the motivation of our study on a new constraint, named Minimum Quantity Commitment (MQC), for the selection and assignment problem. Although the study of this new MQC constraint is scarce in the literature, other various selection and problems are extensively studied. We survey some related problems in Section 1.3, and epitomize their common solving techniques adopted before. They suggest us some idea of how to deal with the new MQC constraint. After the survey, we summarize our research contributions and depict the thesis outline in Section 1.4.

1.2 Motivations

Traditional selection and assignment problems analyze how to select and assign the suppliers to feed the demanders within minimum total costs. The maximum capacities and the fixed selection costs are two major restrictions imposed on the suppliers. However, in practice, the selected suppliers also need to satisfy the constraint of certain minimum capacities. Usually, this takes the form of a commitment by forcing each selected supplier to provide a certain minimum quantity of demands.

Restricted to the minimum quantity commitment, we consider a selection and assign-

ment problem, motivated by the requirement of the Royal Philips Electronics Company. The Philips Electronics is one of the largest electronics companies in the world. Because of its worldwide presence in sourcing of raw materials, manufacturing and sales, Philips has a large number of shipping needs from one city to another all over the world. To handle its shipping needs, Philips conducts an annual tendering exercise in which it informs various shipping agents of its shipping needs in terms of original-destination transportation requests and a container demand for each transportation request. These transportation requests can be serviced by a selected subset of the shipping agents where each agent will bid a transportation cost for shipping a container in every transportation request. In order for Philips to minimize the number of agents that it has to deal with and negotiate, each agent has an associated fixed selection costs. To reduce its reliance on a particular agent, Philips sets a maximum capacity for each agent.

In additional to these traditional restrictions, Philips has to consider a new one imposed on the selection and assignment, because the US Federal Maritime Commission stipulates that the total quantity of containers shipped to the cities in US by an agent for a company must either be zero or be at least as large as a minimum quantity, which is often denoted as Minimum Quantity Commitment (MQC). This rule or constraint may also hold for other regions, each having its particular minimum quantity.

Accordingly, the selection and assignment problem for Royal Philips Electronics is to select shipping agents and assign them the shipping containers in each transportation request, such that the total cost, including the transportation costs and agents' selection costs, is minimized subject to the constraints of satisfying the transportation demands, the maximum handling capacity of each agent, and the important MQC constraint. This explains the practical reasons for introducing the new MQC constraint into the selection and assignment problem, and illustrates the motivations of the study presented in this thesis.

1.3 Related Problems and Techniques

The MQC constraint has been studied in the analysis of supply contracts [6, 24], but never appeared in literature on selection and assignment problems.

However, without considering the MQC constraint, two basic selection and assignment problems have been extensively studied. One is the transportation problem in which only the maximum capacities of agents and the shipping demands need to be satisfied. This simple assignment problem can be solved in polynomial time by the minimum cost flow algorithm [3]. The other one is the facility location problem which can be regarded as an extension of the transportation problem by introducing a fixed selection cost for each agent. Since selections of agents impose costs, the problem turns harder. Unfortunately, the facility location problem unlikely has an efficient algorithm and is proved to be $\mathcal{NP}$-hard, even for the uncapacitated case where the maximum capacities of suppliers are relaxed to be infinity [10].

Other variants have also been studied. For example, the p-median problem [14] is a variant of the transportation problem by restricting the number of selected agents, and the capacitated concentrator location problem [21] is a variant of the facility location problem by stipulating that each shipping request must be served by exactly one agent. For most of these variants, finding an optimal solution is intractably $\mathcal{NP}$-hard. Even for some of them, like the capacitated concentrator location problem, finding only a feasible solution is intractable.

To solve these traditional selection and assignment problems practically, different optimization techniques have been invented and applied in literature. They can be classified into the three categories: search algorithms, approximation algorithms and heuristics.

Search Algorithms

Most selection and assignment problems, like facility location problem and the p-median problem, can be formulated as an integer or mixed integer programming model. As a result, a branch and cut search algorithm [26] can be applied to find the exact optimal solutions for small scale problem instances.

In order to prune invalid branches when searching a solution with a minimum total cost, we need to obtain a lower bound by solving a relaxed linear programming problem and an upper bound by transforming the relaxed infeasible solution to be feasible. The performance of the pruning depends on the tightness of the lower bound and the upper bound, both of which can be improved by adding some strong valid inequalities or facets [26] to the integer or mixed integer programming model.

Aardal and van Hoesel [1][2] illustrated an outline of the branch and cut algorithm and discussed several ways of finding strong valid inequalities for mixed integer programming problems. K. Aardal and Wolsey [17] analyzed a number of valid inequalities and facets for the facility location problem. Moreover, de Farias Jr [11] incorporated a family of non-trivial facets in a branch and cut scheme for p-median problems, and produced optimal solutions even for some large scale instances. They demonstrated the importance of good formulations for integer or mixed integer programming problems.

However, search algorithms can solve only some special cases. As we known, most selection and assignment problems are $\mathcal{NP}$-hard, so that they unlikely have efficient polynomial-time algorithms to generate exact optimal solutions in theory. Fortunately, approximation algorithms might exist for them.

Approximation Algorithms

An approximation algorithm guarantees that the solution it generates is near the optimal. Theoretically speaking, we use the approximation ratio a to indicate its distance from the optimal and define an a-approximation algorithm as a polynomial-time algorithm that computes a solution with cost at most a times the optimum.

The basic techniques for inventing good approximation algorithms for selection and assignment problems are greedy method, linear programming analysis and local search schemes. For instance, Hochbaum [15] gave a simple greedy algorithm for uncapacitated facility location problem in 1982. Its approximation ratio is $O(logn)$ where n is the number of agents. For the same uncapacitated facility location problem, if we restrict the transportation cost to be a distance matrix or metric, (where the triangle inequality are satisfied), the first constant ratio approximation algorithm was obtained through a linear

programming analysis by Shmoys, Tardos and Aardal [25] in 1997, and till now, its best approximation ratio is 1.61 achieved by Jain, Mahdian and Saberi in 2002 [16]. Under the same restriction of the transportation cost, the p-median problem has a local search algorithm which guarantees 5.0 approximation ratio [5]. Furthermore, Korupolu, Plaxton and Rajaraman summarized approximation results of various selection and assignment problems in [19].

Heuristics

Differently from the approximation algorithms, heuristics methods guarantee neither its approximation ratio nor its running time. However, good heuristics can generate near optimal solutions on most practical instances of a problem. it may even outperform the best approximation algorithms in practice.

Most approximation techniques, like greedy methods, linear programming analysis and local search schemes, can also be adopted as heuristics for selection and assignment problems. Furthermore, some advanced local search techniques, like genetic algorithm [22], tabu search [13], and simulated annealing [18] have been extensively applied as well. For example, both tabu search [23] and genetic algorithms [7] have been applied for solving p-median problem, and they exhibit good performance in practical experiments. So does the tabu search for the uncapacitated facility location problem [4].

1.4 Contributions

Our thesis presents the first research work on the MQC constraint for selection and assignment problems. To initiate the study, we begin with the fundamental notations and formulations in Chapter 2. Under the background of the original bidding selection and assignment problem for Philips, various special cases are classified in terms of three aspects of restrictions, which are maximum capacities of agents, fixed selection costs of agents, and the number of regions holding MQC constraints. Among these special cases, their reduction relations establish a hierarchy of difficulties for solving them.

Our main study seeks the answers to the following two questions for solving the

selection and assignment problems with MQC constraints.

1. What is its computational complexity?

2. Can we give efficient algorithms?

For computational complexity, we are interested in how the new MQC constraint increases the difficulty of solving the selection and assignment problem. Chapter 3 elaborates the computational complexity results for various special cases. Their reduction relations allow us to study only some of them, from which a whole mapping of computational complexity can be derived for all. For each case considered, we provide either an efficient algorithm to solve it, or a proof that such an algorithm is unlikely to exist. Our result shows that the new MQC constraint significantly increases the difficulty of solving the selection and assignment problems. Under the new MQC constraint, finding an optimal solution turns out to be intractable for most special cases. And that, for some special cases, finding only a feasible solution is intractable as well. In contrast, without considering the new MQC constraint, feasible solutions are easily found for most cases, and so are optimal solutions except for a few cases.

For the efficient algorithm, although the computational complexity results have shown the selection and assignment problem with MQC constraints to be theoretically intractable, we still interest in how to solve it practically. To simplify the problem, we focus on a basic special case where only the new MQC constraints and the shipping demands are considered. In Chapter 4, three optimization techniques are applied.

Firstly, a branch and cut search scheme can be adopted, since the basic special case can be formulated as a mixed integer programming model. To improve the performance of the search algorithm, we utilize a strong valid inequality which is proved to be a facet of the model in mild conditions. Therefore, the search scheme can generate exact optimal solutions for all small scale instances and most medium ones.

Then, two heuristics algorithms are invented to solve those large problem instances practically. One is a linear programming rounding method, which behaves well in experiments. The other is a greedy method, which guarantees a tight non-constant approxi-

mation ratio in theory under some circumstances.

The extensive experiments have been made and reported in Chapter 5. To measure the performance of our mixed integer programming formulation, we compare the efficiency of branch and cut algorithms under the two different models. One is the mixed integer programming model with the facet we found, and the other is not. The results show that the facet significantly improves the search performance and makes the branch and cut scheme work well on medium scale instances. To measure the performance of the two heuristics algorithms, instances with different features have been tested. During the experiments, although the greedy method runs faster and guarantees a theoretical approximation ratio under some circumstances, the linear programming rounding scheme generates better solution and exhibits stabler behaviors.

Along with the experimental results, the work presented in this thesis may suggest further improvements, and is expected to initiate the research on the new MQC constraints for the selection and assignment problems.

1.5 Summary

This chapter introduced the background and depicted an outline of the thesis. The rest of the thesis will be organized as follows. Chapter 2 gives preliminaries, including the basic notations we adopted and the classification of various special cases we considered throughout this work. Chapter 3 presents the results of computational complexity, along with the proofs and algorithms needed. In Chapter 4, we focus on a basic selection and assignment problem with the new MQC constraint by applying different optimization approaches to solve it practically. Its related experiments are reported in Chapter 5. Finally, we give some conclusions and suggestions for future work in Chapter 6.

Chapter 2

Preliminary

2.1 Overview

This chapter gives the preliminaries for the thesis. We begin with some notations in Section 2.2, and formulate various constraints, especially the new MQC constraint, for selection and assignment problems. Afterwards, in Section 2.3, we classify various special cases for selection and assignment problems with the new MQC constraint. These special cases are specified in terms of three aspects of restrictions, and their reduction relations can be derived to help the study on computational complexity presented in next Chapter 3.

2.2 Notations and Formulations

Let $J = \{1, ..., n\}$ denote the set of transportation requests, each with the demand d_j for $j \in J$. Let $I = \{1, ..., m\}$ denote the set of shipping agents, each with the maximum capacity s_i and fixed selection cost f_i for $i \in I$. The cost bid by agent i for shipping a container in the transportation request j is $c_{i,j}$, for $i \in I$ and $j \in J$. Let $R = \{1, ..., w\}$ denote the set of regions holding MQC rules. The minimum quantity for region r is b_r, and the set of transportation requests with destinations in region r is $J_r \subseteq J$, for $r \in R$. It is reasonable to assume that J_p and J_q are disjoint for any two different regions p and q in R. We suppose throughout that d_j, s_i and b_r are non-negative integers, and that f_i

and $c_{i,j}$ are non-negative real numbers, for $i \in I$, $j \in J$ and $r \in R$.

For a solution to this selection and assignment problem, let $A \subseteq I$ denote the set of agents selected for shipping containers. Let a non-negative integer $z_{i,j}$ denote the quantity of containers assigned to the selected agent i in transportation request j, for $i \in A$ and $j \in J$. Therefore, the total cost of a solution is:

$$\sum_{i \in A} f_i + \sum_{i \in A} \sum_{j \in J} c_{i,j} z_{i,j}. \tag{2.1}$$

In order to feed the demands, all the containers of each transportation request $j \in J$ should be assigned to some selected agents, i.e.

$$\sum_{i \in A} z_{i,j} = d_j. \tag{2.2}$$

For restrictions of maximum capacities, the total quantity assigned to each selected agent $i \in A$ should not exceed s_i, i.e.

$$\sum_{j \in J} z_{i,j} \leq s_i. \tag{2.3}$$

Because of the MQC rules, the total containers assigned to each selected agent $i \in A$ to ship for those transportation requests with destinations in each region $r \in R$ need to be either zero or at least as large as b_r, i.e.

$$\sum_{j \in J_r} z_{i,j} = 0, \text{ or } \sum_{j \in J_r} z_{i,j} \quad \geq \quad b_r \tag{2.4}$$

Accordingly, the solution is feasible if and only if it satisfies the above three constraints (2.2)–(2.4), and is optimal if and only if it is a feasible solution with minimum total costs. Problems of finding an optimal (or feasible) solution is named optimization (or feasibility) problem.

To simplify and make consistent the notations throughout this thesis, let J_0 be the set of transportation requests unrestricted to the MQC rule, and fix its minimum quantity b_0 to be zero. Correspondingly, we let $U = R \cup \{0\}$ denote the universal set, while region 0 can be regarded as the complement of all the regions holding MQC rules. Moreover, we let $D = \sum_{j \in J} d_j$ denote the total demands of transportation requests and $S = \sum_{i \in I} s_i$

denote the total maximum capacity of shipping agents, because these two totals will be often used in this paper.

Figure 2.1 gives an instance of the selection and assignment problem with only one region (U.S.) holding MQC rules. It is easy to verify that the solution shown in Figure 2.1(d) is feasible.

Regions' Info. J_r	MQC b_r	Transportation Requests		
		j in J	Original & Destination	Demands d_j
J_0: destinations outside U.S.	0	1	from Amsterdam to Singapore	450
		2	from Beijing to Shanghai	400
		3	from New York to Singapore	500
		4	from Shanghai to Amsterdam	350
J_1: destinations inside U.S.	450	5	from Shanghai to New York	400
		6	from Singapore to San Francisco	500

(a) The Information of Transportation Requests

Agents i in I	Maximum Capacities s_i	Fixed Selection Costs f_i
1	800	2200
2	1500	1000
3	1500	1100

(b) The Information of Agents

Agents i in I	Bidding Costs $c_{i,j}$					
	j in J_0				j in J_1	
	1	2	3	4	5	6
1	150	100	150	100	150	200
2	100	50	100	100	100	300
3	150	150	200	150	150	150

(c) The Information of Bidding Costs

Selected Agents i in A	Assignments $z_{i,j}$						$\sum_{j\in J_1} z_{i,j}$	$\sum_{j\in J} z_{i,j}$
	j in J_0				j in J_1			
	1	2	3	4	5	6		
2	0	400	500	150	400	50	450	1500
3	450	0	0	200	0	450	450	1100
$\sum_{i\in S} z_{i,j}$	450	400	500	350	400	500		

(d) A Feasible Solution

Figure 2.1: An Instance of the Bid Selection and Assignment Problem

2.3 Classification of Special Cases

We will discuss various special cases of the selection and assignment problem with MQC constraints in this thesis. These cases can be classified in terms of a four-field notation $Obj(\alpha, \beta, \gamma)$, where Obj defines the objective (optimization or feasibility) considered, α indicates the number of regions holding MQC rules, β describes the agents' maximum capacities, and γ describes the agents' fixed selection costs.

The reasons for studying feasibility problem include two aspects. On one hand, some further complexity results of the optimization problem can be derived directly from that of the feasibility problem, because usually the former can be reduced from the latter. On the other hand, experience suggests that for those problems with difficulties of finding feasible solutions, to find their optimal solutions is much harder than for those problems with feasible solutions in hand. So the feasibility is worth studying.

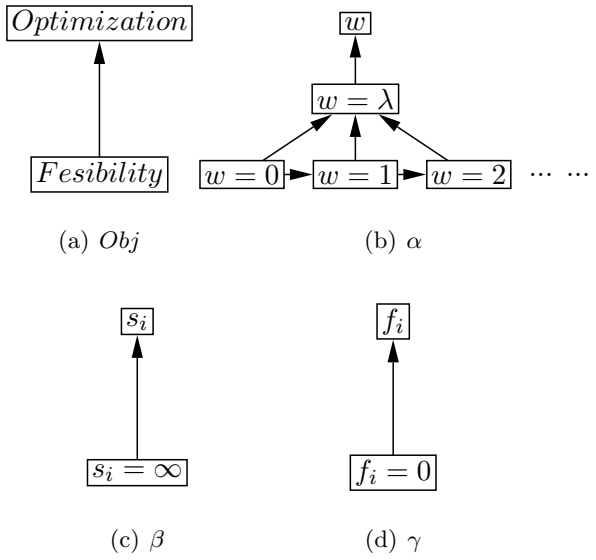
In the rest of this section, we will illustrate the specifications of the four fields in our notation, along with their reduction relations shown in Figure 2.2. A reduction from one special case to the other indicates that solving the latter case implies solving the former one [12][8]. Based on this, only a part of special cases need to be studied for their complexity, while others can be derived by the reduction relations. This helps the study of their computational complexity presented in Chapter 3.

Firstly, the field Obj defines the objective considered. Under Obj, we may have:

- *Optimization*: the objective is to find an optimal solution;
- $Feasibility$: the objective is relaxed to find only a feasible solution.

It is easy to see that if we have an efficient way to find an optimal solution, we can find a feasible solution in the same way, because any optimal solution must be feasible. In other words, the specification $Feasibility$ can be directly reduced to the *Optimization*, as shown in Figure 2.2(a).

Secondly, the field α indicates the number of regions holding MQC rules. Under α, we may have:



(a) *Obj* (b) α

(c) β (d) γ

Figure 2.2: The Reduction Relations

- w: the number of regions holding MQC rules is arbitrary, specified as part of the problem input;
- $w = \lambda$: the number or regions holding MQC rules is a constant, excluded from the problem input;
- $w = 0, 1, 2$, or etc.: the number of regions holding MQC rules is equal to the given integer. For example, if α is equal to $w = 1$, only one region holding the MQC rule.

Figure 2.2(b) shows the reduction relations for the field α. The reduction from the $w = \lambda$ to the w is straightforward by specifying the latter w to be λ. Similarly, we can reduce the $w = 0, 1, 2, ...$ to the $w = \lambda$ by specifying λ to be the given integer of the former case. Furthermore, to reduce the $w = 0$ to the $w = 1$, we set the R_1 to be empty in the latter case. This can be generalized to reduce the $w = k$ to the $w = k + 1$ for any non-negative integer k by setting R_{k+1} to be empty.

Thirdly, we consider the specification of the field β that describes the maximum capacities of agents. The field β may equal to:

- s_i: the maximum capacities of agents are arbitrary;

- $s_i = \infty$: the maximum capacities of agents are all infinity, i.e. the maximum capacity restriction (2.3) is relaxed.

As shown in Figure 2.2(c), the specification $s_i = \infty$ can be reduced to the s_i, because we can set the latter s_i to be sufficiently large, say at least as large as the total demands D, so that the maximum capacity restriction (2.3) can be relaxed in the latter case.

Lastly, we specify the field γ that describes the fixed selection costs of agents. Under γ, we may have:

- f_i: the fixed selection costs of agents are arbitrary;
- $f_i = 0$: the fixed selection costs of agents are all zero, i.e. the total cost of a solution is relaxed from (2.1) to

$$\sum_{i \in A} \sum_{j \in J} c_{i,j} z_{i,j}. \tag{2.5}$$

Clearly, the specification $f_i = 0$ can be directly reduced to the f_i by setting the latter f_i to be zero, as shown in Figure 2.2(d).

2.4 Summary

The basic notations introduced in this chapter will be used throughout the thesis. Although there are many constraints for selection and assignment problems in literature, we only consider and formulate some of them, i.e. those related to the Philips' bidding problem, because that is the motivation of the study on the new MQC constraints for selection and assignment problem. Besides these, our classification of various special cases and their reduction relations will be helpful in later study, especially the study on computational complexity.

Chapter 3

Complexity Results

3.1 Overview

This chapter presents the computational complexity results of the selection and assignment problem with MQC constraints for various special cases, which are specified in terms of a four-field notation $Obj(\alpha, \beta, \gamma)$ in the last Chapter 2.

We summarize the main result of this chapter in Table 3.1. From these results, we can derive a complete mapping of the computational complexity for all the special cases by their reduction relations shown in Figure 2.2. For Table 3.1, we use Unary $\mathcal{NP}$-hard and Binary $\mathcal{NP}$-hard, defined by [12], to express that the particular problem is $\mathcal{NP}$-hard with respect to a unary and binary encoding of the data, respectively. Recall that the only difference between unary $\mathcal{NP}$-hardness and binary $\mathcal{NP}$-hardness is whether or not the $\mathcal{NP}$-hard problem has a pseudo-polynomial algorithm if $P \neq NP$. Along with each result, we provide a reference to where its proof can be found. The entries "$*$" in the γ column and rows of the *Feasibility* group imply that those results holds true for any specification of the fixed selection cost (f_i or $f_i = 0$), because the total cost (2.1) can be neglected when we are finding only a feasible solution.

In this chapter, for each special case considered, we will either give a polynomial algorithm, or prove its binary $\mathcal{NP}$-hardness with a pseudo-polynomial algorithm, or prove its unary $\mathcal{NP}$-hardness to show that no efficient algorithm seems to exist. The presentation

includes two parts, one for the cases of finding optimal solutions, and the other for the cases of finding feasible solutions.

Table 3.1: The Complexity Results

Objective	α	β	γ	Complexity Results	
Optimization	$w = 0$	$s_i = \infty$	$f_i = 0$	$O(nm)$	Thm 3.1
	$w = 0$	s_i	$f_i = 0$	$O((n+m)^4 log(n+m))$	[20]
	$w = 0$	$s_i = \infty$	f_i	Unary $\mathcal{NP}$-hard	[10]
	$w = 1$	$s_i = \infty$	$f_i = 0$	Unary $\mathcal{NP}$-hard	Thm 3.4
Feasibility	w	$s_i = \infty$	$*$	$O(n)$	Thm 3.5
	$w = 0$	s_i	$*$	$O(n+m)$	Thm 3.7
	$w = 1$	s_i	$*$	$O(n+m)$	Thm 3.15
	$w = 2$	s_i	$*$	Binary $\mathcal{NP}$-hard $O(mD^6 + n)$	Thm 3.13 Cor 3.12
	$w = \lambda$	s_i	$*$	Binary $\mathcal{NP}$-hard $O(mD^{2\lambda+2} + n)$	Cor 3.14 Thm 3.11
	w	s_i	$*$	Unary $\mathcal{NP}$-hard	Thm 3.16

3.2 Complexity of Finding an Optimal Solution

We begin with the case $Optimization(w = 0, s_i = \infty, f_i = 0)$, which can be reduced to all the other optimization cases by the reduction relations shown in Figure 2.2. For this easiest optimization case, both the maximum capacity constraint (2.3) and the MQC rule (2.4) are relaxed. Besides these, the absence of the fixed selection costs allows us to select all the shipping agents for free. Therefore, we can assign each transportation request greedily to the agent who bids the lowest transportation cost for that request. This greedy process is shown in Algorithm 3.1, which holds the following result.

Theorem 3.1 *Algorithm 3.1 generates an optimal solution for the case* $Optimization(w = 0, s_i = \infty, f_i = 0)$ *in* $O(nm)$ *time.*

Proof The generating solution is optimal, because for any request $j \in J$, shipping one container will cost at least $\min_{i \in I} c_{i,j}$ and this minimum cost has been exactly achieved

in Algorithm 3.1. Its time complexity is $O(nm)$, because finding a proper agent for one request needs $O(m)$ time and there are n requests in total. ∎

Algorithm 3.1 Solving *Optimization*$(w = 0, s_i = \infty, f_i = 0)$

1: Initially, set $z_{i,j} \leftarrow 0$ for all $i \in I$ and $j \in J$;
2: Select all the shipping agents: $A \leftarrow \{1, 2, ..., m\}$;
3: **for all** the transportation request $j \in J$ **do**
4: Let k be any agent such that $c_{k,j} = \min_{i \in I} c_{i,j}$;
5: Assign full demands of request j to agent k: $z_{k,j} \leftarrow d_j$;
6: **end for**
7: Output the selected set A and assignments $z_{i,j}$ for $i \in A$ and $j \in J$;

The next three cases extend the easiest case *Optimization*$(w = 0, s_i = \infty, f_i = 0)$ by appending the maximum capacity constraint (2.3), the fixed selection costs f_i, and the MQC rule (2.4) respectively.

Firstly, the case *Optimization*$(w = 0, s_i, f_i = 0)$ includes the maximum capacity constraint, but still ignores the fixed selection costs f_i and relaxes the MQC rule. This case is equivalent to the classical transportation problem [3], which can be solved by the minimum cost flow algorithm. Moreover, Orlin [20] presents an efficient algorithm for solving minimum cost flow, leading the following result.

Theorem 3.2 *The minimum cost flow algorithm generates an optimal solution for the case Optimization*$(w = 0, s_i, f_i = 0)$ *in* $O((n + m)^4 log(n + m))$ *time.*

Secondly, we consider the case *Optimization*$(w = 0, s_i = \infty, f_i)$ that counts the fixed selection costs f_i but relaxes the maximum capacity constraint and the MQC rule. Since this case is equivalent to the uncapacitated facility location problem (UFLP) [10], a well-known unary $\mathcal{NP}$-hard problem, we have the following result.

Theorem 3.3 *Finding an optimal solution for the case Optimization*$(w = 0, s_i = \infty, f_i)$ *is unary* $\mathcal{NP}$*-hard.*

Lastly, we consider the case *Optimization*$(w = 1, s_i = \infty, f_i = 0)$ that has one region holding the MQC rule but relaxes the maximum capacity constraint and ignores the fixed selection costs. We now show that the case *Optimization*$(w = 1, s_i = \infty, f_i = 0)$ is intractable by proving the following result.

Theorem 3.4 *Finding an optimal solution for the case Optimization*$(w = 1, s_i = \infty, f_i = 0)$ *is unary* $\mathcal{NP}$*-hard.*

Proof We use a reduction from the following problem, which is known to be unary NP-complete.

Cover By 3-Sets (X3C) [12]: given a set $X = \{1, ..., 3q\}$ and a collection $C = \{C_1, ..., C_m\}$ with each member $C_i \subseteq X$ and $|C_i| = 3$ for $i = 1, ..., m$, does C contain an exact cover for X, i.e. a subcollection $C' \subseteq C$ such that every element of X occurs in exactly one member of C'?

From any arbitrary instance of $X3C$, consider the following polynomial reduction to an instance of *Optimization*$(w = 1, s_i = \infty, f_i = 0)$. Let each element $j \in X$ indicate a transportation request with a unit demand, then we have $J = \{1, ..., 3q\}$ with $d_j = 1$ for $j \in J$. Suppose the shipping agent set I is $\{1, ..., m\}$. For each agent $i \in I$ and request $j \in J$, the bid cost $c_{i,j} = 0$ if the element $j \in C_i$; otherwise $c_{i,j} = 1$. For the only region holding the MQC rule, assume all the transportation requests are shipped to region 1. Suppose its minimum quantity is three units, then we have $R = \{1\}$, $J_1 = \{1, ..., 3q\}$ and $b_1 = 3$. As assumed in the specification, only the demand constraint (2.2) and the MQC rule (2.4) need to be considered, and the total cost is relaxed to (2.5) as well. We now prove that the minimum total cost for the instance of *Optimization*$(w = 1, s_i = \infty, f_i = 0)$ is zero if and only if the $X3C$ has an exact cover.

On one hand, if there exists an exact cover C' for $X3C$, we select all the agents leading $A = \{1, .., m\}$, and assign the requests based on the exact cover C'. For each agent $i \in I$ and request $j \in J$, the assignment $z_{i,j} = 1$ if element j is covered by the subcollection C_i and C_i is in the exact cover C'; otherwise $z_{i,j} = 0$. Because C' is an exact cover, the demand constraint (2.2) is satisfied. Now we examine the MQC constraint (2.4). For each selected agent $i \in A$, if $C_i \in C'$ then its total shipment to region 1 is $\sum_{j \in J_1} z_{i,j} = |C_i| = 3$, otherwise $C_i \notin C'$ then we have $\sum_{j \in J_1} z_{i,j} = 0$. Both satisfy the MQC constraint. It is easy to see that its total cost (2.5) is zero achieving the minimum.

On the other hand, if we have a feasible selection A and assignments $z_{i,j}$ for $i \in A$ and $j \in J$ with zero total cost, consider the subcollection $C' = \{C_i | i \in A \text{ and } \sum_{j \in J} z_{i,j} > 0\}$.

To see C' is an exact cover of X, we need to prove that for each element $u \in X$, there exists a unique index p such that $C_p \in C'$ and $u \in C_p$. Since $X = J$, the corresponding request $u \in J$. Noting $\sum_{i \in A} z_{i,u} = 1$ by the demand constraint (2.2), we can assume p is the unique index with $p \in A$ and $z_{p,u} = 1$, leading $\sum_{j \in J} z_{p,j} > 0$ and $C_p \in C'$. Because the total cost (2.5) is zero and $z_{p,u} = 1$ implying $c_{p,u} = 0$, we have $u \in C_p$ that proves the existence. For the uniqueness, consider any other subset $C_q \in C'$ where $q \neq p$. We are going to prove the element $u \notin C_q$. Because for the request u, its partial assignments $z_{q,u} + z_{p,u} \leq 1$ and $z_{p,u} = 1$, we have $z_{q,u} = 0$. Note $C_q \in C'$ and $J_1 = J$, implying $\sum_{j \in J_1} z_{q,j} > 0$. To satisfy the MQC constraint (2.4), we have $\sum_{j \in J_1} z_{q,j} = 3$. This leads $\sum_{j \in C_q} z_{q,j} + \sum_{j \notin C_q} z_{q,j} = 3$. However, for $j \notin C_q$ the assignment $z_{q,j} = 0$, because its bid cost $c_{q,j} = 1$ and the total cost (2.1) is zero. So $\sum_{j \in C_q} z_{q,j} = 3$. For all $j \in C_q$, since $|C_q| = 3$ and its partial assignment $z_{q,j} \leq 1$, we obtain $z_{q,j} = 1$. Noting $z_{q,u} = 0$ we have $u \notin C_q$, which leads the uniqueness and completes the proof. ∎

3.3 Complexity of Finding a Feasible Solution

Feasibility cases are studied in this section. We only need to consider the constraints (2.2)–(2.4) but the total cost (2.1). We specify the field γ to be "$*$", if the specification of the fixed selection cost doesn't change the complexity results for any feasibility case. Therefore, we can select all the agents for safety, in other words, we can assume $A = \{1, 2, ..., m\}$ for each feasibility case, and concentrate on how to obtain the assignment $z_{i,j}$ for $i \in A$ and $j \in J$.

For uncapacitated cases with $s_i = \infty$, let us look at the most general one $Feasibility(w, s_i = \infty, *)$. Its feasible assignment can be constructed straightforward by assigning all the transportation requests with full demands to a certain agent. The process is described in Algorithm 3.2, which establishes the following result.

Theorem 3.5 *Algorithm 3.2 generates a feasible solution for the case* $Feasibility(w, s_i = \infty, *)$ *in* $O(n)$ *time.*

Proof Note that the maximum capacity constraint has been relaxed and all the demands

have been assigned. If the minimum quantity b_r does not exceed the total demands of transportation requests in region r for $r \in R$, the MQC rule must be satisfied leading its feasibility; otherwise b_r is too large to allow a feasible solution to exist. So Algorithm 3.2 is correct. Its time complexity is $O(n)$, because of the n requests. ∎

Algorithm 3.2 Solving $Feasibility(w, s_i = \infty, *)$

1: **if** for some region $r \in R$, b_r exceed $\sum_{j \in J_r} d_j$, the total demand of requests in region r **then**
2: Output "No Feasible Solution";
3: **else**
4: Initially, $z_{i,j} \leftarrow 0$ for all $i \in I$ and $j \in J$;
5: Select all the shipping agents: $A \leftarrow \{1, 2, ..., m\}$;
6: **for all** the transportation request $j \in J$ **do**
7: Assign full demands of request j to the agent 1: $z_{1,j} \leftarrow d_j$;
8: **end for**
9: Output the selected set A and assignments $z_{i,j}$ for $i \in A$ and $j \in J$;
10: **end if**

For the capacitated case with $\beta = s_i$, we study different specifications for α, the number of regions holding MQC rules. Before studying the particular cases, we can easily obtain a general result for the necessary condition of feasibility as follows.

Lemma 3.6 *For the capacitated case $Feasibility(\alpha, s_i, *)$ with any specification of α, if a feasible solution exists, then the total agents' maximum capacity is at least as large as the total requests' demand, i.e. $S \geq D$ according to our notations.*

Let us study the case $Feasibility(w = 0, s_i, *)$, in which the MQC constraint is relaxed. Since the case $Optimization(w = 0, s_i, f_i = 0)$ is polynomial solvable according to Theorem 3.1, by the same minimum cost flow algorithm [20] we can obtain a feasible solution for the case $Feasibility(w = 0, s_i, *)$ in $O((n + m)^4 log(n + m))$ time as well.

Here, we present a more efficient algorithm to solve the case $Feasibility(w = 0, s_i, *)$. It is shown in Algorithm 3.3, which holds the following result.

Theorem 3.7 *Algorithm 3.3 generates a feasible solution for the case $Feasibility(w = 0, s_i, *)$ in $O(n + m)$ time.*

Proof Because the MQC constraints are relaxed, only the demand constraint (2.2) and the capacity constraint (2.3) need to be satisfied. Algorithm 3.3 maintains a request list containing all the unsatisfied requests, and an agent list holding all the available agents. The head of the request list is denoted by u with remnant demand d'_u, and the head of the agent list is denoted by p with remnant capacity s'_p. For each loop, we examine whether the head agent's remnant capacity s'_p can satisfy the head request's remnant demand d'_u. If it can, we assign all the remnant demand d'_u of the head request u to the head agent p, decrease the remnant capacity s'_p, and remove the head request u; otherwise, we can only assign at most s'_p to the head agent p for the request u, decrease the remnant demand of d'_u, and remove the head agent p. Such a process of examination and assignment continues until all the requests have been satisfied. It is easy to see that if the total agents' maximum capacity is at least as large as the total requests' demand ($S \geq D$), the above process will stop with a feasible solution; otherwise no feasible solution exists by Lemma 3.6. This proves the correctness of Algorithm 3.3. Noting that for each loop either a request or an agent is removed, and that there are n requests and m agents, we obtain its time complexity $O(n+m)$. ∎

Therefore, by Algorithm 3.3 and Theorem 3.7, the necessary feasibility condition stated in Lemma 3.6 is also sufficient for the case $Feasibility(w = 0, s_i, *)$, leading the following corollary.

Corollary 3.8 *A feasible solution exists for the case $Feasibility(w = 0, s_i, *)$, if and only the total agents' maximum capacity is at least as large as the total requests' demand, i.e. $S \geq D$.*

In the rest of this section, we study the feasibility cases holding MQC constraints. We start with the case $Feasibility(w = \lambda, s_i, *)$, where the MQC region number w equal to a fixed integer λ. A new notation $t_{i,r}$ is needed to indicate the total regional shipment assigned to the selected agent i for all transportation requests in region r, where $i \in A$ and $r \in U$. (Recall that the universal set is $U = R \cup \{0\}$, where the region 0 holds request set J_0 unrestricted to MQC rules by fixing the minimum quantity $b_0 = 0$.)

Algorithm 3.3 Solving $Feasibility(w = 0, s_i, *)$

if the total agents' maximum capacity is less than the total requests' demand ($S < D$) **then**
  Output "No Feasible Solution";
else
  Assume the requests and the agents are listed in the order $1, 2, ..., n$ and $1, 2, ..., m$ respectively.
  Let u indicate the head of the unsatisfied request list;
  Let p indicate the head of the available agent list;
  Let d'_j denote the remnant demand of request j for $j \in J$;
  Let s'_i denote the remnant capacity of agent i for $i \in I$;
  Initially, $u \leftarrow 1$, $p \leftarrow 1$, $d'_j = d_j$ for $j \in J$, $s'_i = s_i$ for $i \in I$, and $z_{i,j} \leftarrow 0$ for $i \in I$ and $j \in J$;
  Select all the shipping agents: $A \leftarrow \{1, 2, ..., m\}$;
  while the request list is not empty ($u \leq n$) **do**
    if the head agent's remnant capacity can satisfy the head request's remnant demand ($s'_p \geq d'_u$) **then**
      Assign all remnant demand of request u to agent p: $z_{p,u} \leftarrow d'_u$;
      Decrease head agent's remnant capacity: $s'_p \leftarrow s'_p - z_{p,u}$;
      Remove the head request: $u \leftarrow u + 1$;
    else
      Assign part remnant demand of request u to agent p: $z_{p,u} \leftarrow s'_p$;
      Decrease head request's remnant demand: $d'_u \leftarrow d'_u - z_{p,u}$;
      Remove the head agent: $p \leftarrow p + 1$;
    end if
  end while
  Output the selected set A and assignments $z_{i,j}$ for $i \in A$ and $j \in J$;
end if

Our basic idea for solving the case $Feasibility(w = \lambda, s_i, *)$ is:

Stage 1: Select all the agents, i.e. $A = \{1, 2, ..., m\}$;

Stage 2: For each agent $i \in A$ and region $r \in U$, determine the total regional shipment $t_{i,r}$;

Stage 3: Based on A and $t_{i,r}$, obtain the assignment $z_{i,j}$ for $i \in A$ and $j \in J$.

The reason for selecting all the agents in Stage 1 has been discussed in the beginning of this section. To see what constraints the total regional shipment $t_{i,r}$ for $i \in A$ and $r \in U$ should satisfy, let us examine any feasible assignment $z_{i,j}$ for the case $Feasibility(w = \lambda, s_i, *)$, where $i \in A$ and $j \in J$. By the definition of $t_{i,r}$, we have that for each selected

agent $i \in A$ and region $r \in U$,

$$t_{i,r} = \sum_{j \in J_r} z_{i,j} \tag{3.1}$$

Therefore, the demand constraint (2.2) implies that the total shipments to region r should be the same as its total demands for $r \in U$, i.e.

$$\sum_{i \in A} t_{i,r} = \sum_{j \in J_r} d_j. \tag{3.2}$$

The capacity constraint (2.3) implies that the total shipments to all regions by one agent i should not exceed its maximum capacity s_i for $i \in A$, i.e.

$$\sum_{r \in U} t_{i,r} \leq s_i. \tag{3.3}$$

And, the MQC constraint (2.4) implies that for each agent $i \in A$ and each MQC region $r \in R$, the total regional shipment $t_{i,r}$ must satisfy:

$$t_{i,r} = 0 \text{ or } t_{i,r} \quad \geq \quad b_r. \tag{3.4}$$

Therefore, we obtain a sufficient condition of the feasibility for the case $Feasibility(w = \lambda, s_i, *)$ as follows.

Lemma 3.9 *if there exists a feasible assignment $z_{i,j}$ for $i \in A$ and $j \in J$ for the case $Feasibility(w = \lambda, s_i, *)$, we can have its total regional shipment $t_{i,r}$ for $i \in A$ and $r \in U$ by (3.1) to satisfy the constraints (3.2)–(3.4).*

The condition is also sufficient. We show this by proving the following result, which partially solves the case $Feasibility(w = \lambda, s_i, *)$.

Lemma 3.10 *Algorithm 3.4 generates a feasible assignment $z_{i,j}$ with $i \in A$ and $j \in J$ for the case $Feasibility(w = \lambda, s_i, *)$ in $O(n + \lambda m)$ time, if given the total regional shipment $t_{i,r}$ for $i \in A$ and $r \in U$ such that the constraints (3.2)–(3.4) are satisfied.*

Proof The idea follows. Suppose we have the total regional shipment $t_{i,r}$ satisfying constraints (3.2)–(3.4) for $i \in A$ and $r \in U$. Note the regional request sets $J_0, J_1, ..., J_\lambda$

forms a disjoint partition of the whole set J. We can obtain feasible assignments $z_{i,j}$ of all the agents $i \in A$, for requests j in $J_0, J_1, ..., J_\lambda$ respectively.

For each job set J_r where region $r \in U$, consider the following instance of the case $Feasibility(w = 0, s_i, *)$. The request set becomes J_r holding requests only in region r, and the demand of request $j \in J_r$ is still d_j. The agent set is A, and the maximum capacity of agent $i \in A$ changes to $t_{i,r}$, which indicates the total regional shipment of agent i for region r.

In this new instance, since the total agents' maximum capacity is $\sum_{i \in A} t_{i,r}$ equal to the total demands $\sum_{j \in J_r} d_j$ by the constraint (3.2), we can apply the Algorithm 3.3 to generate feasible assignments $z_{i,j}$ for $i \in A$ and $j \in J_r$. Because the demand is unchanged, $z_{i,j}$ also satisfies the original demand constraint (2.2) for $i \in A$ and $j \in J_r$. In addition, since the total capacity $\sum_{i \in A} t_{i,r}$ is the same as the total demand $\sum_{j \in J_r} d_j$, to satisfy all the demands in this new instance, considering the restriction of capacities $t_{i,r}$ for $i \in A$ and $r \in R$, we have that the relation (3.1) must hold true, i.e. $\sum_{j \in J_r} z_{i,j} = t_{i,r}$. Because $t_{i,r}$ satisfies the constraint (3.4), the original MQC rule (2.4) is satisfied for region r.

We solve those $\lambda + 1$ instances of the case $Feasibility(w = 0, s_i, *)$ for $J_0, J_1, ..., J_\lambda$ respectively. Afterwards, we can obtain assignments $z_{i,j}$ for $i \in A$ and $j \in J$ such that both the original demand constraint (2.2) and MQC rules (2.4) are satisfied. To see the original maximum capacity constraint (2.3) is also satisfied, recall that for each region $r \in U$ the relation (3.1) holds true. This indicates that $t_{i,r}$ is exactly equal to the total regional shipment of $z_{i,j}$ for $i \in A$, $r \in U$ and $j \in J_r$. So by (3.3), we know that $z_{i,j}$ satisfies its original maximum capacity constraint (2.3) for $i \in A$ and $j \in J$.

We give a formal statement of the above process in Algorithm 3.4. Note that the Algorithm 3.3 has been called for $|U|$ times and each consumes $O(|J_r| + m)$ time. So its time complexity of Alogrithm 3.3 is $O(n + \lambda m)$, for $|J_0| + |J_1| + ... + |J_\lambda| = n$ and $|U| = \lambda + 1$. ∎

To complete solving the case $Feasibility(w = \lambda, s_i, *)$, we need to obtain the total regional shipment $t_{i,r}$ to satisfy constraints (3.2)–(3.4) for $i \in A$ and $r \in U$ before using the Algorithm 3.4. This can be achieved by a dynamic programming algorithm proposed

Algorithm 3.4 Solving $Feasibility(w = \lambda, s_i, *)$ with Total Regional Shipments Given

1: The selected set is $A = \{1, 2, ..., m\}$, and the total regional shipment is $t_{i,r}$ for $i \in A$ and $r \in U$, which satisfy constraints (3.2)–(3.4);
2: Initially, set $z_{i,j} \leftarrow 0$ for $i \in A$ and $j \in J$;
3: **for all** region $r \in U$ **do**
4: Construct a instance of $Feasibility(w = 0, s_i, *)$, where the request set is J_r with each request $j \in J_r$ having demand d_j, and the agent set is A with each agent $i \in A$ holding maximum capacity $t_{i,r}$;
5: Solve this instance by Algorithm 3.3 to obtain a feasible assignment $z_{i,j}$ for $i \in A$ and $j \in J_r$;
6: **end for**
7: Output assignments $z_{i,j}$ for $i \in A$ and $j \in J$;

as follows.

Let $I(p) = \{1, ..., p\}$ denote the set of the first p agents in I for $1 \leq p \leq m$. Let $D_r = \sum_{j \in J_r} d_j$ denote the total demands for requests in region $r \in U$. To satisfy demands in the constraint (3.2), we indicate the partial regional shipments transported by those agents $i \in I(p)$ for region $r \in U$ as follows, where $p = 1, ..., m$ respectively. Let $v_r = \sum_{i \in I(p)} t_{i,r}$ denote the partial regional shipments for region $r \in U$, so that state variables of the dynamic programming algorithm are $(p, \overrightarrow{v})$, where vector $\overrightarrow{v} = \langle v_0, v_1, ..., v_\lambda \rangle$ and $v_r \leq D_r$ for all $r \in U$. Then, the algorithm recursively computes $G(p, \overrightarrow{v})$ that denotes the true value of the statement: there exists integer values of $t_{i,r}$ for $i \in I(p)$ and $r \in U$, such that

$$\sum_{i \in I(p)} t_{i,r} = v_r, \text{ for all } r \in U; \tag{3.5}$$

$$\sum_{r \in U} t_{i,r} \leq s_i, \text{ for all } i \in I(p); \tag{3.6}$$

$$t_{i,r} = 0 \text{ or } t_{i,r} \geq b_r, \text{ for all } i \in I(p) \text{ and } r \in U. \tag{3.7}$$

It is easy to see that there exists $t_{i,r}$ to satisfy these constraints (3.2)–(3.4) for $i \in A$ and $r \in U$, if and only if $G(m, \langle D_0, D_1, ..., D_\lambda \rangle)$ is true. Let $\overrightarrow{V}$ denote the vector $\langle D_0, D_1, ..., D_\lambda \rangle$. So our objective is to obtain the value of $G(m, \overrightarrow{V})$.

Initially, set $G(0, \overrightarrow{v})$ true if $\overrightarrow{v} = \langle 0, 0, ..., 0 \rangle$; otherwise set it false.

In the iteration $p = 1, 2, ..., m$, consider each vector $\overrightarrow{v}$ with field $v_r \leq D_r$ for $r \in U$. To obtain the current value of $G(p, \overrightarrow{v})$, we need to enumerate all the possible values

of $t_{p,r}$ for $r \in U$. Note that $t_{p,r}$ can not exceed v_r by (3.5). We let Ψ be the set of vectors $\overrightarrow{t} = \langle t_{p,0}, t_{p,1}, , ..., t_{p,\lambda} \rangle$ with $t_{p,r} \leq v_r$ for $r \in U$ and satisfying (3.6) and (3.7). Accordingly by (3.5), we can obtain the following recursion: the value of $G(p, \overrightarrow{v})$ is true if and only if there exists a vector $\overrightarrow{t} \in \Psi$ such that the value of $G(p-1, \overrightarrow{v} - \overrightarrow{t})$ is true.

After these m iterations, we can examine the value of $G(m, \overrightarrow{V})$. If it is true, then the feasible values of $t_{i,r}$, satisfying constraints (3.2)–(3.4) for $i \in A$ and $r \in U$, can be derived based on the previous process of recurrent computations; otherwise no feasible $t_{i,r}$ exists.

Whenever we having a feasible $t_{i,r}$, we can solve $Feasibility(w = \lambda, s_i, *)$ by Algorithm 3.4. This establishes the following results.

Theorem 3.11 *Algorithm 3.5 generates a feasible solution for the case* $Feasibility(w = \lambda, s_i, *)$ *in* $O(mD^{2\lambda+2} + n)$ *time.*

Proof The correctness of Algorithm 3.5 has been discussed. Let us estimate its time performance. Consider each iteration p and each vector $\overrightarrow{v}$ with field $v_r \leq D_r$ for $r \in U$. By the definition of set Ψ, we know that its size $|\Psi|$ is at most $O(D^{\lambda+1})$. So the time complexity for each recursion is $O(D^{\lambda+1})$. Because there are m iterations, each with at most $D^{\lambda+1}$ vectors, the total time complexity of the whole recurrent computations is $O(mD^{2\lambda+2})$. Noting that Algorithm 3.4 will be called once, consuming $O(n + \lambda m)$ time, we obtain that the time complexity of Algorithm 3.5 is $O(mD^{2\lambda+2} + n)$. ∎

We have given a pseudo-polynomial algorithm for $Feasibility(w = \lambda, s_i, *)$. Moreover, $Feasibility(w = \lambda, s_i, *)$ is also binary $\mathcal{NP}$-hard, which indicates that no polynomial algorithm exists. This binary $\mathcal{NP}$-hardness can be proved even for a restricted case, $Feasibility(w = 2, s_i, *)$, with $w = 2$ particularly. Before proving its binary $\mathcal{NP}$-hardness, we can have the following corollary for $Feasibility(w = 2, s_i, *)$ from Theorem 3.11 directly.

Corollary 3.12 *Algorithm 3.5 generates a feasible solution for the case* $Feasibility(w = 2, s_i, *)$ *in* $O(mD^6 + n)$ *time by specifying* $\lambda = 2$.

Algorithm 3.5 Solving $Feasibility(w = \lambda, s_i, *)$

if the total agents' maximum capacity is less than the total requests' demand ($S < D$) **then**
  Output "No Feasible Solution";
else
  Select all the shipping agents: $A \leftarrow \{1, 2, ..., m\}$;
  Initially, set $G(0, \overrightarrow{v}) \leftarrow$true, if $\overrightarrow{v} = \langle 0, 0, ..., 0\rangle$; otherwise $G(0, \overrightarrow{v}) \leftarrow$false;
  for $p = 1$ to m **do**
    for all $\overrightarrow{v}$ with all fields $v_r \leq D_r$ for $r \in U$ **do**
      let Ψ denote the set of vectors $\overrightarrow{t} = \langle t_{p,0}, t_{p,1}, , ..., t_{p,\lambda}\rangle$, where $t_{p,r} \leq v_r$ for $r \in U$ and constraints (3.6)(3.7) are satisfied;
      $G(p, \overrightarrow{v}) \leftarrow$false;
      Set $G(p, \overrightarrow{v}) \leftarrow$true, if there exists $t \in \Psi$ such that $G(p-1, \overrightarrow{v} - \overrightarrow{t})$ is true; otherwise $G(p, \overrightarrow{v}) \leftarrow$false;
    end for
  end for
  Let $\overrightarrow{V}$ denote the vector $\langle D_0, D_1, ..., D_\lambda\rangle$;
  if $G(m, \overrightarrow{V})$ is false **then**
    Output "No Feasible Solution";
  else
    Retrieve $t_{i,r}$ for $i \in S$ and $r \in U$ based on Ψ;
    Call Algorithm 3.4 to obtain assignments $z_{i,j}$ for $i \in A$ and $j \in J$;
    Output the set A and assignments $z_{i,j}$ for $i \in A$ and $j \in J$;
  end if
end if

Now let us prove that $Feasibility(w = 2, s_i, *)$ is binary $\mathcal{NP}$-hard.

Theorem 3.13 *Finding a feasible solution for the case* $Feasibility(w = 2, s_i, *)$ *is binary* $\mathcal{NP}$*-hard.*

Proof We use a reduction from the following problem, which is known to be binary $\mathcal{NP}$-complete.

Partition [12]: given a set $X = \{1, ..., 2q\}$ with a positive integer value $g(i)$ for each $i \in X$. Does there exists a subset X' with half size and half sum of values of X, i.e. $|X'| = \frac{1}{2}|X|$ and $\sum_{i \in X'} g(i) = \frac{1}{2}\sum_{i \in X} g(i)$?

From any instance of **Partition**, consider the following polynomial reduction to the instance of $Feasibility(w = 2, s_i, *)$. Let each element $i \in X$ indicate a shipping agent, so I equals to $\{1, ..., 2q\}$. Noting X needed to be participated, we consider two transportation request to the two different regions holding MQC rules. So we have $J = \{1, 2\}$,

$R = \{1, 2\}$ and $J_r = \{r\}$ for region $r \in R$. Moreover, to agree with the requirements of **Partition**, we set the maximum capacities, the demands and the minimum quantity properly as follows. Let δ denote an integer larger than the sum of values $g(i)$ for $i \in X$, say $\delta = \sum_{i \in X} g(i) + 1$. Therefore, we set the maximum capacity $s_i = g(i) + \delta$ for $i \in I$, the demand $d_j = \frac{1}{2} \sum_{i \in X} g(i) + q\delta$ for $j = 1$ and 2, and the minimum quantity $b_r = \delta$ for $r = 1$ and 2. We now prove that a feasible solution for the instance of $Feasibility(w = 2, s_i, *)$ exists if and only if there is a solution X' to **Partition**.

On one hand, if **Partition** has a subset $X' \subseteq X$ satisfying $|X'| = \frac{1}{2}|X|$ and $\sum_{i \in X'} g(i) = \frac{1}{2} \sum_{i \in X} g(i)$, we can select all the agents by $A = \{1, ..., 2q\}$, and assign requests based on X'. We assign request 1 (respectively, 2) to agent i for $i \in X'$ (resp., $i \in X - X'$) leading $z_{i,1} = s_i$. It is easy to verify that this assignments is feasible.

On the other hand, if we have a feasible selection set A and assignments $z_{i,j}$ for $j = 1, 2$ and $i \in A$, consider the set X' that consists of all the agents having shipments of request 1, i.e. $X' = \{i | i \in A, z_{i,1} > 0\}$. Clearly $X' \subseteq X$. And for each agent $i \in X'$, by the MQC constraint (2.4) for region 1, we have $z_{i,1} \geq b_1 = \delta$. Then its remanent capacity becomes $s_i - z_{i,1} \leq g(i)$, which is less than the minimum quantity $b_2 = \delta$ of region 2. From the MQC constraint (2.4) for region 2, we have $z_{i,2} = 0$ for each agent $i \in X'$. Note that the total capacity exactly equals to the total demand (i.e. $\sum_{i \in I} s_i = \sum_{j \in J} d_j$). To satisfy all the demands, each agent must have as many of the shipments as its capacity, implying $A = I$ and $z_{i,1} + z_{i,2} = s_i$ for agent $i \in A$. Therefore, if $i \in X'$, we have $z_{i,1} = s_i = g(i) + \delta$ because $z_{i,2} = 0$; otherwise if $i \in X - X'$, we have $z_{i,2} = s_i = g(i) + \delta$ because $z_{i,1} = 0$. Since the demands of two requests 1 and 2 are equal, their shipments must be equal, that is $\sum_{i \in X'} z_{i,1} = \sum_{i \in X - X'} z_{i,2}$. Equivalently, we have:

$$\sum_{i \in X'} g(i) + |X'|\delta = \sum_{i \in X - X'} g(i) + |X - X'|\delta.$$

By $\delta > \sum_{i \in X} g(i)$, we know $|X'| = |X - X'|$ and $\sum_{i \in X'} g(i) = \sum_{i \in X - X'} g(i)$. This leads $|X'| = \frac{1}{2}|X|$ and $\sum_{i \in X'} g(i) = \frac{1}{2} \sum_{i \in X} g(i)$. ∎

The complexity result above can be applied for $Feasibility(w = \lambda, s_i, *)$ as well, by the reduction from $w = 2$ to $w = \lambda$ in the field α shown in Figure 2.2(b). This derived

the following corollary.

Corollary 3.14 *Finding a feasible solution for the case* $Feasibility(w = \lambda, s_i, *)$ *is binary* $\mathcal{NP}$*-hard.*

We have seen that the capacitated feasibility case of $w \geq 2$ is binary $\mathcal{NP}$-hard and has a pseudo-polynomial algorithm. For $w = 1$, we now present an efficient polynomial algorithm. In this case $Feasibility(w = 1, s_i, *)$, we have $R = \{1\}$ and $U = \{0, 1\}$. Recall that the polynomial Algorithm 3.4 can be still applied for the case $Feasibility(w = 1, s_i, *)$ to obtain feasible assignments $z_{i,j}$ for $i \in A$ and $j \in J$, if given the total regional shipments $t_{i,r}$ for $i \in A$ and $r \in U$.

Now let us consider how to obtain $t_{i,r}$ for $i \in A$ and $r \in R$ in this case. We suppose $S \geq D$ because otherwise no feasible solution exists by Lemma 3.6, and assume that the agents are ordered non-decreasingly on their maximum capacities, i.e. $s_1 \geq s_2 \geq ... \geq s_m$. Moreover, let c_1 indicate the number of agents whose capacities are large enough (i.e. at least b_1) to serve request in region 1.

Suppose we have got the total regional shipment $t_{i,r}$ for $i \in A$ and $r \in U$ satisfying constraints (3.2)–(3.4). We now prove the condition $\sum_{i=1}^{c} s_i \geq D_1$ is satisfied, where $c = \min(c_1, \lfloor D_1/b_1 \rfloor)$. Let $A_1 = \{i | t_{i,1} > 0\}$ be the set of agents who has shipments to region 1. Let $D_1 = \sum_{j \in J_1} d_j$ denote the total demands of requests in region 1. By the MQC constraint (3.4), we have $t_{i,1} \geq b_1$ for $i \in A_1$. Thus the number of agents serving requests in region 1 is at most c by (3.2), i.e. $|A_1| \leq c$. So $\sum_{i \in A_1} s_i \leq \sum_{i=1}^{c} s_i$, because of the non-decreasing order of s_i for $i \in I$. According to the capacity and demand constraints (2.2)(2.3), we obtain the necessary condition: $\sum_{i=1}^{c} s_i \geq D_1$.

To see the condition is also sufficient, we will show that if $\sum_{i=1}^{c} s_i \geq D_1$, we can safely choose $A_1 = \{1, ..., c\}$ to serve the requests in region 1 and obtain $t_{i,r}$ for $i \in S$ and $r \in U$ as follows. Firstly, excluding quantity b_1 reserved for the minimum quantity of region 1, the remnant capacity of agent i can be denoted by $s'_i = s_i - b_1$ for $1 \leq i \leq c$ and the total remnant demands in region 1 is reduced to $D'_1 = D_1 - cb_1$. By $\sum_{i \in A_1} s_i \geq D_1$, we have $\sum_{i \in A_1} s'_i \geq D'_1$. Let τ be such an index that $c \geq \tau \geq 1$ and $\sum_{i=1}^{\tau} s'_i \geq D'_1 > \sum_{i=1}^{\tau-1} s'_i$. Then the first τ agents can satisfy the remnant demands D'_1, since we can assign s'_i to each

agent i where $i < \tau$, and assign the rest $D'_1 - \sum_{i=1}^{\tau-1} s'_i$ to agent τ. Furthermore, including the reserved quantity b_1, we can satisfy all the demands D_1 by assigning agents in A_1 with $t_{i,1} = s_i$ if $i \leq \tau - 1$, and $t_{\tau,1} = D_1 - (c - \tau)b_1 - \sum_{i=1}^{\tau-1} s_i$, and $t_{i,1} = b_1$ if $\tau < i \leq c$. It is easy to verify that both the demand constraint (3.2) and the MQC constraints (3.4) are satisfied for region 1. On the other hand, for region 0, note that the total demands is $D_0 = D - D_1$ and the total remnant capacity of all agents is $S_0 = S - D_1$. By $S \geq D$, we have $S_0 \geq D_0$. In the same manner of satisfying the remnant demands D'_1 in region 1, we can obtain the regional shipment $t_{i,0}$ for $i \in A$ to satisfy constraints (3.2)(3.4) for region 0. Moreover, it is easy to see that the capacity constraint (3.3) for $t_{i,r}$, where $i \in A$ and $r \in U$, is satisfied by all.

When $t_{i,r}$ has been obtained for $i \in A$ and $r \in U$, we can employ Algorithm 3.4 to solve the feasible $z_{i,j}$ for $i \in A$ and $j \in J$. The whole process is formalized in Algorithm 3.6, which leads the following result.

Theorem 3.15 *Algorithm 3.6 generates a feasible solution for the case* $Feasibility(w = 1, s_i, *)$ *in* $O(n + m)$ *time.*

Proof The correctness of Algorithm 3.6 has been discussed. Now we consider its the time performance. Because choosing agents having c largest capacities s_i consumes $O(m)$ time [9] and all the other parts cost $O(n+m)$, the total time complexity is $O(n+m)$. ∎

Lastly, we prove that the general capacitated feasibility case with arbitrary w is unary $\mathcal{NP}$-hard.

Theorem 3.16 *Finding a feasible solution for the case* $Feasibility(w, s_i, *)$ *is unary* $\mathcal{NP}$*-hard.*

Proof We use a polynomial reduction from the following problem, which is known to be unary $\mathcal{NP}$-complete.

3-Partition: given a positive integer bound B and a set $X = \{1, 2, ..., 3w\}$ with a positive integer value $g(i)$ for each $i \in X$ such that the sum $\sum_{i \in X} g(i) = wB$ and $B/4 < g(i) < B/2$ for $i \in X$, can X be partitioned into q disjoint sets $X_1, X_2, ..., X_w$

such that $|X_j| = 3$ and $\sum_{i \in X_j} g(i) = B$ for $j = 1, ..., w$?

The reduction to $Feasibility(w, s_i, *)$ is similar to that constructed in the proof of Theorem 3.13, except that instead of only two MQC regions there, we consider w regions here. For any instance of **3-Partition**, let each element $i \in X$ indicate a shipping agent, so we have $I = \{1, ..., 3w\}$. Since X needed to be partitioned into w subsets, we correspondingly consider w regions holding MQC rules, and each region holds exactly one transportation request. In other words, we let $J = \{1, ..., w\}$, $R = \{1, ..., w\}$ and $J_r = \{r\}$ for $r \in R$. Besides, let δ denote an integer, larger than the sum of values $g(i)$ for $i \in X$, say $\delta = wB + 1$. So, we set the maximum capacity $s_i = g(i) + \delta$ for $i \in I$, the demand $d_j = B + 3\delta$ for $j \in J$, and the minimum quantity $b_r = \delta$ for $r \in R$. We now prove that a feasible solution for the instance of $Feasibility(w, s_i, *)$ exists if and only if there is a solution for **3-Partition**.

On one hand, from a solution $X_1, X_2, ..., X_w$ for **3-Partition**, we can obtain a feasible solution for $Feasibility(w, s_i, *)$ straightforward as follows. Let $S = \{1, 2, ..., 3w\}$ indicating that all agents are selected. Then we assign the request j to those agents $i \in X_j$ with quantity s_i for $j \in J$, i.e. for each request $j \in J$, we set $z_{i,j} = s_i$ if $i \in X_j$, otherwise set $z_{i,j} = 0$. It is easy to verify that such an assignment satisfies all the constraints (2.2)–(2.4).

On the other hand, if we have a feasible solution with a selection set A and assignments $z_{i,j}$ for $i \in I$ and $j \in J$, consider the following w subsets of X. For each request $j \in J$, let subset $X_j = \{i | i \in A, z_{i,j} > 0\}$, indicating the set of agents serving the request j. Noting the MQC constraint (2.4), we have $z_{i,j} \geq b_j = \delta$ for $j \in J$ and $i \in X_j$. Because twice the δ is larger than the maximum capacity for every agent (i.e. $2\delta > s_i$ for $i \in I$), each agent can serve at most one request. So the w subsets $X_1, ..., X_w$ are disjoint. Note the total capacity exactly equal to the total demand. To satisfy all those demands, each agent must ship as many quantities as its capacity, i.e. $A = I$ and $z_{i,1} + z_{i,2} + ... + z_{i,w} = s_i > 0$ for $i \in A$. Recalling that $X_1, ..., X_w$ is disjoint, we know that for each element $i \in X$, there is exactly one index j in $\{1, ..., w\}$ with $z_{i,j} > 0$ such that $i \in X_j$. Therefore $X_1, ..., X_w$ forms a disjoint partition of X, and $z_{i,j} = s_i = g(i) + \delta$

for each agent $i \in X_j$ and request $j \in J$. Since the demands of the w requests are equal, the total shipment for each request $j \in J$ must be the same as that for the first request, that is $\sum_{i \in X_j} z_{i,1} = \sum_{i \in X_1} z_{i,1}$. Equivalently, we have:

$$\sum_{i \in X_j} g(i) + |X_j|\delta = \sum_{i \in X_1} g(i) + |X_1|\delta.$$

By $\delta > \sum_{i \in X} g(i)$, we know $|X_j| = |X_1|$ and $\sum_{i \in X_j} g(i) = \sum_{i \in X_1} g(i)$, which leads $|X_j| = 3$ and $\sum_{i \in X_j} g(i) = B$ for $j = 1, ..., w$. ∎

3.4 Summary

The computational complexity results are presented in this chapter. various special cases have been studied here. Furthermore, a number of other results can be directed derived based on the reduction relations shown in Figure 2.2. For example, the case $Optimization(w = \lambda, s_i = \infty, f_i = 0)$ for any $\lambda \geq 1$ is unary NP-hard, by its reduction from $Optimization(w = 1, s_i = \infty, f_i = 0)$. Accordingly, we can generalize the computational complexity results for all special cases as follows.

1. Finding an optimal solution is unary $\mathcal{NP}$-hard unless both MQC restrictions and fixed selection costs are relaxed, i.e. $w = 0$ and $f_i = 0$.

2. Finding only a feasible solution is at least binary $\mathcal{NP}$-hard unless at most one region holds the MQC rule or the maximum capacity are relaxed, i.e. $w \leq 1$ or $s_i = \infty$, and is even unary $\mathcal{NP}$-hard when an arbitrary number of regions hold the MQC rule, i.e. w is arbitrary.

From this statement, we can see that the bidding selection and assignment problem with MQC restrictions is very hard to solve.

Algorithm 3.6 Solving $Feasibility(w = 1, s_i, *)$

if the total agents' maximum capacity is less than the total requests' demand ($S < D$) **then**
  Output "No Feasible Solution";
else
  Select all the shipping agents: $A \leftarrow \{1, 2, ..., m\}$;
  Let c_1 indicate the number of agents with at least b_1 capacities.
  Choose the $c = \min(c_1, \lfloor D_1/b_1 \rfloor)$ agents having c largest capacities s_i to be the first c agents $1, 2, ..., c$, and form the set $A_1 = \{1, ..., c\}$ to serve requests in region 1.
  Let D_0 and D_1 denote the total demands in region 0 and 1 respectively;
  if $\sum_{i \in A_1} s_i < \sum_{j \in J_1} d_j$ **then**
    Output "No Feasible Solution";
  else
    Set remnant capacities: $s'_i \leftarrow s_i$ for $i \in S$;
    Set $t_{i,r} \leftarrow 0$ for $i \in S$ and $r = 0, 1$;
    Set $t_{i,1} \leftarrow b_1$ for all $i \in A_1$ to reserve minimum quantity for region 1;
    Set the remnant demand of region 1: $D'_1 \leftarrow D_1 - cb$;
    Decrease remnant capacities: $s'_i \leftarrow s_i - b$ for $i \in A_1$;
    Let τ be the index s.t. $\sum_{i=1}^{\tau} s'_i \geq D'_1 > \sum_{i=1}^{\tau-1} s'_i$;
    for $i = 1$ to $\tau - 1$ **do**
      $t_{i,1} \leftarrow t_{i,1} + s'_i$;
      Decrease the remnant demand $D'_1 \leftarrow D'_1 - s'_i$ for region 1;
    end for
    $t_{\tau,1} \leftarrow t_{\tau,1} + D'_1$
    Decrease the remnant capacities $s'_i \leftarrow s'_i - t_{i,1}$ for $i \in A_1$;
    Set the remnant demand of region 0: $D'_0 \leftarrow D_0$;
    Let σ be the index s.t. $\sum_{i=1}^{\tau} s'_i \geq D'_0 > \sum_{i=1}^{\sigma-1} s_i$;
    for $i = 1$ to $\sigma - 1$ **do**
      $t_{i,0} \leftarrow t_{i,0} + s'_i$;
      Decrease the remnant demand $D'_0 \leftarrow D'_0 - s'_i$ for region 0;
    end for
    $t_{\sigma,0} \leftarrow t_{\sigma,0} + D'_0$
    Call Algorithm 3.4 to obtain assignments $z_{i,j}$ for $i \in A$ and $j \in J$;
    Output the set A and assignments $z_{i,j}$ for $i \in A$ and $j \in J$;
  end if
end if

Chapter 4

Optimization Approaches

4.1 Overview

As we presented in the last Chapter 3, the new MQC constraint is hard to deal with for the selection and assignment problem theoretically. But in this chapter, we are interested in how to apply optimization approaches to solve it practically. We will focus on solving a basic problem, for which only the MQC constraint and the demand constraint are considered. This basic problem is introduced in Section 4.2 and formulated as a mixed integer programming model in Section 4.3. To solve the basic problem practically, we study its polyhedral structure of the mixed integer programming model, and find a strong valid inequality to strengthen the model. Based on this mixed integer programming model, we apply a branch and cut search scheme in Section 4.5 to solve those medium size instances. For those large size instances, we invent two heuristic algorithms to solve them practically. A linear rounding heuristics is given in Section 4.6, and a greedy algorithm is analyzed in Section 4.7. For the greedy algorithm, a non-constant approximation ratio is proved to be guaranteed under some circumstances. Afterwards, their related experiments will be discussed in the next Chapter 5.

4.2 A Basic Problem

The basic problem studied in this chapter rises from the simple special case $Optimization(w = 1, s_i = \infty, f_i = 0)$ of the selection and assignment problems with MQC constraint. Recall that $Optimization(w = 1, s_i = \infty, f_i = 0)$ relaxes the agents' maximum capacities constraint, ignores their fixed selection costs , and has only one region holding MQC rules. Therefore, the request set J can be partitioned into two subsets J_0 and J_1, and we can write the total cost as the sum of two partial costs:

$$\sum_{j\in J_0}\sum_{i\in I} c_{i,j}z_{i,j} + \sum_{j\in J_1}\sum_{i\in I} c_{i,j}z_{i,j}.$$

Correspondingly, the case $Optimization(w = 1, s_i = \infty, f_i = 0)$ can be regarded as to minimize the two partial costs separately. On one hand, to minimize the cost $\sum_{j\in J_0}\sum_{i\in I_0} c_{i,j}z_{i,j}$, only the demand constraint need to be satisfied. It is easy to see that this sub-case is $Optimization(r = 0, s_i = \infty, f_i = 0)$, which can be solved efficiently according to Theorem 3.2 in the last Chapter 3. On the other hand, we need to minimize the cost $\sum_{j\in J_1}\sum_{i\in I_0} c_{i,j}z_{i,j}$. For this sub-case, both the demand constraint and MQC constraints need to be considered. Since this second sub-case is the only difficult part, we name it the Basic Problem, and will focus on it in the rest of this thesis. Thence, we may assume that J_0 is empty. Under this assumption, the Basic Problem is equivalent to the case $Optimization(w = 1, s_i = \infty, f_i = 0)$.

Formally speaking, we define the Basic Problem as follows.

Basic Problem: suppose the agent set is $I = \{1, ..., m\}$, and the request set is $J = \{1, ..., n\}$. Each request $j \in J$ has demand d_j. Suppose all requests are shipped into the only region holding MQC rule, where its minimum quantity is b. The Basic Problem is to minimize the total cost:

$$\sum_{i\in I}\sum_{j\in J} c_{i,j}z_{i,j}, \tag{4.1}$$

such that the demand is satisfied for each request $j \in J$, i.e.

$$\sum_{i\in I} z_{i,j} = d_j, \tag{4.2}$$

and that the MQC rule is satisfied for each agent $i \in I$, i.e.

$$\sum_{j \in J} z_{i,j} = 0, \text{ or } \sum_{j \in J} z_{i,j} \geq b. \tag{4.3}$$

The feasible solution of the Basic Problem satisfies both the Constraint (4.2) and Constraint (4.3), while the optimal one is not only feasible but has the minimum total cost (4.1) as well.

Since the agents' maximum capacity is relaxed to be infinity, the feasible solution of the Basic Problem is easy to obtained by the Algorithm 3.2 of Theorem 3.5 in Chapter 3. So we have the following proposition.

Proposition 4.1 *Algorithm 3.2 generates a feasible solution for the Basic Problem in $O(n)$ time.*

Then, let us consider the optimal solution. Although we restrict the request subset J_0 to be empty in the Basic Problem, finding an optimal solution for the Basic Problem is still unary $\mathcal{NP}$-hard, which can be shown by the same arguments for the case $Optimization(w = 1, s_i = \infty, f_i = 0)$ in Theorem 3.4 of the last Chapter 3.4. This establishes the following result.

Proposition 4.2 *Whenever the minimum quantity $b \geq 3$, finding an optimal solution for the Basic Problem is unary $\mathcal{NP}$-hard.*

Moreover, the Basic Problem is found too difficult to be solved even approximatively. We prove this non-approximation result as follows.

Proposition 4.3 *There exists no approximation algorithm that guarantees finite approximation ratio for the Basic Problem, unless $\mathcal{P} = \mathcal{NP}$.*

Proof We prove it by contradiction. Assume $\mathcal{P} \neq \mathcal{NP}$. Suppose there exists an approximation algorithm $\mathcal{A}$ that guarantees finite approximation ratio a for the Basic Problem. By the same arguments for Theorem 3.2, we can transform any instance of the unary $\mathcal{NP}$-complete problem $X3C$ to an instance of the Basic Problem, and insure that the instance of $X3C$ has an exact cover if and only if the minimum total cost for the instance

of the Basic Problem is zero. Since the approximation ratio a is finite and then $a \cdot 0 = 0$, we know that the algorithm $\mathcal{A}$ can generate a feasible solution with zero total cost for the instance of the Basic Problem in polynomial time, if and only if its exact minimum total cost is zero. This condition is equivalent to the existence of an exact cover for the instance of $X3C$. So $X3C$ can be solved by the algorithm $\mathcal{A}$ polynomially. This implies $X3C \in \mathcal{P}$, leading contradiction to our assumption of $\mathcal{P} \neq \mathcal{NP}$. ∎

The last two propositions demonstrate the theoretical difficulties to solve the Basic Problem efficiently. However, the rest of this chapter is interested in how to solve it in practice.

4.3 Mixed Integer Programming Model

We begin with an integer programming model of the Basic Problem by introducing a binary decision variable x_i for each agent i. We assign $x_i = 1$ if the agent i has containers to ship, otherwise $x_i = 0$. Therefore, the Basic Problem can be formulated as the following integer programming model.

Integer Programming Model (IP)

$$\min \quad \sum_{i \in I} \sum_{j \in J} c_{i,j} z_{i,j} \tag{4.4}$$

$$\text{s.t.} \quad \sum_{i \in I} z_{i,j} = d_j, \text{ for } j \in J \tag{4.5}$$

$$b x_i \leq \sum_{j \in J} z_{i,j} \leq D x_i, \text{ for } i \in I \tag{4.6}$$

$$x_i \in \{0, 1\}, \text{ for } i \in I \tag{4.7}$$

$$z_{i,j} \in \mathcal{Z}_+, \text{ for } i \in I \text{ and } j \in J. \tag{4.8}$$

where $\mathcal{Z}_+$ is the set of non-negative integers, and D is the total demands of requests, i.e. $D = \sum_{j \in J} d_j$.

In the *IP* model, the objective function (4.4) and the first constraint (4.5) are straightforward from the total cost (4.1) and the demand constraint (4.2) of the Basic Problem respectively. Moreover, by the second constraint (4.6) of the *IP* model, we have either

$\sum_{j\in J} z_{i,j} = 0$ (when $x_i = 1$), or $b \leq \sum_{j\in J} z_{i,j} \leq D$ (when $x_i = 1$), for each agent $i \in I$. Since the inequality $\sum_{j\in J} z_{i,j} \leq D$ can be independently derived from the previous constraint (4.5), the second constraint (4.6) is equivalent to the MQC constraint (4.3) of the Basic Problem. Therefore, the *IP* model describes the the Basic Problem exactly.

There are $m + mn$ integer variables in this *IP* model, including m binary x_i and mn integral $z_{i,j}$, for $i \in I$ and $j \in J$. Fortunately, we can relax every integral $z_{i,j}$ to be real variable, and then, only m binary variables x_i are restricted integral. To explain the correctness of such a relaxation for $z_{i,j}$, we need to prove that after the relaxation, the convex hull of feasible solutions for the IP model is unchanged, i.e. the vertex of the convex hull must be unchanged. So, if this relaxation is correct, we can only solve a mixed integer programming model instead of the pure integer programming one. For further concepts and theories of the integer programming, please refer the textbook [26].

In the following elaboration, we will firstly show that the relaxation of $z_{i,j}$ is correct if the values of x_i are determined, and then generalize the fact for the original IP model, which conducts a mixed integer programming model for the Basic Problem.

First, suppose the values of m binary variables x_i for $i \in I$ have been determined. Let's see what the *IP* model will become. We use S to denote the set of agents i with $x_i = 1$, where $i \in I$. Given the subset $S \subseteq I$, we can rewrite the *IP* model by replacing x_i with its value. Moreover, by the MQC constraint (4.6) and $x_i = 0$ for $i \notin S$, we know that for $j \in J$ and $i \notin S$, the assignments $z_{i,j} = 0$ can be removed from the *IP* model . This leads a new integer programming model depending on the subset S, or $IP(S)$ in short.

Integer Programming Model on S (IP(S))

$$\min \quad \sum_{i\in S}\sum_{j\in J} c_{i,j} z_{i,j} \tag{4.9}$$

$$\text{s.t.} \quad \sum_{i\in S} z_{i,j} = d_j, \text{ for } j \in J \tag{4.10}$$

$$b \leq \sum_{j\in J} z_{i,j}, \text{ for } i \in S \tag{4.11}$$

$$z_{i,j} \in \mathcal{Z}_+, \text{ for } i \in S \text{ and } j \in J. \tag{4.12}$$

Basically, the new $IP(S)$ model is a minimum cost network flow problem with lower bounds on arcs. To see this, consider a bipartite network with cross arcs between S and J. Besides nodes in S and J, there are a source node s and a demand node t. Then the whole node set of the network is $V = S \cup J \cup \{s, t\}$. Let $z_{s,i}$ denote the total number of containers shipped by agent $i \in S$, i.e.

$$z_{s,i} - \sum_{j \in J} z_{i,j} = 0,$$

and let $z_{j,t}$ indicate the total number of containers shipped for request $j \in J$, i.e.

$$\sum_{i \in S} z_{i,j} - z_{j,t} = 0.$$

Imagine $z_{u,v}$ to be the flow on arc (u, v) for $u, v \in V$, then the constraint (4.10) and (4.11) imply the lower bound or upper bound on some arcs, that is $z_{j,t} = d_j$ for $j \in J$, and $b \leq z_{i,t}$ for $i \in S$. The cost of each arc from $i \in S$ to $j \in J$ is $c_{i,j}$, otherwise is zero. This keeps the total cost of flow the same as (4.9). Therefore, the $IP(S)$ model is nothing but a minimum cost network flow problem with lower bounds on arcs. Since the latter problem can be solved in polynomial time[3], we obtain the following proposition.

Proposition 4.4 *A minimum cost network flow algorithm can solve the $IP(S)$ model in polynomial time.*

Now, let us consider the relaxation of integral variables $z_{i,j}$ to real for the $IP(S)$ model. Since the $IP(S)$ is a minimum cost network flow problem with lower bounds on arcs, the following integrity property holds true.

Lemma 4.5 *In the $IP(S)$ model, if the minimum quantity b and the demands d_j for $j \in J$ are integral, constraints (4.10)–(4.11) and $z_{i,j} \in R_+$ (for $i \in S$ and $j \in J$) describe the convex hull of the $IP(S)$, where R_+ is the set of non-negative real numbers.*

Proof According to the same arguments for the integrity property of the minimum cost network flow problem presented in [26], we know that the above result is true for the $IP(S)$ model. [26]. ∎

Lemma 4.5 insures that the relaxation of integral variables $z_{i,j}$ to real will keep the convex hull of the feasible solutions in the $IP(S)$ unchanged. So, even if we relaxing $z_{i,j}$ to real, the convex hull of the relaxed $IP(S)$ model still consists of only integral vertex. Since the $IP(S)$ is derived from the original IP model by assigning the values of only x_i for $i \in I$, the integrity property of $z_{i,j}$ is still held for the IP model. This implies the following theorem.

Theorem 4.6 *In the IP model, if the minimum quantity b and the demands d_j for $j \in J$ are integral, the constraints (4.5)–(4.7) and $z_{i,j} \in R_+$ (for $i \in I$ and $j \in J$) describe the convex hull of the IP.*

Proof We prove it by contradiction. Suppose the statement is false. Then, on the convex hull of the relaxed IP model, where $z_{i,j} \in R_+$ for $i \in S$ and $j \in J$, there exists a vertex with integral x_i and fractional $z_{i,j}$ for $i \in I$ and $j \in J$. Let $S = \{i : x_i = 1, i \in I\}$. We have that the point $\{z_{i,j}\}$ is a fractional vertex on the convex of the relaxed $IP(S)$ model, leading contradiction to Lemma 4.5. ∎

We have shown the correctness of relaxing integral variables $z_{i,j}$ to real for the IP model. Therefore, we can obtain the mixed integer programming model for the Basic Problem as follows.

Mixed Integer Programming Model (MIP)

$$\min \quad \sum_{i \in I} \sum_{j \in J} c_{i,j} z_{i,j} \tag{4.13}$$

$$\text{s.t.} \quad \sum_{i \in I} z_{i,j} = d_j, \text{ for } j \in J \tag{4.14}$$

$$b x_i \leq \sum_{j \in J} z_{i,j}, \text{ for } i \in I \tag{4.15}$$

$$\sum_{j \in J} z_{i,j} \leq D x_i, \text{ for } i \in I \tag{4.16}$$

$$x_i \in \{0, 1\}, \text{ for } i \in I \tag{4.17}$$

$$z_{i,j} \geq 0, \text{ for } i \in I \text{ and } j \in J. \tag{4.18}$$

In the *MIP* model, we separate the MQC constraint into two inequalities (4.15) and (4.16), for (4.16) will be tightened in the later Section 4.4.

Compared with the *IP* model that has $n + mn$ integral variables, the *MIP* has m only and is expected to be easier for solving. Moreover, although a feasible solution to the *MIP* model might have fractional assignments $z_{i,j}$ for $i \in I$ and $j \in J$, the values of x_i are integral for $i \in I$. Based on x_i, we can solve a corresponding *IP(S)* model, by Proposition 4.4, to get an integral feasible solution of the *IP* model with even better objective value. For this reason, the only necessary work on solving the Basic Problem is to solve its *MIP* model.

4.4 Strengthening the Model

We propose to solve the *MIP* model by branch and cut search scheme and other heuristics algorithms. The performance for most of these methods depends on the tightness of the model. In this section, we will examine the tightness of the *MIP* model, and explore ways to strengthen it as well.

To measure the tightness of a model, we adopt the concept of *facet* [26], which is effective in the analysis of the models for selection and assignment problems [11][17]. Intuitively speaking, facets are the necessary inequalities that determines the convex hull of the feasible set for an integer (or mixed integer) programming model. The convex hull forms a polyhedron, and its boundaries are defined by the facets. So, the more facets we have, the stronger the model we defined is. If we had all of them, the integrity condition could be totally relaxed and turns to a linear programming model. However, finding all the facets for the integer (or mixed integer) programming model is intractable, although the number of them is finite [26].

Now, let's examine the tightness of the *MIP* model to see whether all the inequalities are facets. The following theorem shows that our expectation is almost true, except the inequalities (4.16) of the MQC constraint.

Theorem 4.7 *If $2b < D$, for the convex hull of the MIP model's feasible set,*

1. *its dimension is $mn + m - n$, and*

2. *the inequalities (4.15), $x_i \leq 1$ for $i \in I$, (4.18), and equality (4.14) are facet-defining.*

Proof Suppose $2b < D$ is satisfied. Here, we present the proof of the facet only for the most difficult case (4.15). For other three cases, we can follow similar arguments, described in Appendix A.1, to prove that they are facets as well. Besides that, the dimension of the convex hull can also be derived by the end of the following proof.

For any agent $p \in I$, we are going to show the inequality (4.15) for the agent p, i.e.

$$bx_p \leq \sum_{j \in J} z_{p,j}, \tag{4.19}$$

is a facet. To see this, we must prove that if all feasible solutions of the *MIP* model that satisfy (4.19) at equality also satisfy

$$\sum_{i \in I} \sum_{j \in J} \alpha_{i,j} z_{i,j} + \sum_{i \in I} \beta_i x_i \leq \theta \tag{4.20}$$

at equality, then (4.20) must be a linear combination of (4.19) and the equality constraint (4.14), which implies the inequality (4.19) is necessary, or facet-defining, for the *MIP* model.

Firstly, let u and v denote any two different agents in I other than p, and we now show $\alpha_{u,j} = \alpha_{v,j}$ for all $j \in J$. Let us construct a feasible solution (x^1, z^1) as follows. Let $x^1_u = x^1_v = 1$ and other $x^1_i = 0$ for $i \in I - \{u, v\}$. Since $2b < D$, there exist a feasible assignment $z^1_{i,j}$ for $i \in I$ and $j \in J$, such that $z^1_{u,j} > 0$ and $z^1_{v,j} > 0$ for all $j \in J$, and that $\sum_{j \in J} z^1_{u,j} > b$ and $\sum_{j \in J} z^1_{v,j} > b$ as well. Since $x_p = 0$, we know (x^1, z^1) satisfies (4.19) at equality. Consider another feasible solution (x^2, z^2), where $x^2_i = x^1_i$ for all $i \in I$, and $z^2_{i,j} = z^1_{i,j}$ for all $i \in I - \{u, v\}$ and $j \in J$, but

$$z^2_{u,j} = z^1_{u,j} + \epsilon \text{ and } z^2_{v,j} = z^1_{v,j} - \epsilon, \text{ for } j \in J,$$

where $\epsilon > 0$ is arbitrary close to zero. It is easy to see that by choosing proper small ϵ we can keep (x^2, z^2) feasible and satisfying (4.19) at equality. Substituting (x^1, z^1) and (x^2, z^2) into (4.20) and subtracting one from the other results in $\alpha_{u,j} = \alpha_{v,j}$. So, we can

assume $\alpha_{u,j} = \alpha_j$ for each agent $u \in I - \{p\}$, and rewrite (4.20) as

$$\sum_{j \in J} \alpha_j \sum_{i \in I} z_{i,j} + \sum_{j \in J} (\alpha_{p,j} - \alpha_j) z_{p,j} + \sum_{i \in I} \beta_i x_i \leq \theta. \tag{4.21}$$

Secondly, we will show $\beta_u = 0$ for any $u \in I - \{p\}$. We construct another feasible solution (x^3, z^3) from (x^1, z^1) as follows. Let $x_i^3 = x_i^1$ for all $i \in I - \{u\}$, but $x_u^3 = 0$. Since only $x_v^3 = 1$, we assign $z_{v,j} = d_j$ for all $j \in J$. It is easy to see that (x^1, z^1) is feasible and satisfies (4.19) at equality. Note

$$\sum_{j \in J} \alpha_j \sum_{i \in I} z_{i,j} = \sum_{j \in J} \alpha_j d_j \tag{4.22}$$

and $z_{p,j} = 0$ for $j \in J$. Substituting (x^1, z^1) and (x^3, z^3) into (4.21) at equality and subtracting one from the other results in $\beta_u = 0$, for $u \in I - \{p\}$. We can rewrite (4.21) as

$$\sum_{j \in J} \alpha_j \sum_{i \in I} z_{i,j} + \sum_{j \in J} (\alpha_{p,j} - \alpha_j) z_{p,j} + \beta_p x_p \leq \theta. \tag{4.23}$$

Thirdly, we will prove $(\alpha_{p,g} - \alpha_g) = (\alpha_{p,h} - \alpha_h)$, for any two different requests $g, h \in J$. We construct a new feasible solution (x^4, z^4) by setting $x_p^4 = x_u^4 = 1$ and other $x_i^4 = 0$ for $i \in I - \{p, u\}$. Since $2b < D$, there exists a feasible assignment $z_{i,j}^4$ for $i \in I$ and $j \in J$, such that $z_{p,j}^4 > 0$ and $z_{u,j}^4 > 0$ for $j \in J$, and that (4.19) is satisfied at equality. Consider another feasible solution (x^5, z^5), where $x_i^5 = x_i^4$ for $i \in I$, and $z_{i,j}^5 = z_{i,j}^4$ for $i \in I - \{p, u\}$ or $j \in J - \{g, h\}$, but

$$\begin{aligned} z_{p,g}^5 &= z_{p,g}^4 + \epsilon, z_{p,h}^5 = z_{p,h}^4 - \epsilon, \\ z_{u,g}^5 &= z_{u,g}^4 - \epsilon, z_{u,h}^5 = z_{u,h}^4 + \epsilon. \end{aligned}$$

It is easy to verify that (x^5, z^5) keeps feasible and satisfying (4.19) at equality. Again, note the equality (4.22). Substituting (x^4, z^4) and (x^5, z^5) into (4.23) at equality and subtracting one from the other results in $(\alpha_{p,g} - \alpha_g) = (\alpha_{p,h} - \alpha_h)$. Assuming $(\alpha_{p,g} - \alpha_g) = \lambda_p$ for each $g \in J$, we can rewrite (4.23) as

$$\sum_{j \in J} \alpha_j \sum_{i \in I} z_{i,j} + \lambda_p \sum_{j \in J} z_{p,j} + \beta_p x_p \leq \theta. \tag{4.24}$$

Finally, because of (4.22) and $\sum_{j\in J} z^4_{p,j} = b$, substituting (x^1, z^1) and (x^4, z^4) into (4.24) at equality and subtracting one from the other results in $\beta_p = -\lambda_p b$. For the same reason, by substituting (x^1, z^1) into (4.24) only, we have $\theta = \sum_{j\in J} \alpha_j d_j$. Therefore, we can rewrite (4.24) as

$$\sum_{j\in J} \alpha_j \sum_{i\in I} z_{i,j} + \lambda_p \sum_{j\in J} z_{p,j} \leq \sum_{j\in J} \alpha_j d_j + \lambda_p b x_p, \tag{4.25}$$

which is a linear combination of (4.19) and the equality constraint (4.14).

By the end of the proof, let us consider the dimension of the convex hull. In (4.20), there are $mn+m+1$ unknown parameters $\alpha_{i,j}$, β_i and θ for $i \in I$ and $j \in J$. Along with the *MIP* model and (4.19) at equality, we solve these equations in (4.20) and obtain (4.25). Since only $n+1$ unknown ones are left in (4.25), noting that n independent equations are in 4.14, we know that the maximum number of affinely independent feasible solutions in the facet is exactly $mn+m-n$. This implies the dimension of the facet is $mn+m-n-1$, which is one less than that of the whole convex hull. So the dimension of the convex hull is $mn + m - n$. ∎

We have shown that if $2b < D$, most constraints of the *MIP* model are facet-defining, except the (4.16). Actually, inequalities in (4.16) are not facets, because if (4.16) is satisfied at equality for an agent $p \in I$, there is only a unique feasible solution, in which $x_p = 1, z_{p,j} = d_j$ for all $j \in J$ and others variables are zero.

To strengthen the model, we reformulate (4.16) as

$$z_{i,j} \leq d_j x_i, \text{ for } i \in I \text{ and } j \in J, \tag{4.26}$$

which is a family of facets of the *MIP* model if $2b < D$. This establishes the following theorem.

Theorem 4.8 *If $2b < D$, the inequalities in (4.26) are facet-defining for the convex hull of the MIP model's feasible set.*

Proof Suppose $2b < D$ is satisfied. For any agent $p \in I$ and any request $q \in J$, we are going to show that the inequality $z_{p,q} \leq d_q x_p$ is a facet. To see this, we must prove that

if all feasible solutions of the *MIP* model that satisfy $z_{p,q} = d_q x_p$ also satisfy (4.20) at equal, then (4.20) must be a linear combination of $z_{p,q} = d_q x_p$ and the equalities (4.14).

In the same way as the first and second steps of the proof for the case (4.15) in Theorem 4.7, we can show $\alpha_{u,j} = \alpha_{v,j} = \alpha_j$ and $\beta_u = 0$ for $u \in I - \{p\}$, $v \in I - \{p, u\}$ and $j \in J$. So (4.20) can be rewritten as (4.23).

Afterwards, we will show $(\alpha_{p,g} - \alpha_g) = 0$ for any request g in J but other than p. Similarly to the proof in Theorem 4.7, we construct a new feasible solution $(\widetilde{x}^4, \widetilde{z}^4)$ as follows. We still let $\widetilde{x}^4_p = \widetilde{x}^4_u = 1$ and other $\widetilde{x}^4_i = 0$ for $i \in I - \{p, u\}$. For the feasible assignments, we still require $\widetilde{z}^4_{p,j} > 0$ and $\widetilde{z}^4_{u,j} > 0$ for $j \in J - \{q\}$, but $\widetilde{z}^4_{p,q} = d_q$ and $\widetilde{z}^4_{u,q} = 0$ in addition. Since $2b < D$, it is easy to see such a feasible assignment exist. Now let us consider another feasible solution $(\widetilde{x}^5, \widetilde{z}^5)$, where $\widetilde{x}^5_i = \widetilde{x}^4_i$ for $i \in I$, and $\widetilde{z}^5_{i,j} = \widetilde{z}^4_{i,j}$ for $i \in I - \{p, u\}$ or $j \in J - \{p\}$, but we let

$$\widetilde{z}^5_{p,g} = \widetilde{z}^4_{p,g} + \epsilon,\ \widetilde{z}^5_{u,g} = \widetilde{z}^4_{u,g} - \epsilon.$$

By choosing sufficient small positive ϵ, we can keep $(\widetilde{x}^5, \widetilde{z}^5)$ feasible and satisfying $z_{p,q} = d_q$. By (4.22), substituting $(\widetilde{x}^4, \widetilde{z}^4)$ and $(\widetilde{x}^5, \widetilde{z}^5)$ into (4.23) at equality and subtracting one from the other results in $(\alpha_{p,g} - \alpha_g) = 0$ for any $g \in J - \{q\}$. Therefore, we can rewrite (4.23) as

$$\sum_{j \in J} \alpha_j \sum_{i \in I} z_{i,j} + (\alpha_{p,q} - \alpha_q) z_{p,q} + \beta_p x_p \leq \theta. \tag{4.27}$$

Finally, recall that (x^1, z^1) is a feasible solution constructed in the proof of Theorem 4.7, and $x^1_p = 0$ and $z^1_{p,j} = 0$ for $j \in J$. Because of (4.22) and $\widetilde{z}^4_{p,q} = d_j \widetilde{x}^4_p$, substituting (x^1, z^1) and $(\widetilde{x}^4, \widetilde{z}^4)$ into (4.24) at equality and subtracting one from the other results in $\beta_p = (\alpha_q - \alpha_{p,q}) d_q$. For the same reason, by substituting (x^1, z^1) into (4.24) only, we have $\theta = \sum_{j \in J} \alpha_j d_j$. Therefore, we can rewrite (4.24) as

$$\sum_{j \in J} \alpha_j \sum_{i \in I} z_{i,j} + (\alpha_{p,q} - \alpha_q) z_{p,q} \leq \sum_{j \in J} \alpha_j d_j + (\alpha_{p,q} - \alpha_q) d_q x_p, \tag{4.28}$$

which is a linear combination of $z_{p,q} = d_q z_p$ and the equality constraint (4.14). ∎

Moreover, it is easy to see that the old inequalities in (4.16) are redundant, because they can be derived by summing up some new inequalities in (4.26). So, we can replace

(4.16) with (4.26) in the *MIP*, and have the following strengthened mixed integer programming.

Strengthened Mixed Integer Programming Model (S-MIP)

$$\min \quad \sum_{i\in I}\sum_{j\in J} c_{i,j}z_{i,j} \tag{4.29}$$

$$\text{s.t.} \quad \sum_{i\in I} z_{i,j} = d_j, \text{ for } j \in J \tag{4.30}$$

$$bx_i \leq \sum_{j\in J} z_{i,j}, \text{ for } i \in I \tag{4.31}$$

$$z_{i,j} \leq d_j x_i, \text{ for } i \in I \text{ and } j \in J, \tag{4.32}$$

$$x_i \in \{0,1\}, \text{ for } i \in I \tag{4.33}$$

$$z_{i,j} \geq 0. \text{ for } i \in I \text{ and } j \in J \tag{4.34}$$

Before we end this section, let us make some comments on the condition $2b < D$ of facets. Intuitively speaking, if the minimum quantity b is close to the total demand D, only one or two agents can be selected to have $x_i = 1$ at the same time, and the Basic Problem will become very simple. For instance, if $2b > D$, then at most one agent can have shipments, and only m feasible solutions exist to be explored. Therefore, except for a few simple instances, the condition $2b < D$ appears trivial for the Basic Problem.

4.5 Branch and Cut

Given a mixed integer programming model, the traditional way to solve it is branch and cut search scheme [26]. In this section, we will illustrate the basic idea of applying this search method to solve the Basic Problem. Along with the illustration, we will point out that the tightness of the formulated model is important to the performance of the search process.

We use the *S-MIP* model as an example to illustrate the outline of the branch and cut algorithm, and begin with a simple exhaustive search scheme as follows. In *S-MIP*, there are m binary variable x_i for $i \in I$ to be determined. Remember that if $\{x_i\}$ are determined, we can let $S = \{i \in I : x_i = 1\}$ and solve the $IP(S)$ model to generate

its optimum assignment $z_{i,j}$ for $i \in I$ and $j \in J$. Therefore, simply enumerating all the possible values of $\{x_i\}$ is enough for us to find the optimum solution to the *S-MIP* model. However, such a simple exhaustive search scheme can not work in practise, because it always explores almost the whole search space $\{0,1\}^m$ whose size is 2^m exponentially. Even though by summing up (4.31) we have $x_1 + x_2 + ... + x_m \leq w$ where $w = \lfloor D/b \rfloor$, the size of search space is reduced to $C_1^m + C_2^m + ... + C_w^m$ but is still exponentially large.

To improve the efficiency, we need to prune the invalid branch during the search process. This leads a following branch and bound method. Let (x^*, z^*) denote the current best feasible solution. Suppose we have assigned values to some x_i, and accordingly, let Π_0 (or Π_1) indicate the set of x_i determined to be zero (or one). So we can use a duplet (Π_0, Π_1) to represent a partial determined $\{x_i\}$. Before exploring the rest undetermined x_i further, let us make a lower estimation of the objective cost that we could at least have. Unless the lower estimation is less than the current minimum cost of (x^*, z^*), we don't need to assign the values to the rest x_i further for $i \in I - (\Pi_0 \cup \Pi_1)$. To make such a lower estimation, we can keep those values of determined x_i for $i \in \Pi_0 \cup \Pi_1$, and relax the rest undetermined x_i from binary to the closed interval $[0, 1]$. This changes the *S-MIP* model to a relaxed linear programming model, i.e.

Linear Programming Model on (Π_0, Π_1) **by relaxing the *S-MIP* (LP**(Π_0, Π_1)**)**

$$\min \quad \sum_{i \in I} \sum_{j \in J} c_{i,j} z_{i,j} \tag{4.35}$$

$$\text{s.t.} \quad \sum_{i \in I} z_{i,j} = d_j, \text{ for } j \in J \tag{4.36}$$

$$b x_i \leq \sum_{j \in J} z_{i,j}, \text{ for } i \in I \tag{4.37}$$

$$z_{i,j} \leq d_j x_i, \text{ for } i \in I \text{ and } j \in J, \tag{4.38}$$

$$x_i = 0, \text{ for } i \in \Pi_0 \tag{4.39}$$

$$x_i = 1, \text{ for } i \in \Pi_1 \tag{4.40}$$

$$x_i \in [0,1], \text{ for } i \in I - (\Pi_0 \cup \Pi_1) \tag{4.41}$$

$$z_{i,j} \geq 0. \text{ for } i \in I \text{ and } j \in J, \tag{4.42}$$

whose optimum solution, denote by (x', z'), gives an lower estimation of the further exploration. If the cost of (x', z') is not less than the cost of the current best solution (x^*, z^*), no better solution might appear in further exploration for (Π_0, Π_1). So we stop exploring its rest undetermined x_i for $i \in I - (\Pi_0 \cup \Pi_1)$, and turn to other unexplored duplets (Π_0', Π_1'). Otherwise, the cost of (x', z') is better. Let us try different values for an undetermined x_i, where $i \in I - (\Pi_0 \cup \Pi_1)$, and generate new duplets for further exploration. This pruning technique reduces the size of space needed to be explored, and then improves the performance of the search scheme.

Obviously, the accuracy of the lower estimation plays an important role in the branch and bound algorithm. If the estimation is close to the actual value of the best solution in further exploration, most invalid branches will be cut and only a few steps of searching is needed for us to find the optimum. This explains the reason for strengthening our mixed integer programming to be tighter (or closer) to its relaxed linear programming model.

To enhance the tightness further, we have the branch and cut algorithm. This algorithm is based on the branch and bound method, but adds new constraints to strengthen the model during the search process. Remember that during the process of branch and bound, we have obtained a solution (x', z') for the relaxed linear programming model $LP(\Pi_0, \Pi_1)$. If x' is integral, then (x', z') is a feasible solution of the *S-MIP* model, and must be exactly the best solution that can be found in further exploration. However, we might not be always fortunate to have integral x'. So in most of the time, we have fractional x', and the solution (x', z') is infeasible to the *S-MIP* model. In this case, an intuitive way to strengthen the *S-MIP* model is to add a constraint, or cutting plane, with which all the feasible solution are satisfied but (x', z') is not. So the infeasible (x', z') is now excluded out of the relaxed linear programming model, and the mixed integer programming model becomes tighter.

Techniques of cutting plane are proved to be effective in the search scheme, and have been extensively studied in literature, like gormory's fractional cutting plane, mixed integer cuts, etc [26]. A detailed survey of the branch and cut method is presented in [1]. Our Algorithm 4.1 summarizes the outline of solving the *S-MIP* model through branch

and cut.

Moreover, in the next Chapter 5, we will measure the performance of the branch

Algorithm 4.1 Branch and Cut Search Scheme to Solve the Basic Problem

List L stores duplets (Π_0, Π_1) representing those partial determined $\{x_i\}$ for further exploration. Suppose we explore $\{x_i\}$ on the increasing order of its index i, therefore, we have $\Pi_0 \cup \Pi_1 = \{x_1, x_2, ..., x_t\}$ where $t = |\Pi_0 \cup \Pi_1|$ indicating the number of determined x_i.
Initially, L contains only one element $(\emptyset, \emptyset)$, implying that no x_i has been decided;
Let (x^*, z^*) denote the current optimum solution found for the *S-MIP* model.
Initially, (x^*, z^*) is empty but has infinite cost;
while the list L is not empty **do**
  From L, extract any one duplet (Π_0, Π_1), for which suppose $x_1, x_2, ..., x_t$ has been determined, i.e. $\Pi_0 \cup \Pi_1 = \{x_1, x_2, ..., x_t\}$.
  Solve the relaxed linear programming model $LP(\Pi_0, \Pi_1)$, whose optimum solution is denoted by (x', z');
  if the cost of (x', z') is less than the cost of current best (x^*, z^*) **then**
    if x' is integral **then**
      The solution $\{x', z')$ is feasible.
      Update the current optimum solution by replacing (x^*, z^*) with (x', z');
    else
      Add a proper cut plane to the *S-MIP* model, such that the infeasible solution (x', z') is excluded.
    end if
    Try to assign zero to the undetermined x_{t+1}, and
    add a new duplet $(\Pi_0 \cup \{x_{t+1}\}, \Pi_1)$ to the list L;
    Try to assign one to the undetermined x_{t+1}, and
    add a new duplet $(\Pi_0, \Pi_1 \cup \{x_{t+1}\})$ to the list L;
  end if
end while
Return the optimum solution (x^*, z^*).

and cut algorithm and examine the effectiveness of the facet we add in the *S-MIP* model. The experimental results show that the branch and cut algorithm works well for small or some medium instances, and the facet we add has significantly improved the searching performance.

4.6 Linear Programming Rounding Heuristics

From Section 4.2 we know that there is no efficient algorithm that can generate exactly optimal solution or only approximative one for the Basic problem, although we have an

exact branch and cut search scheme for the small or some medium problem instances yet.

To solve those large instances practically for the Basic Problem, we invent two heuristics algorithm. One is a combinatorial greedy method, which will be studied in the next Section 4.7. The other is a linear programming rounding heuristic, or LP rounding heuristic in short, which is as follows.

Similarly to the branch and bound algorithm, the LP rounding heuristic employs the optimum solution (x', z') of the relaxed linear programming model $LP(\Pi_0, \Pi_1)$ also. But this time, the fractional solution (x', z') is used not only for stopping invalid explorations, but for rounding fractional x'_i to integral as well. The basic idea is whenever having a fractional solution (x', z') for some $LP(\Pi_0, \Pi_1)$, we can round some x'_i with big fractional values to one, and others to zero. By solving a corresponding $IP(S)$ model, we will get its best assignments, and then have a feasible solution whose objective cost is expected to near that of the fractional (x', z'). However, it is difficult to decide how many x_i should be rounded to one. For this reason, we round x_i one by one, until we believe that no better solution might exist in further rounding. To make such a judgement, after each rounding, we need to make a lower estimation for further exploration by solving a new $LP(\Pi_0, \Pi_1)$ to .

Algorithm 4.2 describe the LP rounding heuristic. We let (x^*, z^*) denote the current best feasible solution found for the *S-MIP* model, and let t indicate the number of x_i determined. Since no x_i will be determined to be zero, the set Π_0 is always empty. So t is nothing but the size of Π_1. Then for each rounding, we solve the relaxed linear programming model $LP(\emptyset, \Pi_1)$ first, obtaining its optimum fractional solution (x', z'). The cost of (x', z') gives an lower estimation for further exploration. If it is not less than the current minimum cost of (x^*, z^*), we stop the algorithm and return (x^*, z^*) as the near-optimum solution. Otherwise, the undetermined x_i who has the largest fractional value x'_i will be assigned to one. Therefore, we have a new $LP(\emptyset, \Pi_1 \cup \{i\})$ model for next rounding. By solving the $IP(S)$ where $S = \Pi_1 \cup \{i\}$, we can have the best assignment as well, and get a feasible solution $(\widetilde{x}, \widetilde{z})$. If its cost is better than that of the current best

(x^*, z^*), we should replace (x^*, z^*) by $(\widetilde{x}, \widetilde{z})$.

Obviously, the performance of LP rounding heuristics depends on the accuracy of the lower estimation generated by the $LP(\emptyset, \Pi_1)$ model, since if the lower estimation is closer to the exactly optimum value, the fractional solution will be nearer to integral one, and the cost increased by rounding fractional to integral will be smaller. This again stresses the importance of our efforts on strengthening the mixed integer programming model in Section 4.4. Moreover, the experiments in the next Chapter 5 shows that our LP rounding heuristics performs well in both time consuming and solution generation, and exhibit stable behavior as well.

Algorithm 4.2 LP Rounding Heuristic to Solve the Basic Problem

1: Let (x^*, z^*) be the current best feasible solution founded. Initially, (x^*, z^*) is empty with infinite cost.
2: Let t indicate the number of x_i determined. Initially, $t \leftarrow 0$.
3: Let Π_1 denote the set of x_i determined to be one. Since no x_i will be determined to be zero along the following iterations, we have $t = |\Pi_1|$. Initially, $\Pi_1 \leftarrow \emptyset$.
4: **while** $t < m$ **do**
5: Solve the relaxed linear programming model $LP(\emptyset, \Pi_1)$ and obtain its optimum fractional solution (x', z');
6: **if** the cost of (x', z') is NOT less than that of the current best (x^*, z^*) **then**
7: There is no need to explore further. Stop iteration and goto 19;
8: **else**
9: Select an x'_k whose fractional value is largest among x'_i for $i \in I - \Pi_1$;
10: Round x'_k to one by $S \leftarrow \Pi_1 \cup \{k\}$;
11: Solve the $IP(S)$ model, obtaining its best feasible assignments $\widetilde{z}$;
12: Set $\widetilde{x}_i \leftarrow 1$ for $i \in S$, and $\widetilde{x}_i \leftarrow 0$ otherwise;
13: **if** the cost of $(\widetilde{x}, \widetilde{z})$ is less than that of the current best (x^*, z^*) **then**
14: Replace $(x^*, z^*) \leftarrow (\widetilde{x}, \widetilde{z})$.
15: **end if**
16: $t \leftarrow t + 1$. By $\Pi_1 \leftarrow S$, we have the new $LP(\emptyset, \Pi_1)$ model for next iteration;
17: **end if**
18: **end while**
19: Return (x^*, z^*) as the near-optimum.

4.7 Greedy Approximation Heuristics

Let us change our approach from mathematical programming to the combinatorial manner. We turn back to the original definition of the Basic problem, which is to find

assignments $z_{i,j}$ for each agent $i \in I$ and each request $j \in J$ so that the total cost (2.1) is minimized under the demand constraint (2.2) and the MQC constraint (2.4). We will design a greedy algorithm to solve the Basic Problem, such that a theoretical performance can be guaranteed under some circumstances.

Before we illustrate the greedy algorithm, let us make an assumption that the demand d_j of each request $j \in J$ is unit, because otherwise, if $d_j > 1$ for some $j \in J$, we can split the request j to d_j requests each of which has unit demand. Later, we will prove that such a splitting will not invalidate the polynomial time complexity of our implementation for the greedy method.

Since the demands are assumed unit, the request set J can represent the containers set. Let C denote the set of containers that have been assigned yet, and Π denote the set of agents who have been selected to ship containers. The idea of our greedy method is as follows.

Firstly, we define the following two elemental operations for the Basic Problem.

1. For each unselected agent $i \in I-\Pi$, we define an operation *selection(i)*, which selects the agent i to ship containers. To satisfy the minimum quantity commitment for the new selected agent i, we assign it b containers whose transportation costs are the b cheapest $c_{i,j}$ for $j \in J - C$.

2. For each selected agent $i \in \Pi$, we define an operation *assignment(i)*, which assigns the agent i a new unassigned container j that minimizes the cost $c_{i,j}$ for $j \in J - C$.

For each operation, we measure its cost by the average assignment cost. Accordingly, for *selection(i)*, its cost is $(\sum_{j \in A} c_{i,j})/b$, where A is the set of b containers assigned to the agent i. For *assignment(i)*, its cost is $c_{i,j}$, where j is the container assigned to the agent i.

Then, a feasible solution can be constructed by a series of the two elemental operations. In our greedy scheme, we do the operation that has the minimum cost iteratively, until all the containers in J have been assigned. We formalize the greedy scheme in Algorithm 4.3.

To measure the performance of the solution generated, let us look at the approxima-

Algorithm 4.3 Greedy Method to Solve the Basic Problem

1: Suppose all demands d_j are unit for $j \in J$, since otherwise, if some $d_j > 1$, we can split the request j into d_j request each with unit demand.
2: Initially, the selected agent set Π is empty. So is the assigned container set C;
3: **while** NOT all containers have been assigned, i.e. $C \neq J$ **do**
4: Choose an operation σ with minimum cost among all *selection(i)* for $i \in I - \Pi$ and *assignment(i)* for $i \in \Pi$;
5: **if** σ is *selection(i)* **then**
6: Select agent i by $\Pi \leftarrow \Pi \cup \{i\}$;
7: Let A denote the set of b containers whose transportation costs are the b cheapest $c_{i,j}$ for $j \in J - C$.
8: For each container $j \in A$, assign it to agent i by $z_{i,j} \leftarrow 1$ and $C \leftarrow C \cup \{j\}$;
9: **else if** σ is *assignment(i)* **then**
10: Let j denote the unassigned container that minimizes the transportation cost $c_{i,j}$ for $j \in J - C$;
11: Assign the container j to the agent i by $z_{i,j} \leftarrow 1$ and $C \leftarrow C \cup \{j\}$;
12: **end if**
13: **end while**
14: Return $z_{i,j}$ as the near-optimum solution.

tion ratio between the cost of generated solution and that of the optimum. As we proved in Proposition 4.3, no efficient algorithm can guarantee finite approximation ratio for the Basic Problem. But in the following case, our greedy algorithm can.

We consider a metric version of the Basic Problem, where the transportation cost are nonnegative, symmetric, and satisfy the triangle inequality. For instance, suppose all the shipping agent and containers locate at the same place. So the transportation cost of shipping a container j to its destination by the agent i may linearly depends on the distance between the source and destination of j, and therefore, the transportation cost forms a metric.

The metric version of selection and assignment problems is widely studied for the facility location problem, the p-median problem, and etc [19]. For solving the metric version of the Basic Problem, we know that our greedy method have a non-constant approximation ratio, by the following theorem.

Theorem 4.9 *For the metric version of the Basic Problem, Algorithm 4.3 generates a feasible solution whose cost is at most* $2b$ *times the optimum.*

Proof The proof is relatively complicated, so we present it in Appendix A.2.

Theorem 4.9 implies that in the metric case, the approximation ratio of the greedy algorithm is at most $2b$. To see that $2b$ is tight for our greedy algorithm, consider the following instance whose greedy cost is $b-1$ times the optimum.

Suppose the minimum quantity is b. We have $2b$ containers to ship and two agents to select. The agent 1 can ship each of the first $b+1$ containers for free, but of the rest $b-1$ ones with c cost. On the contrary, the agent 2 can ship each of the first $b+1$ containers with c cost, but of the rest $b-1$ ones for free. It is easy to verify the metric satisfaction of this instance.

By applying our greedy algorithm on this instance, we first do $select(1)$ and assign it containers $1, 2, ..., b$ for its zero average cost. Secondly, we will do $assign(1)$ that assigns the container $b+1$ to the selected agent 1 with zero cost. Afterwards, since no other operation can be done, we will do $assign(1)$ for j times while each cost is c, so that the containers $b+2, ..., 2b$ are assigned to the agent 1. Totally, the greedy method generate a feasible solution with cost $(b-1)c$. However, it is easy to see that the optimum solution is to assign the first b containers to agent 1 freely and the rest b to agent i with cost c only. So the greedy solution is $b-1$ times the optimum, which proves the following result.

Theorem 4.10 *The approximation ratio of Algorithm 4.3 is at least* $b-1$ *for the metric version of the Basic Problem.*

Finally, let us discuss the time complexity of the greedy Algorithm 4.3. When d_j is unit for $j \in J$, the number of containers to be assigned is n, so the number of iterations is at most n. During each iteration, exactly m operations will be considered. The cost of each operation can be computed in $O(b)$ times, since we can list the containers $j \in J$ on the non-decreasing order of $c_{i,j}$ for each agent $i \in I$ respectively, and then compute the operation cost in the following way. For each unselected agent $i \in \Pi$, the first b unassigned containers in its container list contribute to the cost of $selection(i)$. On the other hand, for each selected agent $i \in I - \Pi$, only the first unassigned container in its

container list contributes to the cost of $assignment(i)$. Moreover, since those assigned agents will be deleted from the list when met again, totally there are n deletions for each list. Amortizedly speaking, computing the cost of each operation takes at most $O(b+n/n)$, i.e. $O(b)$ time, in average. Since we may assume b is at most n otherwise no feasible solution exists, we have that the time complexity of Algorithm 4.3 is $O(nmb)$, or $O(n^2m)$, which is polynomial to the length of instance.

In a general case where d_j is not unit for some $j \in J$, the number of containers equals to the total demand D, which leads the time complexity to $O(Dmb)$. Since $O(Dmb)$ might be exponentially to the length of the instance, we have to improve our implementation in a more careful manner to keep the polynomial time complexity for this general case.

The improvement is based on the fact that those containers of the same request $j \in J$ have the same transportation cost for all $i \in I$. Therefore, if $assignment(i)$ is the current operation that assigns a container of request j to the agent i, then the same operation will be kept doing until all the rest containers of j are assigned to i. For this reason, we can assign the rest containers of request j together to the agent i in one iteration. In this way, at most m operations of $selection(i)$ and n operations of $assignment(i)$ will be done, so the number of iteration is at most $m+n$. For the same reason, to compute the cost of each operation, we only need to know the number of unassigned containers left for every request in its list. Since the length of the list is at most n, the cost of each operation can be computed in $O(n)$ time. Since there m operations considered for each iteration, the total time complexity is reduced to $O((n+m)mn)$, or $O(n^2m+nm^2)$, which keeps polynomial for the general case. This proves the following theorem.

Theorem 4.11 *The greedy Algorithm 4.3 can be implemented in a careful way so that its time complexity is polynomial to the length of instance, that is $O(n^2m)$ for the unit demand case and $O(n^2m+nm^2)$ for the general case.*

Although the greedy Algorithm 4.3 can guarantee its polynomial time complexity, its approximation ratio shows that the solution it generates might become worse if the minimum quantity b turns larger. This will be examined in practice through our experiments reported in the next Chapter 5.

4.8 Summary

We focus on a Basic Problem with the demand constraint and minimum quantity commitments only. To solve this problem, several optimization approaches are applied. Since the Basic Problem can be formulated as a mixed integer programming model strengthened by facets, a branch and cut algorithm works for small or some medium instances. Moreover, two heuristics are invented to solve large instances practically. One is a linear rounding heuristics, and the other is a greedy method. These approaches might be extended to solve more complicated selection and assignment problem with MQC constraints. For instance, it is easy to see that the most general case $optimization(w, s_i, f_i)$ can also be formulated as a mixed integer programming, therefore, the search techniques and other similar heuristics algorithms can be applied on it as well.

Besides that, the later Chapter 5 will measure the performance of the models and algorithms proposed here through the experiments.

Chapter 5

Experiments

5.1 Overview

The goal of this chapter is to measure the performance of the optimization methods, including branch and cut, linear programming rounding heuristics, and greedy approximation heuristics, which we proposed for the Basic Problem in the last Chapter 4.

We implemented all the algorithms by Microsoft Visual C++ 6.0, and made all the experiments on a Pentium III 800MHZ PC with 128M memory.

Instances with different features have been generated for the experiments, which will be described in Section 5.2. The next two sections will present and discuss the experimental results. We firstly examine the branch and cut search scheme in Section 5.3. It works well for small size instances and even for medium ones as well if the new facet is introduced. This rewards our efforts of strengthening the model, and allows us to have good solutions for all test cases by consuming endurable long time. Based on this, we test the two heuristics algorithms in Section 5.4. By comparing their results, we find that the linear programming rounding method generates better solutions than the greedy one. Although both of the two heuristics algorithms generate better solutions when the minimum quantity decreases, behaviors of the linear programming rounding method are more stable. Finally, we summarize this chapter in Section 5.5.

5.2 Configurations

An instance of the Basic Problem consists of five features, including the number of agents m, the number of requests n, the demand d, the transportation costs c, and the minimum quantity b. These features were generated in the following way.

We let the duplet (m, n) to indicate the instance size, because the mixed integer programming model of our Basic Problem has m integral variables and mn real variables. In our experiments, three scales of instance size were considered. They were (30,60) for small size, (60,120) for medium size, and (90,120) for large size.

For the demand d_j of each request j, we generated it randomly by a uniform distribution on the integral set {10,11,12,...,99,100}, where $1 \leq j \leq m$.

For transportation cost c, we had two types in our experiments. One was the random type, where each cost $c_{i,j}$, for $1 \leq i \leq m$ and $1 \leq j \leq n$, was chosen randomly by a uniform distribution on the continuous interval $[0, 1]$. The other was the metric type, where the transportation costs were nonnegative, symmetric, and satisfy the triangle inequality, as we mentioned in the last Chapter 4. To generate the metric transportation costs, we randomly chose $(m+n)$ integral points $P_0, P_1, ..., P_{m+n}$ from a planar rectangle $[0, 100] \times [0, 100]$, and let the cost $c_{i,j}$ be the distance from P_i to P_{m+j} for $1 \leq i \leq m$ and $1 \leq j \leq n$. Obviously, in this manner, the generated transportation cost c formed a metric.

The minimum quantity b was generated in a little complicated way. Recall that the maximum number of selected agents, who transport some shipments, is equal to $\lfloor D/b \rfloor$, where $D = \sum_{j=1}^{n} d_j$ is the total demands. To let $\lfloor D/b \rfloor$ distribute uniformly, we defined w_b to be the value of D/b, as a percentage of the agent number m, i.e.

$$w_b = \frac{D}{bm} \times 100\%.$$

We assigned w_b by one of the following 20 values: $5\%, 10\%, ..., 100\%$, and calculate the minimum quantity by $b = D/(mw_b)$. But to keep b integral in practice, we preferred $b = \lfloor D/(mw_b) \rfloor$ instead. It is easy to see that when b is decreasing, w_b is increasing.

The whole test cases were generated as follows. Since we had two types of trans-

portation costs and three scales of instance size, our test cases are categorized into six groups, as shown in Table 5.1. For random (or metric) type, denoted by character 'R' (or 'M'), we had three groups of different scales, R-I (or M-I) for small size, R-II (or M-II) for medium size, and R-III (or M-III) for large size.

In each group of test cases, we generated 10 different sets of transportation cost $c_{i,j}$ and d_j for $1 \leq i \leq m$ and $1 \leq j \leq n$. Then, for each set, we had 20 instances with different values of minimum quantity b, which is from 5%, 10%, ..., to 100%. Totally, there were 200 test instances within each of the 6 groups.

Our experiments had two parts. First was for the branch and cut search algorithm. Besides testing its performance, we expected this search scheme to find and prove the exact optimum solutions for small or some medium size instances, and to generate good lower bounds and near-optimum solutions for large ones. By comparing the heuristic solutions with the best lower bounds or best solutions we found, we could test the performance of the heuristics algorithms in the second part of our experiments.

To obtain good lower bounds and solutions for comparisons, we had run our branch and cut search scheme for endurable long time until the results had good qualities. The qualities are shown in Table 5.2. In total, we have found and proved all the optimum solutions for small instance groups M-I and R-I and the medium group M-II. For the other medium group R-II and the large group M-III, we achieved 199 optimums over 200 instances within each. And their maximum gaps, between the best lower bound and best solution value, were only 3.30% and 1.58%. For the large random group R-III, we proved 180 optimums over 200 instances. Although the maximum gap was relatively big (13.11%), its average gap was only 0.04%. The above observations tell that the best lower bounds and best solutions we got for the whole six groups of test cases were good enough for us to measure other experimental results by comparisons.

Table 5.1: Configurations of each group of test cases

Group ID[a]	Number of Agents m	Number of Requests n	Transportation Cost $c_{i,j}$
R-I	30	60	Random
R-II	60	120	Random
R-III	90	180	Random
M-I	30	60	Metric
M-II	60	120	Metric
M-III	90	180	Metric

[a]Character 'R'/'M' represents that the type of transportation cost is Random/Metric, and its roman index indicate the instance size, i.e. I for small, II for medium, and III for large.

Table 5.2: Qualities of best lower bound and best solution we found

Group ID	Number of Optimums Proved[a]	Average Gap of LB from BEST (%)[b]	Maximum Gap of LB from BEST (%)[c]
R-I	200	0.00%	0.00%
R-II	199	0.02	3.30
R-III	180	0.04	13.11
M-I	200	0.00	0.00
M-II	200	0.00	0.00
M-III	199	0.01	1.58

[a]The number of optimum solutions found and proved among 200 instances within each group.
[b/c]Average/Maximum difference between best lower bound (LB) and best solution value (BEST), as a percentage of best solution value, and over 200 instances within each group.

5.3 Performance of Branch and Cut

We had applied the branch and cut search scheme on the mixed integer programming model (*MIP*) of the Basic Problem, and on the strengthened model (*S-MIP*) as well. Recall that the only difference between the two models is the new facet $z_{i,j} \leq x_i$ for $i \in I$ and $j \in J$, which is excluded from the *MIP*, but included in *S-MIP*.

We implemented them by Microsoft Visual C++ 6.0 along with the libraries of CPLEX 8.0, and adopted the default configurations of the integer programming method in CPLEX 8.0, except the time limit that was set differently for different situations.

Table 5.3: Performance of the branch and cut algorithm tested over small problem instances within groups R-I and M-I.

	R-I		M-I	
	MIP	S-MIP	MIP	S-MIP
Average Time (s)[a]	49.19	34.58	10.13	14.93
Maximum time (s)[b]	680.38	514.62	162.51	281.98
Number of Optimums proved in 300s[c]	193	195	200	200

[a/b]Average/Maximum time in seconds, consumed to find and prove an optimum solution, over 200 small problem instances within each test case group.
[c]The number of optimum solutions found and proved in 300 seconds, and over 200 instances within each test case group.

To examine the effectiveness of the new facet in *S-MIP*, we tested both *MIP* and *S-MIP* on the small and medium instance groups. For small instances, we examined the consuming time of finding and proving the optimums. As shown in Table 5.3, *MIP* and *S-MIP* had similar performance, since both of them achieved the optimums in relatively short average time, and proved optimums for most cases in 300 seconds. Their maximum consuming time was also comparative. In a word, the branch and cut search scheme worked well for small size instances, whether or not the new facet was included.

However, for the medium size instances, the new facet did improve the search performance. We set the time limits of both *MIP* and *S-MIP* to be 300 seconds for each test case, because *MIP* was observed to consume quite a long time before stopping for some medium cases. When the time limits were met, we recorded their best lower bounds and best solution values respectively. Table 5.4 shows their comparisons. Among all the 200 medium instances within each group, *S-MIP* proved much more optimums than *MIP*, that was 152 vs. 49 in group R-II and 184 vs. 38 in group M-II. By comparing their average gaps between the lower bounds and the best solution values, we found that *S-MIP* had generated much closer lower bounds and solutions than *MIP* did, and therefore converged much faster.

Since *S-MIP* converged relatively fast, we could use it to prove more optimums for medium test cases, and to generate more good lower bounds and good solutions even for large cases. By setting the time limit of *S-MIP* to be 30 hours, we obtain the best lower

Table 5.4: Convergence speed of branch and cut over medium problem instances within groups R-II and M-II.

	R-II		M-II	
	MIP	S-MIP	MIP	S-MIP
Number of Optimums proved in 300s [a]	49	152	38	184
Average Gap of LB from BEST in 300s [b]	18.21%	6.48%	6.54%	0.32%

[a]The number of optimum solutions, found and proved in 300 seconds, and over 200 medium problem instances within each test case group.
[b]Average difference between the best lower bound (LB) and the best solution value (BEST), found in 300 seconds, as a percentage of the latter, and over 200 medium problem instances within each test case group.

bounds and solution values with pretty good qualities, as we seen in Table 5.2 of the last Section 5.2.

The new facet, included in the *S-MIP*, had imposed significant improvements on the search efficiency. To reveal its reason further, we compared *S-LP* and *LP*, which denoted the lower bounds of optimums, by solving the relaxed linear programming model of *S-MIP* and *S-MIP* respectively. Their average deviations from the best solution values are shown in Table 5.5. Clearly, *S-LP* was much closer to the best solution values. Among all the six test case groups, the minimum average deviation of *LP* was 20.63%, much bigger than 6.10%, the maximum average deviation of *S-LP*. In other words, the relaxed *S-MIP* model exhibited to be much tighter to the integral solution set than the relaxed *MIP* one, which could explain the reason of the significant improvements of convergence speeds when the new facet were introduced for the *S-MIP*.

5.4 Performance of Two Heuristics

The two heuristics, proposed in the laster Chapter 4, were implemented by Microsoft Visual C++ 6.0. One was the linear programming rounding heuristics, denoted by *LP-R*. The other was the greedy approximation heuristics, denoted by Greedy.

To measure their performance, we run the two heuristics over all the 200 instances within each six test case groups. For comparison, we also run the *S-MIP* search scheme

Table 5.5: Linear programming lower bound % gap from best solution value.

	Gap from Best Solution Value (%)[a]	
Group ID	LP	S-LP
R-I	33.77%	6.10%
R-II	34.35	4.89
R-III	34.45	5.07
M-I	22.58	2.39
M-II	21.72	1.07
M-III	20.63	1.02

[a]Figures represent average deviation between linear programming lower bound and best solution value, as a percentage of best solution value, and over 200 instances within each test case group.

but with 300 seconds time limit. This various search scheme is denoted by *S-MIP(300s)*. Its time limit is set to 300 seconds, because the longest time consumed by other two heuristics seldom exceeded 300 seconds.

Heuristics solutions by the three algorithms were compared with the best lower bounds we obtained. Their average gaps are shown in Table 5.6. Obviously, the *LP-R* performed much better than the *Greedy*, and even better than the *S-MIP(300s)* for some large size instances. Over all the cases, the maximum average gap of *LP-R* was only 1.16%, much less than 6.85%, the minimum average gap of Greedy. Moreover, the *LP-R* performed well for both random groups (1.29% on average), and metric groups (0.97% on average). On the contrary, the Greedy generated worse solutions for random groups (27.54% on average) than metric groups (6.63% on average). In addition, over groups R-III and M-III of large size instances, the average gap of *S-MIP(300s)* became worse, 38.02% for R-III and 0.93% for M-III, however, the average gap of *LP-R* still kept small, 1.85% for R-III and 0.66% for M-III. These observations demonstrated the good stability of the *LP-R*.

Besides the average gaps, we also counted the number of optimum solutions achieved by heuristics. Table 5.7 summarizes the numbers, and shows that Greedy generated few optimums, but *LP-R* generated some. It is interesting to see that the *LP-R* achieved more optimums for random cases than metric ones, inversely with *S-MIP(300s)*.

Furthermore, we had examined the time consuming for each heuristics algorithms.

Table 5.6: Average gaps of heuristics solution from best lower bound

	Average Gap from best lower bound (%)[a]		
Group ID	S-MIP(300s)	LP-R	Greedy
R-I	0.02%	1.16%	21.99%
R-II	12.62	0.86	28.57
R-III	38.02	1.85	32.05
M-I	0.00	1.24	6.06
M-II	0.09	1.02	6.98
M-III	0.93	0.66	6.85

[a]Figures represent the average gaps of the near-optimum solutions, generated by heuristics, from best lower bounds, over 200 instances within each test case group.

Table 5.7: The number of optimums found by heuristics

	Number of Optimums found[a]		
Group ID	S-MIP(300s)	LP-R	Greedy
R-I	199	131	0
R-II	167	86	0
R-III	124	47	0
M-I	200	96	5
M-II	190	56	0
M-III	167	30	0

[a]Figures represent the number of optimum solutions found by heuristics, over 200 instances within each test case group.

It is reported in Table 5.8. The Greedy was no doubt the fastest heuristic, for its average time consuming was under 1.0 seconds for all the groups. The *LP-R* is slower than Greedy, but much faster than the *S-MIP(300s)*.

As we seen, the *LP-R* had good performance among all the groups, but we could not prove its theoretical performance guarantee. Remember that *Greedy* has a $2b$ approximation ratio, which make us suspect that the solution generated by *Greedy* may become better if b is decreasing. But is it true in practice? And is it true for *LP-R* as well?

To examine our suspicion, let us see how the two heuristics, *LP-R* and *Greedy*, performed in practice when b was decreasing. To make b decrease, we could let w_b increase, since we have $b = \lfloor D/(mw_b) \rfloor$. Figure 5.1 shows the trends of the average gaps between the heuristic solutions and the best lower bounds when w_b was increasing. Within each

Table 5.8: Average time consumed by heuristics

	Average Time (s)[a]		
Group ID	S-MIP(300s)	LP-R	Greedy
R-I	30.83s	1.21s	0.00s
R-II	101.66	11.94	0.03
R-III	159.44	56.54	0.10
M-I	14.93	0.65	0.01
M-II	52.17	4.22	0.03
M-III	121.90	15.95	0.10

[a]Figures represent the average time in seconds, consumed by heuristics, over 200 instances within each test case group.

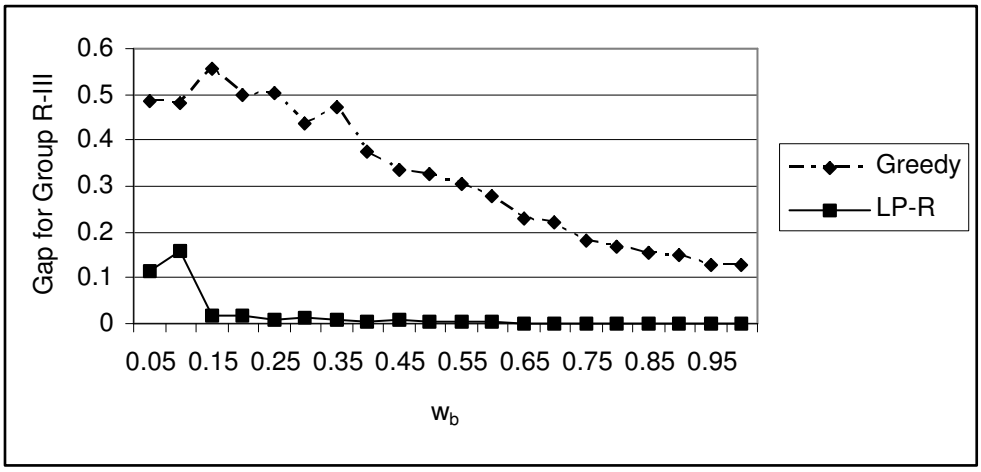
Table 5.9: Statistics of heuristics performance with w_b increasing, over large problem instances within groups R-III and M-III.

	R-III		M-III	
	LP-R	Greedy	LP-R	Greedy
Average Gap (%)[a]	1.85%	32.05%	0.66%	6.85%
Standard Deviation[b]	0.04	0.15	0.01	0.03
Linear Trend[c]	-0.08	-0.48	-0.02	-0.11

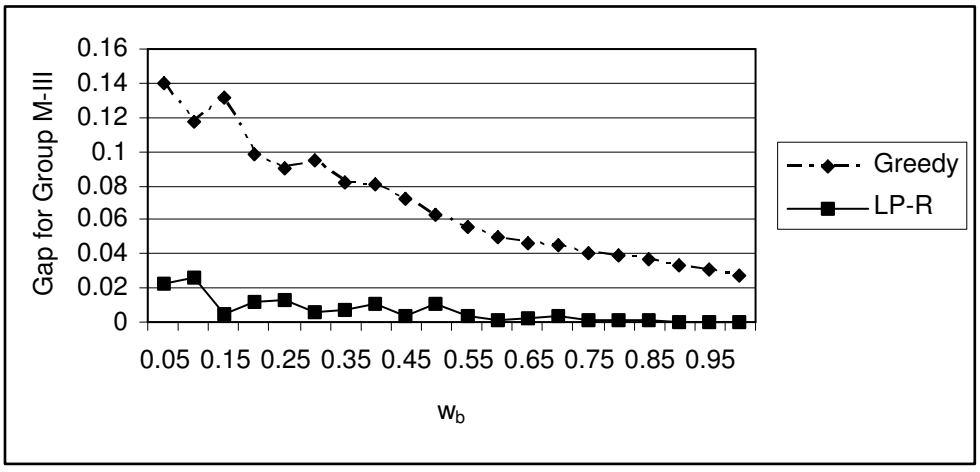
[a]Average difference between heuristics solution value and best lower bound, as percentage of best lower bound, and over 200 large problem instances within each test case group.
[b/c]Standard deviation/Linear trend of the difference between heuristic solution value and best lower bound, over 200 large problem instances within each test case group.

of the two large test groups, R-III and M-III, the average gaps turned smaller for both *LP-R* and *Greedy*. That confirms our suspicion on the effects of b. Moreover, we can also observe that *LP-R* not only generated better solution than *Greedy*, but also had stabler behaviors as well. To see this more clearly, we present the statistics of Figure 5.1 in Table 5.9. Compared with *Greedy*, *LP-R* had smaller average gap, smaller standard deviation, and smaller absolute linear trends as well.

In summary, the linear programming rounding heuristics has good performance among all the various test instances in practice. But the greedy approximation heuristics has not, except for its fast running speed.



(a)



(b)

Figure 5.1: Average difference between heuristics solution and best lower bounds, over 10 large problem instances for each w_b increasing from 0.05 to 1.00, and within groups R-III and M-III respectively.

5.5 Summary

We measured the performance of algorithms proposed for the Basic Problem in this chapter. The branch and cut search scheme on the strengthened mixed integer programming model helped us to generate almost all the optimum solutions for small and medium test instances, and good lower bounds and solutions for large ones. Compared with the best lower bounds, we examined the performance of the two heuristics methods proposed, that is the linear programming rounding and the greedy approximation. Although the greedy method can guarantee a theoretical $2b$ approximation ratio, the linear programming rounding heuristics over-performed the greedy and have a stable behaviors over all the groups of various test cases.

However, there are still some problems in our experiments. Firstly, the exact search scheme could not handle large instances in short time, although it produced good lower bounds, and therefore, fasten the speed of convergence. Its major cause was the speed of generating good solutions. So, to overcome this disadvantage, we plan to combine the search scheme with the linear programming rounding method, since the latter heuristic can generate good solution relatively fast. The other problem is for the test cases. In current experiments, only random data are generated. To make it more practical, we will try to collect some real cases for our later experiments and studies.

Chapter 6

Conclusion

The minimum quantity commitment, studied in this thesis, is motivated from the bidding problem of the Philips, and has broad applications for the selection and assignment problems under other real circumstances. For example, when we try to select locations to open schools for districts, each school need to hold sufficient number of students. Or, when we open warehouses for regions, each should keep enough goods in storage. However, in previous literature, this practical constraint has never been studied for selection and assignment problems. For this reason, we expect the work presented in this thesis to initiate the research on this subject.

We formulated and classify various special cases of the selection and assignment with minimum quantity commits, along with a complete figure of their computational complexity. For most special cases, finding an optimum solution, or sometimes finding a feasible solution, is intractable. So we focused the work on finding a near-optimum solution for a basic selection and assignment problem, where only the minimum quantity commitments and the demand constraints were considered. This basic problem was proved to be $\mathcal{NP}$-hard as well, and even finding its approximate solution was intractable in theory. To solve it practically, we formulated it by a mixed integer programming model, and strengthened the model by a new facet. Afterwards, we applied on it a branch and cut search algorithm, which worked well for small or some median size problem instances. To solve large size instances, we designed two heuristics. One was linear programming

rounding method, which performed well in experiments. The other was greedy approximation heuristics, which could theoretically guarantee a approximation ratio for the metric version of the Basic Problem.

In the study of the Basic Problem, some open problems and further works are still remained. At the theoretical level, we do not know whether finding an optimal solution is $\mathcal{NP}$-hard if the minimum quantity b is equal to two. Remember, for $b \geq 3$ the problem is proved to be $\mathcal{NP}$-hard, while for $b = 1$ it is trivially solvable. The other theoretical problem is about its approximation. As we shown in Chapter 4, the Basic Problem can not be solved approximately for general cases, but has a greedy algorithm that guarantees a non-constant ratio for the metric version. Noting that the metric condition is satisfied in many practical situations, we are very interested in improving the approximation ratio for the metric version of the Basic Problem. This seems possible, because for most traditional selection and assignment problems, like facility location problem and p-median problem, we have no constant ratio approximation algorithms for general cases, but have constant approximation ratio for their metric versions. In fact, we are now considering the approximation ratio of the linear programming rounding heuristics we proposed, because it over-performed the greedy during the experiments.

At the practical level, we need to improve the branch and cut search scheme to handle the larger size instances of the Basic Problem. Our improvements may have the following two ways. On one hand, since the Basic Problem can be formulated as a mixed integer programming model, we can analyze the polyhedral structures of the integral solution sets and find more effective strong inequalities, or facets, for the model. This might strengthen the model further and generate better lower bounds during the search process. On the other hand, to improve the branch and cut scheme, we can try to generate good integral solutions along the searching. Remember during the original branch and cut scheme, we have got a lot of fractional solutions to compute the lower bounds. If we can apply an efficient heuristics, like the linear programming rounding method, to transform those fractional solutions to good feasible solutions, the searching performance will certainly be improved a lot.

Besides the study for mixed integer programming, we will apply more heuristics methods to generate near-optimum solutions of the Basic Problem as well. Ideas can be illuminated by the meta-heuristics algorithms, like genetic algorithm [22], tabu search [13], and simulated annealing [18]. Those heuristics algorithms have been extensively studied for solving other traditional selection and assignment problems. For instance, tabu search [23] and genetic algorithms [7] have been applied for solving p-median problem, and exhibit good performance in practical experiments. Therefore, we can apply them on the Basic Problem, to see whether they are still effective if the new minimum quantity commitment constraint is introduced.

At the experiment level of the Basic problem, we need to collect more real data, because currently, only random test cases are available. Besides this, we also have to enlarge the instance size. Note that the current maximum instance size holds 90 agents and 180 requests. As we know, in the real application of Philips' bidding problem, the number of agent is not more than 100, but the requests number is much bigger, about 1000 or so. So, we should enlarge the instance size for our testing. And correspondingly, we have to improve the speed of our algorithms as well.

In addition to the Basic Problem, there are many other special cases that need to consider more constraints, like fixed selection costs, the maximum agents' capacity, the number of regions, etc. To solve these complicated special cases, we can employ results and techniques of the Basic Problem for reference. For example, it is easy to see that all the special optimization problems we studied in Chapter 3 can be formulated as a mixed integer programming model. Similarly to the Basic Problem, the branch and cut algorithm and the linear programming rounding heuristics can also be applied for them. However, since the special cases turns complicated, the structure of the mixed integer programming model is difficult to be analyzed and strengthened.

Our practical object is to build up a bidding system for the Philips to help them improve their performance of selection and assignments agents with minimum quantity commitments. During our efforts towards this object, we hope to achieve some theoretical discoveries as well.

Bibliography

[1] K. Aardal and C. van Hoesel. Polyhedral techniques in combinatorial optimization i: Theory. *Statistica Neerlandica*, 50(3), 1995.

[2] K. Aardal and C. van Hoesel. Polyhedral techniques in combinatorial optimization ii: Applications and computations. *Statistica Neerlandica*, 50(3), 1995.

[3] K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network flows : theory, algorithms, and applications*. Prentice Hall, Englewood Cliffs, N.J., 1993.

[4] K.S. Al-Sultan and M.A. Al-Fawzan. A tabu search approach to the uncapacitated facility location problem. *Annals of Operations Research*, 86:91–103, 1999.

[5] Vijay Arya, Naveen Garg, Rohit Khandekar, Kamesh Munagala, and Vinayaka Pandit. Local search heuristic for k-median and facility location problems. In *ACM Symposium on Theory of Computing*, pages 21–29, 2001.

[6] Y. Bassok and R. Anupindi. Analysis of supply contracts with total minimum commitment. *IIE Trans*, 29:373–381, May 1997.

[7] B. Bozkaya, J. Zhang, and E. Erkut. A genetic algorithm for the p-median problem. In Z. Drezner and H. Hamacher, editors, *Facility Location: Applications and Theory*. Springer, Berlin, 2002.

[8] Perter Brucker. *Scheduling Algorithms*. Springer-Verlag, 1995.

[9] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. The MIT press, Cambridge, Massachusetts, 1997.

[10] G. Cornùejols, G. L. Nemhauser, and L. A. Wolsey. The uncapacitated facility location problem. In P Mirchandani and R. Francis, editors, *Discrete Location Theory*, pages 119–171. Wiley, New York, NY, 1990.

[11] Ismael Regis de Farias Jr. A family of facets for the uncapacitated p-median polytope. *Operations Research Letters*, 28, 2001.

[12] Michael R. Garey and David S. Johnson. *Computers and intractability: a guide to the theory of NP-completeness*. Freeman, New York, 1983.

[13] Fred Glover and M. Laguna. Tabu search. In C. Reeves, editor, *Modern Heuristic Techniques for Combinatorial Problems*, Oxford, England, 1993. Blackwell Scientific Publishing.

[14] S.L. Hakimi. Optimum distribution of switching centers in a communication network and some related graph theoretic problems. *Operations Research*, 13:462–475, 1965.

[15] Dorit S. Hochbaum. Heuristics for the fixed cost median problem. *Mathematical Programming*, 22:148–162, 1982.

[16] K. Jain, M. Mahdian, and A. Saberi. A new greedy approach for facility location problems. In *Proceedings of 34th ACM Symposium on Theory of Computing (STOC)*, 2002.

[17] Y. Pochet K. Aardal and L. A. Wolsey. Capacitated facility location: Valid inequalities and facets. *Mathematics of Operations Research*, 20:552–582, 1995.

[18] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science, Number 4598, 13 May 1983*, 220, 4598:671–680, 1983.

[19] M. R. Korupolu, C. G. Plaxton, and R. Rajaraman. Analysis of a local search heuristic for facility location problems. In *Proceedings of 9th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 1998.

[20] James Orlin. A faster strongly polynomial minimum cost flow algorithm. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 377–387. ACM Press, 1988.

[21] Hasan Pirkul. Efficient algorihtms for the capacitated concentrator locatin problem. *Computers and Operations Research*, 14, 1987.

[22] Nicholas J. Radcliffe. Genetic set recombination. In L. Darrell Whitley, editor, *Foundations of Genetic Algorithms 2*, San Mateo, CA, 1993. Morgan Kaufmann Publishers.

[23] E. Rolland, D. A. Schilling, and J. R. Current. An efficient tabu search procedure for the p-median problem. *European Journal of Operational Research*, 21:329–342, 1996.

[24] Alan Scheller-Wolf and Sridhar Tayur. Reducing international risk through quantity contracts, 1998. working paper, Manufacturing and Operations Management, Carnegie Mellon, Pittsburgh, PA.

[25] David B. Shmoys, Éva Tardos, and Karen Aardal. Approximation algorithms for facility location problems (extended abstract). In *Proceedings of the 29th Annual ACM Symposium on the Theory of Computing*, pages 265–274, 1997.

[26] Laurence A. Wolsey. *Integer Programming*. Wiley, John & Sons, Incorporated, 1998.

Appendix A

Details of Some Proofs

A.1 Proof of facets for other three cases in Theorem 4.7

Proof Recall that we assume $2b < D$. Now we prove that the other three cases in Theorem 4.7 are also facet-defining for the convex hull of the *MIP* model's feasible set. These three cases includes $x_i \leq 1$ for $i \in I$, (4.18) and (4.14).

Firstly, for any agent $p \in I$, we are going to prove the inequality $x_p \leq 1$ is a facet. To prove this, we must show that if all feasible solutions of the *MIP* model that satisfy $x_p = 1$ also satisfy

$$\sum_{i \in I} \sum_{j \in J} \alpha_{i,j} z_{i,j} + \sum_{i \in I} \beta_i x_i = \theta, \tag{A.1}$$

at equality, then (A.1) must be a linear combination of $x_p \leq 1$ and the equality (4.14).

In the similar way of the first and second steps of the proof for the case (4.15) in Theorem 4.7, we can show $\alpha_{v,j} = \alpha_{p,j}$ and $\beta_u = 0$ for $v \in I$, $u \in I - \{p\}$ and $j \in J$. By assuming $\alpha_{v,j} = \alpha_j$, we can rewrite (A.1) as

$$\sum_{j \in J} \alpha_j \sum_{i \in I} z_{i,j} + \beta_p x_p \leq \theta. \tag{A.2}$$

It is easy to see $\theta = \sum_{j \in J} \alpha_j d_j$, therefore, (A.2) is a linear combination of $x_p \leq 1$ and the equality constraint (4.14).

Secondly, for any agent $p \in I$ and any request $q \in J$, we are going to prove the

inequality $z_{p,q} \geq 0$ is a facet. To prove this, we must show that if all feasible solutions of the *MIP* model that satisfy $z_{p,q} = 0$ also satisfy (A.1) at equality, then (A.1) must be a linear combination of $z_{p,q} \geq 0$ and the equality (4.14).

Similarly to the first and second steps of the proof for the case (4.15) in Theorem 4.7, we can show $\alpha_{u,j} = \alpha_{v,j} = \alpha_j$ for and $\beta_u = 0$ for $u \in I$, $v \in I - \{u\}$ and $j \in J - \{q\}$. In the same way, we can also show $\alpha_{u,q} = \alpha_{v,q} = \alpha_q$ for $u, v \in I - \{p\}$. Therefore, (A.1) can be written as

$$\sum_{j \in J} \alpha_j \sum_{i \in I} z_{i,j} + (\alpha_q - \alpha_{p,q}) z_{p,q} \leq \theta. \tag{A.3}$$

It is easy to see $\theta = \sum_{j \in J} \alpha_j d_j + \beta_p$, therefore, (A.3) is a linear combination of $z_{p,q} = 0$ and the equality constraint (4.14).

Lastly, for any request $q \in J$, we are going to prove the equality (4.14) for the request q, i.e.

$$\sum_{i \in I} z_{i,q} = d_q, \tag{A.4}$$

is a facet. To prove this, we must show that if all feasible solutions of the *MIP* model that satisfy (A.4) also satisfy (A.1), then (A.1) must be a linear combination of the equality (4.14).

In the same way as the first and second steps of the proof for the case (4.15) in Theorem 4.7, we can show $\alpha_{u,j} = \alpha_{v,j}$ and $\beta_u = 0$ for $u \in I$, $v \in I - \{u\}$ and $j \in J$. By assuming $\alpha_{u,j} = \alpha_j$, we can rewrite (A.1) as

$$\sum_{j \in J} \alpha_j \sum_{i \in I} z_{i,j} = \theta. \tag{A.5}$$

It is easy to see $\theta = \sum_{j \in J} \alpha_j d_j$, therefore, (A.5) is a linear combination of the equality constraint (4.14). ∎

A.2 Proof of the $2b$ approximation ratio of Algorithm 4.3 for Theorem 4.9

Under the assumption that the transportation cost $c_{i,j}$ forms a metric for $i \in I$ and $j \in J$, we are going to prove that the greedy Algorithm 4.3 will generate a feasible solution whose total cost is at most $2b$ times the optimum. Since we have shown that any instance can be transformed to the unit demand case by splitting, we can assume that $d_j = 1$ for all $j \in J$ without invalidating the approximation ratio we will prove. Therefore, the request set J represents the container set as well.

For $b = 1$, since we assign each container to the agent bidding lowest cost for it, the greedy Algorithm 4.3 generates the optimum solution and its approximation ratio is certainly at most $2b$.

Otherwise, let us assume $b \geq 2$. Under this assumption, the proof of $2b$ approximation ratio needs more notations. On one hand, in the optimum assignment, for each agent $i \in I$, we let A_i denote the set of containers assigned to i , and n_i denote the size of A_i. By the minimum quantity commitment, we have either $n_i = 0$ or $n_i \geq b$. We use $\overline{opt}_i$ to denote the cost of assignments to the agent i, so that $\overline{opt}_i = \sum_{j \in A_i} c_{i,j}$. On the other hand, in the greedy assignment, for each container $j \in J$, let $\delta(j)$ denote the agent to whom the container j is assigned, and $f(j)$ represent the cost of the operation, $selection(\delta(j))$ or $assignment(\delta(j))$, which assigns j to $\delta(j)$ in the greedy Algorithm 4.3. So we have that the optimum total cost is $\sum_{i \in I} \overline{opt}_i$, and the greedy total cost is $\sum_{i \in I} \sum_{j \in A_i} f(j)$ by partitioning the container set J into $A_1, A_2, ..., A_m$. The only thing we need to prove becomes

$$\sum_{i \in I} \sum_{j \in A_i} f(j) < 2b \sum_{i \in I} \overline{opt}_i, \tag{A.6}$$

which can be directly drawn from the following lemma.

Lemma A.1 *For each agent $i \in I$, we have*

$$\sum_{j \in A_i} f(j) < 2b\overline{opt}_i. \tag{A.7}$$

Proof We prove (A.7) for the first agent, i.e. $i = 1$ only, because for others, the proof is the same. We can also assume $n_1 \geq b$, since otherwise A_1 is empty so that (A.7) is obviously true for $i = 1$.

Now, let us consider the n_1 containers assigned to the agent 1 in the optimum assignments. Without loss of generality, they are supposed to be the containers $1, 2, ..., n_1$ that form the set A_1, and to be assigned in an non-decreasing order of time by the greedy algorithm. In other words, for any two container p and q where $1 \leq p < q \leq n_1$, the container p is assigned at least as early as q in the greedy Algorithm 4.3. Therefore, before we try to assign the container q greedily, the agent $\delta(p)$ has already been selected. Since the cost of $selection(\delta(p))$ is at most $c_{\delta(p),q}$ at that time, the operation cost $f(q)$ of assigning q must be at most $c_{\delta(p),q}$ as well. Noting that $c_{i,j}$ forms a metric for $i \in I$ and $j \in J$, we have $c_{\delta(p),q} < c_{\delta(p),p} + c_{1,p} + c_{1,q}$, and therefore,

$$f(q) \quad \leq \quad c_{\delta(p),p} + c_{1,p} + c_{1,q}, \text{ for } 1 \leq p < q \leq n_1, \tag{A.8}$$

which gives an upper bound of the greedy operation cost.

Moreover, its lower bound can be computed as well. Consider any container j where $1 \leq j \leq m$. It is easy to see that either $f(j) = c_{\delta(j),j}$ if its greedy operation is $assignment(\delta(j))$, or $f(j) \geq c_{\delta(j),j}/b$ if that is $selection(\delta(j))$. Both satisfy

$$f(j) \geq \frac{c_{\delta(j),j}}{b}, \text{ for } 1 \leq j \leq n_1. \tag{A.9}$$

However (A.9) and (A.8) are not enough. To obtain (A.7), we need a more sophisticated upper bound of the greedy operation for some containers that have large transportation cost to the agent 1, say larger than or equal to $\overline{opt}_1/n_1$. Because $\sum_{j=1}^{n_1} c_{1,j} = \overline{opt}_1$, we know that $t = \min\{j | c_{1,j} \leq \overline{opt}_1/n_1, j \in A_1\}$ must exist. So we have

$$c_{1,j} \geq \frac{\overline{opt}_1}{n_1}, \text{for } 1 \leq j \leq t-1. \tag{A.10}$$

Consider any container j where $1 \leq j \leq min(t, n_1 - b + 1)$. We now going to prove its greedy operation cost $f(j)$ is at most $\overline{opt}_1/n_1$ as follows. Before we assign the container j, containers $1, 2, ..., j-1$ have been assigned. Because neither $selection(\delta(j))$

or $assignment(\delta(j))$ will cost more than the average of $c_{1,p}$ for all unassigned containers p in A_1, we have

$$f(j) \quad \leq \quad \frac{\overline{opt}_1 - \sum_{p=1}^{j-1} c_{1,p}}{n_1 - j + 1}.$$

By (A.10), we have $\sum_{j=1}^{g-1} c_{1,j} \leq (g-1)\overline{opt}_1/n_1$, and therefore, $f(g) \leq \overline{opt}_1/n_1$, leading:

$$f(j) \leq \frac{\overline{opt}_1}{n_1}, \text{ for } 1 \leq j \leq min(t, n_1 - b + 1). \tag{A.11}$$

Now, we finish the proof of (A.7) for $i = 1$, by considering the following two cases.

One one hand, suppose $t \leq n_1 - b + 1$, which implies $min(t, n_1 - b + 1) = t$. We written the greedy total cost in the following form.

$$\sum_{j \in A_1} f(j) \quad = \quad \sum_{j=1}^{t} f(j) + \sum_{j=t+1}^{n_1} f(j).$$

By (A.10), we know $f(j) \leq \overline{opt}_1/n_1$ for $1 \leq j \leq t$. For $t+1 \leq j \leq n_1$, we have $f(j) \leq (b+1)\overline{opt}_1/n_1 + c_{1,j}$, because $f(j) \leq c_{\delta(t),t} + c_{1,t} + c_{1,j}$ by (A.8), $c_{\delta(t),t} \leq bf(t) \leq b(\overline{opt}_1/n_1)$ by (A.9) and (A.11), and $c_{1,t} \leq \overline{opt}_1/n_1$. Therefore, noting $b \geq 2$, $1 \leq t$ and $\sum_{j=t+1}^{n_1} c_{1,j} \leq \overline{opt}_1$, we have:

$$\begin{aligned} \sum_{j \in A_1} f(j) \quad &\leq \quad t(\overline{opt}_1/n_1) + (n_1 - t)(b+1)\overline{opt}_1/n_1 + \overline{opt}_1 \\ &\leq \quad (b+2)\overline{opt}_1 \\ &\leq \quad 2b\overline{opt}_1. \end{aligned}$$

On the other hand, suppose $n_1 - b + 1 < t$, which implies $min(t, n_1 - b + 1) = n_1 - b + 1$. Similarly to the first case, we written the greedy total cost in a new form, that is

$$\sum_{j \in A_1} f(j) \quad = \quad \sum_{j=1}^{n_1-b+1} f(j) + \sum_{j=n_1-b+2}^{n_1} f(j).$$

By (A.11), we get $f(j) \leq \overline{opt}_1/n_1$, for $j \leq n_1 - b + 1$. For $n_1 - b + 2 \leq j \leq n_1$, we can obtain $f(j) \leq b(\overline{opt}_1/n_1) + \overline{opt}_1 + c_{1,j}$, because $f(j) \leq c_{\delta(1),1} + c_{1,1} + c_{1,j}$ by (A.8), $c_{\delta(1),1} \leq bf(1) \leq b(\overline{opt}_1/n_1)$ by (A.9) and (A.11), and $c_{1,1} \leq \overline{opt}_1$ obviously. Therefore,

noting $\sum_{j=n_1-b+2}^{n_1} c_{1,j} \leq \overline{opt}_1 - c_{1,1}$ and $b \leq n_1$, we have

$$\begin{aligned}
\sum_{j \in A_1} f(j) &\leq (n_1 - b + 1)\overline{opt}_1/n_1 + (b-1)b(\overline{opt}_1/n_1) + (b-1)c_{1,1} + (\overline{opt}_1 - c_{1,1}) \\
&\leq \left(b + \frac{(b-1)^2}{n_1}\right)\overline{opt}_1 \\
&\leq 2b\overline{opt}_1
\end{aligned}$$

The lemma is proved completely. ∎

By summing up (A.7) for $i \in I$, we obtain (A.6), which proves the $2b$ approximation ratio of the greedy Algorithm 4.3.