國立臺灣大學管理學院資訊管理學系

碩士論文

Department of Information Management
College of Management
National Taiwan University

Master Thesis

考慮顧客偏好與內生產能決策之設施位置問題

A Facility Location Problem with Customer Preference and Endogenous Capacity Decision

江柏宣

Po-Hsuan Chiang

指導教授：孔令傑 博士

Adviser: Ling-Chieh Kung, Ph.D.

中華民國 106 年 8 月

August, 2017

doi:10.6342/NTU201703025

國立臺灣大學碩士學位論文
口試委員會審定書

A Facility Location Problem with Customer Preference and Endogenous Capacity Decision

本論文係江柏宣君（學號 r04725020）在國立臺灣大學資訊管理學系、所完成之碩士學位論文，於民國 106 年 6 月 22 日承下列考試委員審查通過及口試及格，特此證明

口試委員：

所　　長：

doi:10.6342/NTU201703025

謝辭

總算完成了！用一份碩士論文來總結兩年來的碩士生活。這一路走來，要感謝的人真的太多太多了。

首先最要感謝的是指導老師孔令傑教授。從一開始懵懵懂懂搞不清楚狀況時就一路帶領著我進入作業研究的領域，一步一步帶我完成這本論文。除此之外更重要的是我在他的身上看到了認真負責的做事態度，老師對於教學很有理想、上過他課的人也都知道老師用心準備的課程品質。儘管平日工作繁忙依然能夠把每一件事都做到最好。這樣子的態度儼然成為我們的榜樣。

其次要感謝我的父母親，他們讓我可以心無旁騖的做自己想做的事，雖然有時也不太明白我在做什麼，但還是無條件的支持我，也在我快沒能量時給予充電鼓勵，每次回家都能夠得到滿滿的動力繼續向前。

再來要感謝的是 IDEO Lab 的學長姊、同學們以及學弟妹。這兩年的碩士生活中我們一起念書、一起吃飯、一起打電動、一起買飲料、一起趕死線、一起念 paper 做研究寫論文。碩班兩年因為你們豐富了許多，也因為大家的 carry 輕鬆了不少。

特別要感謝嫩雅，雖然有時因為忙碌而對妳有點疏忽了，但在崩潰時跟妳說說話心情就會好很多，感謝一路上的支持與照顧。資管系壘的學長學弟讓我在球場上暫時逃離繁重的課業，提供了一個放鬆心情的好去處，感謝各位的友善包容與不離不棄。

最後要感謝所有曾經跟我說過加油的人，你們的一句話也許只是不經意的說出口，但在我耳中都是很巨大的能量。謝謝大家！

江柏宣　謹啟

于台大資訊管理研究所

民國一百零六年八月

doi:10.6342/NTU201703025

摘要

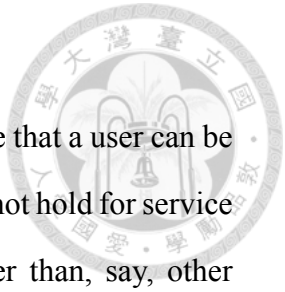
以往在考慮位置設施問題時，常常會假設使用者能夠被任意指派到任何一個設施。這樣的假設在討論提供服務的設施時就不太適用，一旦設施需要直接面對顧客而不是面對生產線上的下游時，消費者對於每個設施可能會有屬於自己的偏好，這些偏好可能來自於設施的位置、大小、提供的服務品質等等。決策者並不能強迫顧客去特定的設施，只能被動提供設施，由顧客來選擇。

在本研究中，我們研究一個決策者要如何根據已知的消費者偏好，去選擇設施的位置以及規模，使其利益最大化。我們考慮的情境有兩個階層，首先決策者決定設施的建造計畫後，接著消費者根據自己的偏好決定要去的設施。

針對這個兩階層的情境，我們設計出一個單階層的整數規劃模型。由於最大覆蓋問題是我們的問題的一個特例，我們便藉由一些經典的最大覆蓋問題演算法中得到靈感，再將消費者如何選擇設施的過程轉化為網路最大流問題，進而設計出一個以貪婪法為基礎的演算法，並且證明在特定情況下所提出的解與最佳解相距在一定比例內。我們同時提出另一個版本的演算法，將網路最大流問題以簡單的方式得到估計值，以大幅縮短求解時間。最後，我們透過數值分析驗證了我們的演算法的表現與求解時間。

關鍵字：設施位置問題、服務設施位置問題、近似演算法、最大覆蓋問題、網路最大流問題。

doi:10.6342/NTU201703025



Abstract

When we talk about facility location problems, we often assume that a user can be assigned to any facility by the decision maker. This assumption does not hold for service facilities. When facilities are providing service to customers rather than, say, other entities in a supply chain, customers often have their own preferences influenced by the location, capacity, service level, etc, of the facilities. The decision maker cannot enforce customer to go to a certain facility. Instead, he can only decide the locations and scales of built facilities. Customers will choose where to go by themselves.

In this study, we formulate our facility location problem as a single-layer integer program and find that the maximum cover problem is a special case of our problem. Inspired by some famous algorithms for the maximum cover problem, we design a greedy algorithm by transforming customers' decision into a maximum flow problem. We show that the algorithm has worst-case performance guarantees in some special cases.

By using a simple method to estimate the value of maximum flow, we propose a modified algorithm, which may perform worse than the first one but runs much faster than it. Finally, we study the average performance and computation time of the modified algorithm in various scenarios through numerical experiments

Keywords: Facility location problem, Service facility location problem, Approximation algorithm, Maximum coverage problem, Maximum flow problem.

doi:10.6342/NTU201703025

Contents

doi:10.6342/NTU201703025

doi:10.6342/NTU201703025

doi:10.6342/NTU201703025

List of Figures

doi:10.6342/NTU201703025

List of Tables

doi:10.6342/NTU201703025

Chapter 1

Introduction

1.1 Background and motivation

Over the years, the facility location problem has been a very important issue in both academia and industry (Daskin, 2013). A typical facility location problem is to decide where to build facilities within some given candidate locations in order to maximize the profit of the decision maker. The term "facility" cannot only be treated as a node of supply chain but also be considered as service nodes that serve end customers directly.

As mentioned by Camacho-Vallejo et al. (2014) and Lee and Lee (2012), most of the facility location problems mentioned in previous studies do not take customer's preference into account. Sometimes a customer does not have preference to certain facility. In this case, assigning different facility will not make any difference since all the facilities are homogeneous to the customer. However, sometimes facilities are heterogeneous for customers. One may have various preference levels to different facilities due to their

doi:10.6342/NTU201703025

different service types, traveling distances, and many other reasons.

Take YouBike, a public bicycle rental service in Taiwan, as an example. The system owner decides where to build stations, but they cannot decide where the customer will go. Customers will make their decision base on their own preferences. Traditionally, studies assume that customer will choose the nearest facility. But there may be many other reasons which can influence customer's preference in practice, making customers' and system owner's interests unaligned. This property also appears when building parks, hospitals, retail stores, and other "service facility".

When people think about facility location in the past, they often consider the facilities as a part of supply chain. Most of the papers are discussing about this type of facility. However, we need to consider preference when we are building service facilities. In order to fit this real life scenarios, we study the service facility location problem with customer preference.

1.2 Research objectives

There are groups of customers distributed in a given set of locations who want to be served. The decision maker is going to decide where to build facilities to maximize the benefits collected form serving customers while not exceed the given budget. If the decision maker decides to build a facility, she also needs to decide the scale of it. A larger scale means more customers the facility can serve, but the construction cost will also be higher. Once facilities are built, a customer will choose a facility according to his preference. A customer always goes to the facility he prefers the most as long as the

doi:10.6342/NTU201703025

facility still have residual capacity.

In this research we make two assumptions. First, we assume that all facilities provide the same kind of service. The decision maker can only modify the capacity of a facility. The second assumption is that one's preference is fixed. Preferences are given as parameters and will not be affected by whether there exists other facility or by the decision of other customers.

It is generally believed that there is no polynomial time algorithm for an NP-hard problem unless P = NP. However, we sometimes do not need an optimal solution in real world practices when the cost of finding an optimal solution is too high. Some researchers turn to develop heuristic algorithms for a near-optimal solutions. Numerical studies are often used to measure the performance of a heuristic algorithm. It will be better when there exists a worst-case performance guarantee. If an upper bound of the gap between optimal solution and the solution provided by the algorithm can be found, the algorithm is called an approximation algorithm (Williamson and Shmoys, 2011). We plan to construct an approximation algorithm for our facility location problem.

1.3 Research plan

In Chapter 2, we review some related works about the facility location problem and approximation algorithms. In Chapter 3, we introduce some notations and formulate our problem as a mixed integer program. The physical meanings of our model will also be explained. Chapter 4 contains worst-case performance analysis in some special cases and in Chapter 5, we present the results of a numerical study. Chapter 6 concludes.

doi:10.6342/NTU201703025

Chapter 2

Literature Review

2.1 Facility location problems

Facility location problems have been studied for several decades (Francis and White, 1974). Based on different objective functions and constraints given, traditional facility location problems can be classified into the following six different types.

1. (Set cover) Given a set of elements, and several sets of collections. Every collection is a set of elements associated with a cost. The objective is to find a set of collection who can cover all elements with minimum weight.

2. (Maximum cover) Given a number k and several sets who may have common elements. The objective is to maximize the elements covered with fewer than k sets.

3. (k-median) Given a weighted strongly connected graph, and a number k. The objective is to find k vertices to build facilities on, and minimize the moving distance between each vertex and his nearest facility.

doi:10.6342/NTU201703025

4. (k-center) Given a weighted strongly connected graph and a number k. The objective is to find a subset of vertices with cardinality no greater than k in order to minimize the maximum distance between each vertex to elements in the chosen subset.

5. (Uncapacitated facility location) Given a set of customer demand sets and a set of potential facility location, the task is to find a subset of potential facility locations to build facilities that the total travel distance for a customer to his nearest facility and the construction cost can be minimized.

6. (Capacitated facility location) Similar to the uncapacitated facility location problem, the objective is to decide where to build facilities in order to collect customers, but facilities have capacity constraints in this case. A facility can only provide service to a limited amount of demands. The uncapacitated facility location problem is a special case of the capacitated facility location problem.

Traditional problems assume that facility construction does not affect demand realization, which may not be true when facilities provide service to end customers. This motivates the studies of service facility location problems.

2.2 Service facility location problems

As we mention in Chapter 1, a service facility is very different from a node in the supply chain. Service facilities directly face customer demands. The decision maker has to consider more than just capacity and construction cost when building service facilities.

doi:10.6342/NTU201703025

Wu and Lin (2001) think customer demands should not be fixed at a point. The authors believe that customer demands happen on his scheduled trips. For example, customer will buy milk on their way home from work. This kind of problems are called flow-capturing location allocation problem. The authors includes the competition relationship between facilities. The objective is to maximize customers collected by building facilities considering existing facilities on the map. They design a greedy heuristic algorithm to solve this problem.

Aboolian et al. (2007) focus on two properties of service facilities, market expansion effect and cannibalization effect. When a new facility is opened, the new facility may stimulate hidden demands and benefits everyone. This is the so-called market expansion effect. However, the new facility may also bring negative impact on others since he "eats" the market share of other existing facilities. Such a cannibalization effect may therefore reduce customers of each facilities. A complex formulation is created in order to capture these two properties. A tangent line approximation is used to transform a continuous concave function into a piece-wise linear function and obtain a near optimal solution with worst-case performance guarantee.

Lee and Lee (2012) consider the effect caused by facility scales. A facility can provide more kinds of service if the scale is larger. However, a facility can only be built when there are sufficient potential demands. After formulating the problem, they apply Lagrangian relaxation and break the original problem to a knapsack problem and several facility location problems. Solutions from sub-problems are then combined via a heuristic algorithm.

doi:10.6342/NTU201703025

Liao (2016) also considers demand cannibalization effect and market expansion effect. Opening a facility not only affects customer demands but also how other facilities affect consumer demands. These interdependent effects are modeled as stand-alone benefit and network benefit.The sum of these two benefits will then be input into a nondecreasing concave function to capture the diminishing marginal benefit property. They design a heuristic algorithm to solve the problem and provide worst case performance guarantee analytically under some restrictions.

In the above studies, the authors assume that a customer always prefers the open facility closest to her. In effect, the facility builder can "assign" a facility to each customer as long as the facility is the closest. In the next section, we will review some works related to preference of customers in a more general setting.

2.3 Service facility location problem with preferences

Customers often have preferences with respect to each facility. The preference can be caused by distance, service type, facility scale, etc. Customers tend to go to the facility with the highest preference level. The facility builder also needs to take this into account in order to maximize his profit.

Camacho-Vallejo et al. (2014) build a bi-level facility location model. Customers hold their preferences with respect to each facility. In the second level problem, a customer decides where to go by choosing a facility that maximizes his preference level. Knowing customers' reaction, the system owner tries to maximize his profit by building facilities. By using an evolutionary based algorithm, a near-optimal solution is generated.

doi:10.6342/NTU201703025

Hanjoul and Petters (1987) provide several types of ways to formulate customer preference. One useful approach is an inequality proposed by Wagner and Falkson (1975). This inequality makes sure that a customer will go to facility A only when she has no option better than A. This can be considered as an alternative way to combine the facility builder's and customers' decisions into one single mathematical program. Following this idea, we generalize this inequality to include facilities with capacity constraints and scale variability.

2.4 Maximum coverage problem

Our goal is to design an algorithm to build facilities with finite capacity serving customers with limited budget. The problem reduces to variants of the maximum coverage problem if we relax capacity constraints by setting capacity to a large number. Consequentially, we focus on studies of approximation algorithms for maximum coverage problem.

Hochbaum and Pathria (1994) proved that a greedy-based algorithm of weighted maximum cover problem that selects the best current candidate facility to build is a $1-\frac{1}{e}$ approximation algorithm.

Khuller et al. (1999) designed a modified greedy algorithm that selects the highest ratio between collected element weights and the weight of set. They also find the approximation factor of $1-\sqrt{\frac{1}{e}}$ for this modified algorithm.

Nemhauser et al. (1978) proved that for a monotone and submodular objective function, we can design a greedy algorithm with approximation factor of $1-\frac{1}{e}$.

Hence, we are going to design a greedy-based algorithm and try to find an approxi-

doi:10.6342/NTU201703025

mation bound of the algorithm.

doi:10.6342/NTU201703025

Chapter 3

Problem Description and Formulation

We consider a scenario where the decision maker (she) decides where to build her facilities and the scale of them. A customer (he) will choose among built facilities according to preference. Consider $J = \{1, 2, 3, ..., n\}$ as the set of locations where a facility can be built on, $I = \{1, 2, 3..., m\}$ as the set of locations of customer, $K = \{1, 2, 3..., s_j\}$ as the set of different facility scales. For each location $j \in J$ and scale $k \in K$, there is a fixed construction cost $f_{jk} > 0$ and facility capacity $q_{jk} > 0$. Without lost of generality, $g_{j1} < g_{j2} < ... < q_{jm} \forall j \in J$. The decision variable y_{jk} is 1 if a facility is built on location j with scale k and 0 otherwise. For simplicity, we sometimes call customer at location i customer i and facility at location j facility j. The number of customers at location i is d_i. Figure 3.1 is an example of three customer nodes with demands equal to 1 and two single-layer facility nodes with capacities equal to 2.

For each customer $i \in I$, p_{ij} measures the degree of preference to location j. We

doi:10.6342/NTU201703025

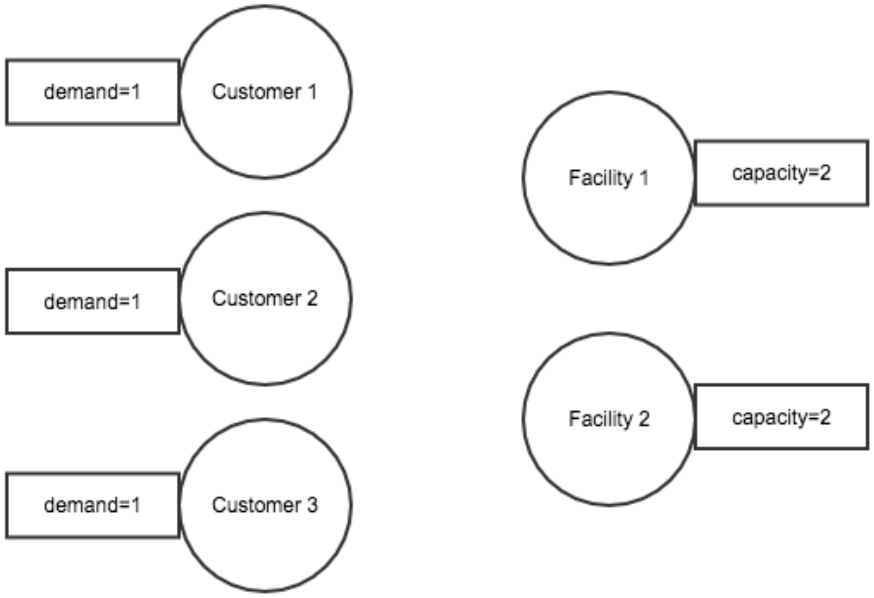


Figure 3.1: An example illustrating facilities and customers

say customer i prefers j_1 to j_2 if $p_{ij_1} > p_{ij_2}$. The decision variable x_{ij} stands for the proportion of customers at location i going to facility built at location j.

Aside from the customers, the facility builder also has a preference level r_{ij}, which may be different from the preference of customer p_{ij}.

Our goal is to maximize total preference levels we collected by serving customers with the total construction less than a given budget B. In order to formulate the problem, we define two auxiliary variables w_j and T_{ij}. w_j is a binary variable showing whether facility j is still available due to the capacity constraint, and T_{ij} is a set of facilities that are preferred to facility j by customer i, i.e.,

$$T_{ij} = \{j' \in J | p_{ij'} > p_{ij}\}. \tag{3.1}$$

doi:10.6342/NTU201703025

The objective function is $\sum_{i\in I}\sum_{j\in J} d_i r_{ij} x_{ij}$, which stands for the total builder's preference level collected by building facilities. In this study, we focus on the scenario where all customers are equivalent to the decision maker. That is $r_{ij} = r$. Without loss of generality we set $r_{ij} = 1$.

Collectively, the problem is formulated as

$$
\begin{aligned}
\max \quad & \sum_{i\in I}\sum_{j\in J} d_i x_{ij} \\
\text{s.t.} \quad & \sum_{j\in J} x_{ij} \le 1 \quad \forall i \in I \\
& \sum_{k\in K} y_{jk} \le 1 \quad \forall j \in J \\
& \sum_{i\in I} x_{ij} \le M_j^1 \sum_{k\in K} y_{jk} \quad \forall j \in J \\
& \sum_{k\in K} q_{jk} y_{jk} - \sum_{i\in I} d_i x_{ij} \le M_j^2 w_j \quad \forall j \in J \\
& x_{ij} \le 1 - w_{j'} \quad \forall i \in I, j \in J, j' \in T_{ij} \\
& \sum_{j\in J}\sum_{k\in K} f_{jk} y_{jk} \le B \\
& x_{ij} \ge 0, w_j \in \{0,1\}, y_{jk} \in \{0,1\} \quad \forall i \in I, j \in J, k \in K
\end{aligned}
$$

where M_1 and M_2 are two large numbers. For example, we can set $M_j^1 = m$ and $M_j^2 = q_{j,s_j}$.

The first constraint allocates customers to all facilities. The second constraint states that only one facility can be built at a single location. The third constraint ensures that customers can only go to locations where facilities are built at, and the sixth constraint is the budget constraint.

We now take a closer look at the fourth and fifth constraints. The fourth constraint

doi:10.6342/NTU201703025

Customers	Facility	
	j_1	j_2
i_1	1	3
i_2	1	2
i_3	1	5

Table 3.1: An example of the preference table (p_{ij})

ensures that $w_j = 1$ if facility j is open and still has residual capacity. The fifth constraint makes sure that the customer will not go to facility j when there still are preferred options.

In order to know how the objective function and constraints work, consider the following example with two facility locations and three customers. Preference level p_{ij}s are listed in Table 3.1. For each customer, facility 2 is his favorite. By definition, we know that $T_{12} = T_{22} = T_{32} = \{\emptyset\}$ and $T_{11} = T_{21} = T_{31} = \{2\}$. Like the example above, the capacity levels of both facilities are $q_1 = q_2 = 2$, and the customer demand d_is are set to 1 for all customer locations. In this example, though all three customers prefer facility 2 to 1, the capacity of facility 2(2) is not enough to serve all customers. Therefore, some must go to facility 1.

Now we show that each customer will not go to facility 1 when facility 2 is open with residual capacity. As we show in Figure 3.2, suppose that customer 1 goes to facility 1 and customer 2 goes to facility 2 initially. Since facility 2 still has residual capacity, the left hand side of the fourth constraint is $2 - 1 = 1 > 0$ which forces $w_2 = 1$ at the right-hand side. Then according to the fifth constraint $x_{31} \leq 1 - w_2$ for facility 2 and customer 3, x_{31} cannot be greater than 0 if $2 \in T_{31}$, which is exactly when there still exists

doi:10.6342/NTU201703025

a preferred candidate location. Therefore, we know that customer 3 will go to facility 2 rather than 1.

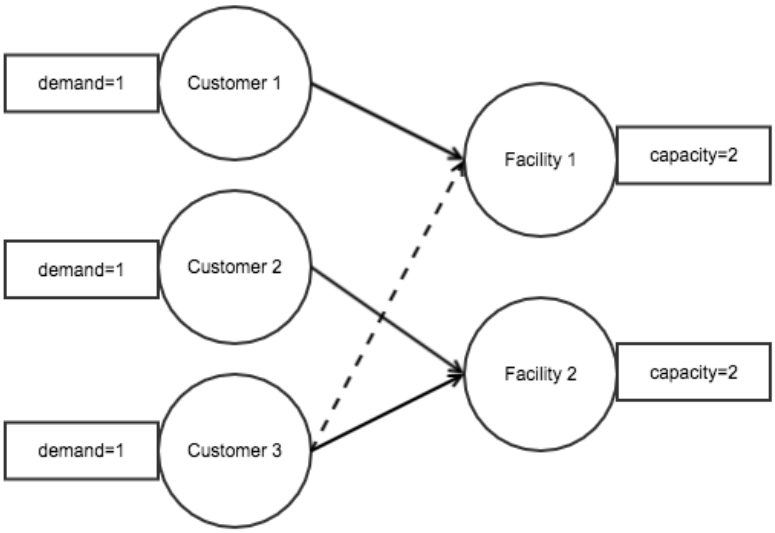


Figure 3.2: An example illustrating the preference constraints

Figure 3.2 is a feasible solution whose objective value is 3, suggesting that all three customers are served. However, this is not an optimal solution since we include a small number m_0 in the objective function. A solution with a higher total customer preference level is more likely to realize when there are solutions that can serve the same amount of people. Figure 3.3 shows an optimal solution. In this solution, customer 1 has higher preference level then 2, so he has a higher priority when choosing facilities. Customer 1 will go to facility 2, and customer 2 will go to facility i_1 as we show in Figure 3.3.

Table 3.2 introduces all the notations mentioned above.

doi:10.6342/NTU201703025

Parameters	
I	set of customers
J	set of potential facilities
d_i	customer demand in location i
p_{ij}	preference for location j of customer i
f_{jk}	construction cost of facility in j with scale k
q_{jk}	capacity of facility scale k in location j
T_{ij}	set of facilities more favorable than j for customer i; $\{j' \in J \vert p_{ij'} > p_{ij}\}$
B	construction budget
Decision variables	
x_{ij}	proportion of customer in location i going to facility j
y_{jk}	1 if building facility j with scale k or 0 otherwise

Table 3.2: List of notations

doi:10.6342/NTU201703025

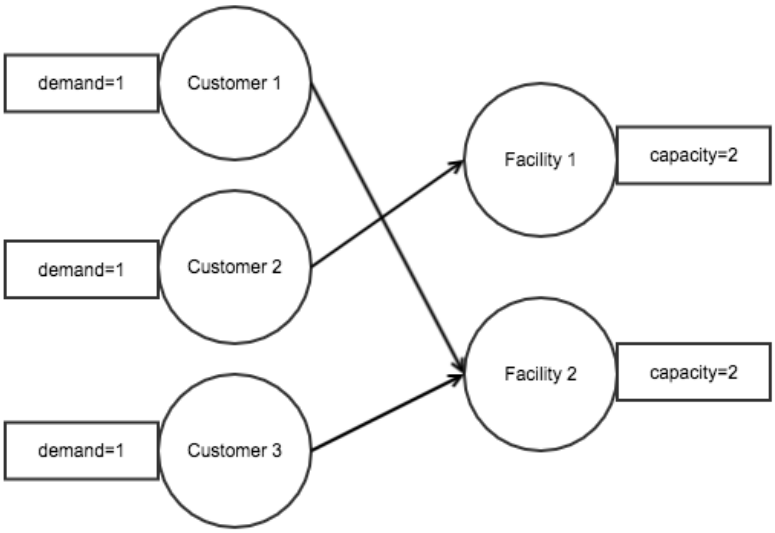


Figure 3.3: An optimal solution to the example

doi:10.6342/NTU201703025

Chapter 4

Algorithm and Analysis

In this chapter, we propose a heuristic that obtains a feasible solution to the problem defined in Chapter 3.

4.1 Algorithm

4.1.1 Overview

The algorithm that we propose is a greedy algorithm, which selects the facility with highest ratio between the increased objective value and the construction cost in each iteration. Figure 4.1 illustrates the outline of our algorithm.

In each iteration, our algorithm considers all unbuilt facilities and selects the facility and scale level with highest ratio between the objective value and the cost of building that facility. Repeat this iteration until there are no more budget left to build any new facility.

doi:10.6342/NTU201703025

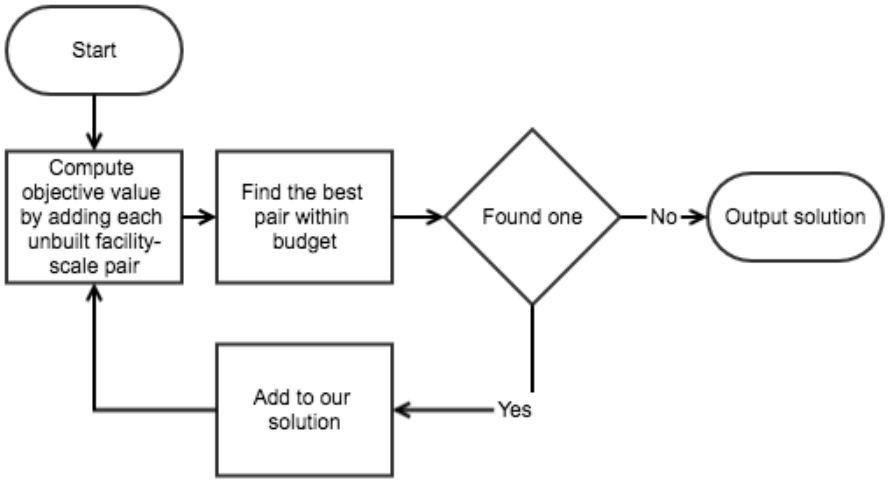


Figure 4.1: An flowchart of our algorithm

As described in Chapter 3, customers will try to maximize their preference by going to the facility with residual capacity once the facility is built. This indicates that every time we want to decide which facility to build, we have to find the amount of customers that will be served under that plan. Given a construction plan represented by $y = [y_{ij}]_{i\in I, j\in J}$, the objective value $z(y)$ can be found by solving the following program

$$
\begin{aligned}
z(y) = \max_{x_{ij}, w_j} & \sum_{i\in I}\sum_{j\in J} d_i x_{ij} \\
\text{s.t.} & \sum_{j\in J}\sum_{k\in K} f_{jk} y_{jk} \le B \\
& \sum_{j\in J} x_{ij} \le 1 \quad \forall i \in I \\
& \sum_{i\in I} x_{ij} \le M_j^1 \sum_{k\in K} y_{jk} \quad \forall j \in J \\
& \sum_{k\in K} q_{jk} y_{jk} - \sum_{i\in I} x_{ij} d_i \le M_j^2 w_j \quad \forall j \in J \\
& x_{ij} \ge 0, w_j \in \{0,1\}, y_{jk} \in \{0,1\} \quad \forall i \in I, j \in J, k \in K.
\end{aligned}
$$

doi:10.6342/NTU201703025

We can formulate the problem to a maximum flow problem.

4.1.2 Maximum flow problem

For a construction plan y, we can formulate a graph $G = \{V, E\}$ which has a similar structure as that in Figure 4.2. For every customer point and every facility point in our original problem, we create a corresponding node in the graph. A source node s is created and linked to customer point i with capacity d_i for all $i \in I$. All facility points are linked to the destination point t with capacity of q_{jk} if facility j with scale k has been built (i.e., $y_{jk} = 0$) and 0 otherwise. Capacity of edges c_{ij} connecting customer i and facility j are set to 0 if $p_{ij} < 0$ and ∞ otherwise. Let V and E be the number of nodes and edges in the graph, we can find that the cardinalities of V and E satisfy $|V| \leq m + n + 2$ and $|E| \leq mn + n + m$.

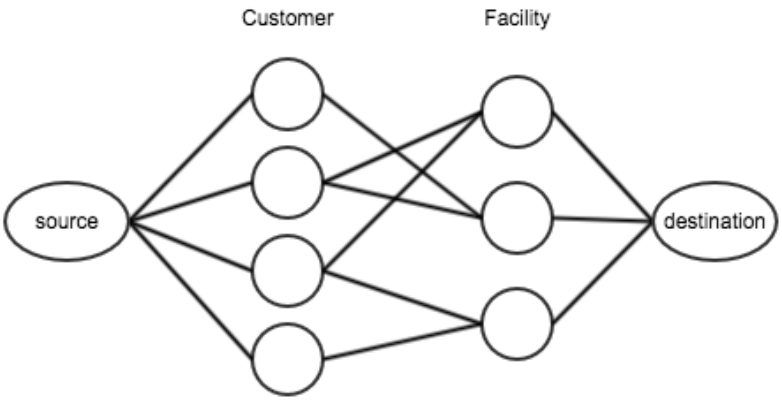


Figure 4.2: Maximum flow problem for finding the number of served customers

A maximum flow problem can be solved efficiently with a $O(|V|^3)$ bound (Goldberg and Tarjan, 1988). Ahuja et al. (1994) propose a different method on a bipartite maximum flow that can achieve a bound of $O(|V_1||E| + |V_1|^3)$, where V_1 and V_2 are the two groups

doi:10.6342/NTU201703025

of nodes in the bipartite maximum flow problem and $|V_1| \leq |V_2|$. In our problem, we have

$$O(|V_1||E| + |V_1|^3) = O(n + m + 2)^3 = O(n^3 + 3n^2m + 3nm^2 + m^3)$$

and

$$O(|V_1||E| + |V_1|^3) = O(n^3 + n^2(m + 1) + nm)$$

if the number of facility locations n is smaller than that of customers m. For this typical case, we find out that the later algorithm is more efficient for our problem with a $O(n^3 + n^2m)$ bound.

4.1.3 Greedy algorithm

With the maximum flow problem in mind, we are now ready to describe our algorithm. To calculate the number of customers that can be served by opening one addition facility at a certain scale level, the algorithm solves a maximum flow problem as we describe above. The algorithm terminates when there is no sufficient budget to build any more facility.

Define $f(y)$ as the cost of construction plan y, that is

$$f(y) = \sum_{j \in J, k \in K} f_{jk} y_{jk},$$

the pesdocode of the greedy algorithm is presented in Algorithm 1.

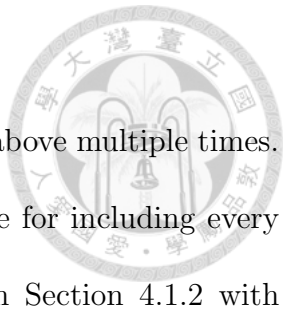
doi:10.6342/NTU201703025

Algorithm 1 Greedy Algorithm

$y \leftarrow 0$, $S \leftarrow \emptyset$
repeat
    $\max \leftarrow 0$, $(j', k') \leftarrow (0, 0)$
    for $j \in J \setminus S$ **do**
        for $k \in K$ **do**
            if $f(y) + f_{jk} \leq B$ **then**
                $y_{jk} \leftarrow 1$
                $t \leftarrow z(y)$
                if $\frac{t}{f_{jk}} > \max$ **then**
                    $\max \leftarrow \frac{t}{f_{jk}}$, $(j', k') \leftarrow (j, k)$
                end if
                $y_{jk} \leftarrow 0$
            end if
        end for
    end for
    $y_{j'k'} \leftarrow 1$, $S \leftarrow S \cup \{j'\}$
until $(j', k') = (0, 0)$
return y

doi:10.6342/NTU201703025

4.1.4 Time complexity analysis



Our algorithm requires solving the maximum flow problem defined above multiple times. In each iteration, the algorithm has to calculate the objective value for including every possible facility by solving the maximum flow problem defined in Section 4.1.2 with complexity of $O(n^3 + n^2m)$.

The algorithm will run at most m iterations and in the i^{th} iteration, the algorithm needs to solve at most $(n-i)s$ instances of the maximum flow problem, and each maximum flow instance has time complexity $O(i^3 + i^2m)$, where s is the number of facility scale levels.

Total computation time can be express as

$$O(\sum_{i=1}^{n}(n-i)s(i^3+i^2m)) = O(s\sum_{i=1}^{n}(ni^3 - i^4 + nmi^2 - i^3m))$$

$$= O(s((n-m)\sum_{i=1}^{n} i^3 - \sum_{i=1}^{n} i^4 + nm\sum_{i=1}^{n} i^2))$$

$$= O(s(n^5 + mn^4)).$$

4.1.5 Modified greedy algorithm

In larger cases with more facility locations, it may takes a long time to compute the value of maximum flow. The algorithm becomes very slow and is thus not very useful. Note that in this algorithm, we do not need to know the solution of the maximum flow. All we need to know is the objective value. This motivates us to estimate the objective value as quickly as possible rather than exactly calculate it by solving a maximum flow problem.

We know that flow on customer point is bounded by its demand and the capacity of

doi:10.6342/NTU201703025

facilities that the customer point is connected, and flow on a facility location is bounded by its capacity and the demand of all connected customers. Let I_j be the set of customers with $p_{ij} \geq 0$ for facility j and J_i be the set of facilities with $p_{ij} \geq 0$ for customer i. We define

$$P_i = \min\{\sum_{j \in J_i} q_{jk} y_{jk}, d_i\} \forall i \in I,$$

which is the "potential" of customer i. Similarly, for facility j we define its potential as

$$P_j = \min\{\sum_{i \in I_j} d_i, q_{jk} y_{jk}\} \forall j \in J.$$

For a given construction plan y, we define

$$z'(y) = \min\{\sum_{i \in I} P_i, \sum_{j \in J} P_j\}.$$

as a way to estimate the maximum flow, and we modify the greedy algorithm by using $z'(y)$ instead of $z(y)$. This reduce the complexity of our algorithm. For a given S, $z'(y)$ can be found in $O(mn)$, and the total computation time of the modified greedy algorithm is $O(smn^3)$

The pesdocode of modified greedy algorithm is listed in Algorithm 2.

4.2 An illustrative example

In this section, we use an example to understand how the algorithms work. Assume there are three customers and three facilities with single scale level. Demands, capacities and the link between them are shown in Figure 4.3. Let budget $B = 2$ and construction cost $f_{jk} = 1$ for all facilities.

doi:10.6342/NTU201703025

Algorithm 2 Modified Greedy Algorithm

$y \leftarrow 0$, $S \leftarrow \emptyset$
repeat
  $\max \leftarrow 0$, $(j', k') \leftarrow (0, 0)$
  for $j \in J \setminus S$ **do**
    for $k \in K$ **do**
      if $f(y) + f_{jk} \leq B$ **then**
        $y_{jk} \leftarrow 1$
        $t \leftarrow z'(y)$
        if $\frac{t}{f_{jk}} > \max$ **then**
          $\max \leftarrow \frac{t}{f_{jk}}$, $(j', k') \leftarrow (j, k)$
        end if
        $y_{jk} \leftarrow 0$
      end if
    end for
  end for
  $y_{j'k'} \leftarrow 1$, $S \leftarrow S \cup \{j'\}$
until $(j', k') = (0, 0)$
return y

doi:10.6342/NTU201703025

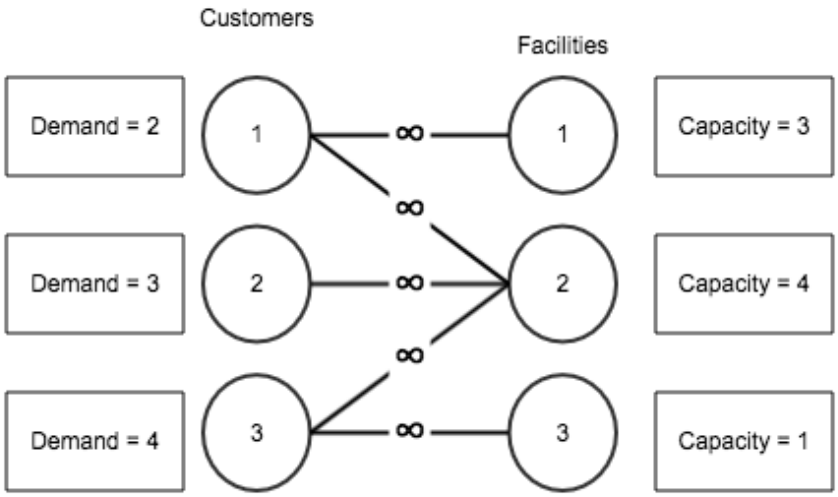


Figure 4.3: An illustrative example

4.2.1 An example of Algorithm 1

In the first iteration, we select facility 2 since by building it we can capture four people. In the next iteration, we consider the case where we add facility 1 or facility 3 by solving the maximum flow problem. Figure 4.4 shows an example of the maximum flow problem we are facing when we select facility 1 and facility 2. In this case, we can find out that the maximum flow value is 6.

Now we consider the case where we select facility 2 and facility 3, shown in Figure 4.5. We can find out that the maximum flow value is 5. By doing so we can find out that adding facility 1 to our construction plan y is better than adding facility 3. The algorithm stops and report $y = (y_{11}, y_{21}, y_{31}) = (1, 1, 0)$ here since we cannot add any more facility and not exceed budget. The objective value of this solution is 6.

doi:10.6342/NTU201703025

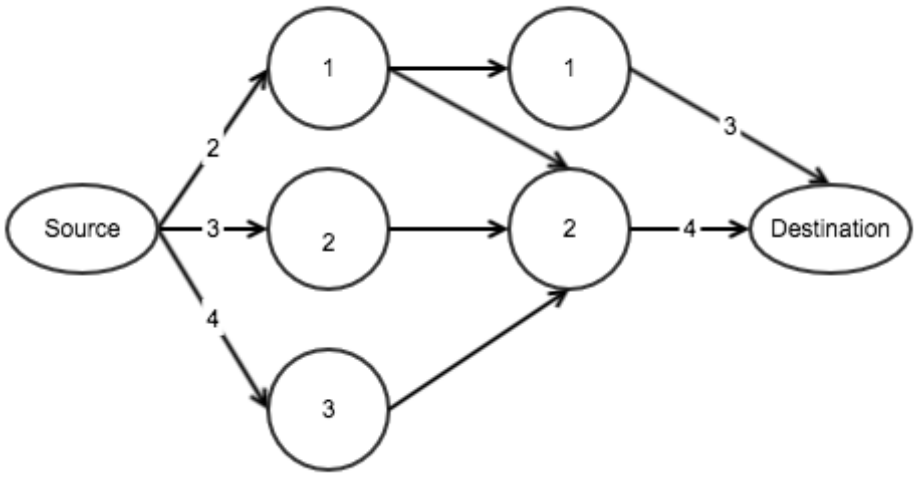


Figure 4.4: An example of maximum flow problem when building facility 1 and 2

4.2.2 An example of Algorithm 2

Algorithm 2 uses "potential" P_i and P_j to estimate the objective value of maximum flow problem. If we select facility 1 and facility 2 shown in Figure 4.4. Potentials for customers are $P_1 = \min\{2, 7\} = 2$, $P_2 = \min\{3, 4\} = 3$ and $P_3 = \min\{4, 4\} = 4$ respectively, on the other hand, potentials for facilities are $P_1 = \min\{2, 3\} = 2$ and $P_2 = \min\{9, 4\} = 4$. The estimation value is $\min\{\sum_{i\in I} P_i, \sum_{j\in J} P_j\} = \min\{2 + 3 + 4, 2 + 4\} = 6$

In the case describes in Figure 4.5, potentials for customers are $P_1 = \min\{2, 4\} = 2$, $P_2 = \min\{3, 4\} = 3$ and $P_3 = \min\{4, 5\} = 4$ respectively, on the other hand, potentials for facilities are $P_2 = \min\{9, 4\} = 4$ and $P_3 = \min\{4, 1\} = 1$. The estimation value is $\min\{\sum_{i\in I} P_i, \sum_{j\in J} P_j\} = \min\{2 + 3 + 4, 4 + 1\} = 5$

In this example, the estimation of maximum flow value and the real maximum flow value are the same, thus the output of both algorithms will be identical. However, this may not hold always hold. In some cases the estimation may differ from the maximum

doi:10.6342/NTU201703025

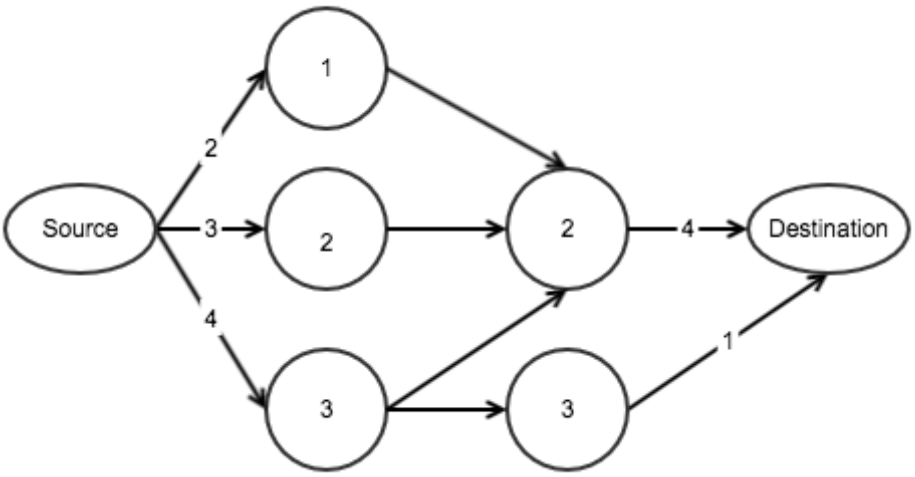


Figure 4.5: An example of maximum flow problem when building facility 2 and 3

flow value, this can make the output of these two algorithms different.

4.3 Worst-case performance analysis for some special cases

The problem defined in Chapter 3 can be simplified in two ways: We can relax the capacity constraint or we can make all construction costs homogeneous. On one hand, by relaxing the capacity constraint, we set $q_{jk} = \infty$. When there is no capacity constraint, the scale of facility has no meaning so we can say that under this circumstance there will be only one scale level. On the other hand, when we want to make the construction costs of all facilities the same, without lost of generality we set $f_{jk} = 1$.

With these two ways of specialization, we can formulate some special cases shown in Figure 4.6. Each box represents a special case of the model. In this section, we are going

doi:10.6342/NTU201703025

to perform the worst-case performance analysis on these special cases.

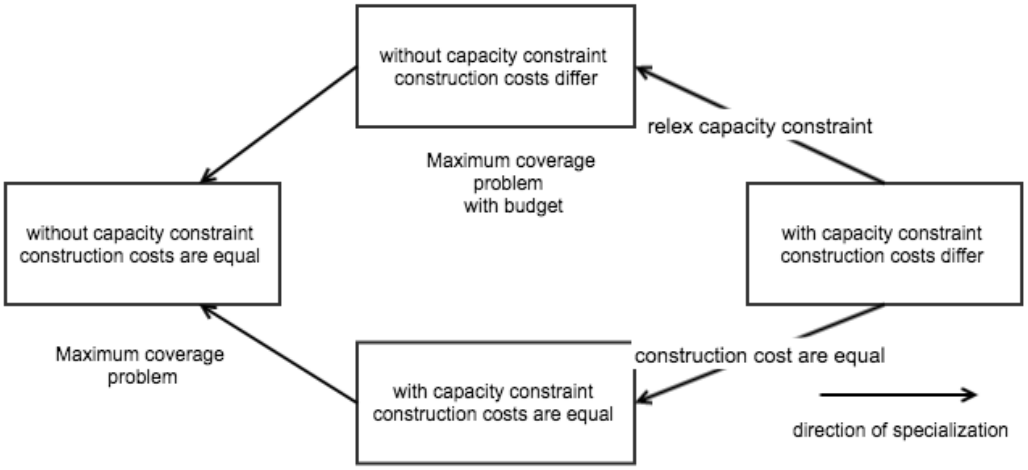


Figure 4.6: Relationship among the special cases

4.3.1 Weighted maximum cover problem

A maximum cover problem is defined as follows: For a given number k and some sets with common elements, the goal is to select at most k sets such that the union of selected sets has maximum size.

When the elements have weights on them, the problem is called weighted maximum cover problem (WMCP). The objective of WMCP is to maximize total weights collected by the selected k sets. By relaxing capacity constraint and set $f_{jk} = 1$, our problem becomes a WMCP.

In this case, our algorithm does not need to solve the maximum flow problem since there are no capacity constraint. According to Hochbaum and Pathria (1994), both our greedy algorithm and modified greedy algorithm have an approximation factor $1 - \frac{1}{e} =$

doi:10.6342/NTU201703025

0.632.

4.3.2 Weighted maximum cover problem with a budget constraint

In our model, there are construction costs when building facilities. The sum of total construction costs will not exceed the budget B. It is similar to the case where a WMCP but with costs for each set. Since there is no capacity for each facility, we do not need to solve the maximum flow problem. Khuller et al. (1999) propose a modified greedy algorithm with approximation factor of $1 - \sqrt{\frac{1}{e}}$. In each iteration, the algorithm picks the facility with highest ratio between weights of uncovered set and the weight of set and repeat until there are no residual budget for building new facilities. At the end of iterations where there is no more budget left for another set, report sets selected by the greedy iterations or the set that contains most element weights alone.

4.3.3 Weighted maximum cover problem with capacity constraints

Now we put the capacity constraint back to the model. Every facility has a maximum capacity. Once the facility has no residual capacity, people can no longer go to this facility.

Proposition 1. *Our algorithm solves WMCP with capacity constraint with approximation factor of* $1 - \frac{1}{e}$.

doi:10.6342/NTU201703025

Proof. Let y^A and y^B be two construction plans. As we build more facility, we can find out that the objective value will not decrease. That is

$$z(y^A) \leq z(y^B) \quad \forall y^A < y^B$$

We then define e_{jk} as an n by s matrix where the intersection of the j^{th} row and k^{th} column is 1 and all others are 0. When we add a new facility to set y^A and y^B, it is obvious that adding facility to y^A can increase the objective value more than adding facility to y^B. That is

$$z(y^A + e_{jk}) - z(y^A) \geq z(y^B + e_{jk}) - z(y^B) \quad \forall y^A < y^B, \quad (j,k) \in J \times K.$$

From the above equations, we can find out that the objective function is a monotone and submodular function. According to Nemhauser et al. (1978), for a monotone submodular function, our greedy algorithm has an approximation factor $1 - \frac{1}{e}$. □

doi:10.6342/NTU201703025

Chapter 5

Numerical Study

5.1 Experiment setting

In order to find out the performance of the algorithm, we compare the result with optimal solution in different cases. Four different factors are adopted to observe the performances under different circumstances. First factor is the size of problem. $n = 10$, $m = 20$ as the small size. $n = 30$, $m = 60$ as the medium size and $n = 100$, $m = 200$ as the large size. The second factor is the number of scales each facility has. We consider two scenarios where all facilities have only one scale and facilities having three scales.

Third factor is about preference with five scenarios. One of the scenarios is random preference. In other scenarios, every facility location and customer have a location on a two-dimensional coordinate. Customers are placed uniformly on the coordinate while facilities distributed uniformly or normally. For every customers and facilities with distance d_{ij}, we set $p_{ij} = \frac{1}{d_{ij}}$. In each of the above cases we conduct two more scenarios

doi:10.6342/NTU201703025

which indicates higher and lower preference level. In scenarios with higher preference level, there are more positive preference meaning that customers have more option when considering where to go. Define $p_{0.3}$ as the 30^{th} percentile and $p_{0.7}$ as the 70^{th} percentile of all preferences. In cases of higher preference level, preference $p_{ij} \leftarrow p_{ij} - p_{0.3}$ making some preference of customer i to facility j negative. In cases of lower preference level, $p_{0.7}$ is used, making more negative preferences.

The last factor is about capacity and budget. There are two cases for capacity, small and big. Same two cases also applies on budget. Together there are four combinations in this factor, however we believe that the scenario of high budget and big facility capacity will lead to building most of the facilities. So we remove this scenario and thus there are three scenarios left for this factor.

The four factors generate $3 \times 2 \times 5 \times 3 = 90$ scenarios together, each of which we generate 100 instances. We solve the problems with solver CPLEX running on a personal computer with Windows 10, 8GB RAM and Intel i7-3770 3.4GHz CPU. We use Java 1.8.0 to implement our algorithm and invoke CPLEX.

CPLEX cannot solve scenarios in large size within a tolerable time since the problem is an ILP (integer linear programming problem). We relax the problem to an LP (linear programming problem) instead to get an upper bound of the objective value.

5.2 Benchmark algorithm

In order to find out the performance of our algorithm, we implement a genetic algorithm as follows. First, we randomly create a pool of 100 feasible solutions. In each iteration

doi:10.6342/NTU201703025

we apply the tournament selection method (Miller et al., 1995). We randomly select five solutions and find the best two solutions among them. Then we perform a crossover on the selected two solution by randomly select a cross-point which divide the selected solutions to head part and tail part. Make two new feasible solutions with one's head connect to the tail of another selected solution.

The two new solutions is given a 10% chance to mutate. When mutation happens, one of the facilities will change its status from not building to building and vice versa. Finally in the end of each iteration, we compare the two worst solutions form the original pool and the new solutions, find the better two solution and add them back to the pool. This step makes the solutions in the pool better in each iteration.

This iteration will repeat for 2000 times. Then the algorithm will report the best solution in the pool.

5.3 Solution performance

We now compare the result of our algorithm with optimal solution and genetic algorithm to find out how each factor influence the performance of our algorithm. All computation results are shown in Appendix A. Each row represents a scenario. In first column, L stands for large scale, M stands for medium scale and S stands for small scale. Second column is the number of scale levels.

In third column, H stands for higher preference level and L stands for lower preference level. N and U denotes the distribution of facilities, they stands for normal distribution and uniform distribution respectively. R stands for random preference levels.

doi:10.6342/NTU201703025

For the fourth column, "L,T" stands for loose capacity and tight budget; "T,L" stands for tight budget and loose budget; and "T,T" means both capacity and budget are tight.

In Table 5.1 we found out that the average performance of our algorithm becomes better as problem size increases. This is caused by the following fact: as problem size grows, an error in our algorithm will have less impact on the objective value. So the performance of larger problem size is better than the smaller size problem.

Problem Size	Average		Minimum	
	$\frac{z^{\mathrm{ALG}}}{z^*}$	$\frac{z^{\mathrm{GA}}}{z^*}$	$\frac{z^{\mathrm{ALG}}}{z^*}$	$\frac{z^{\mathrm{GA}}}{z^*}$
Small	0.939	0.733	0.543	0.430
Medium	0.981	0.653	0.809	0.402
Large	0.982	0.571	0.881	0.349

Table 5.1: Impact of the instance size

For the genetic algorithm, performance gets worse as the problem size grows. The main reason is that when the problems become larger, there are more combination of facilities to build which makes GA difficult to deliver a better solution.

Table 5.2 shows that the performance of our algorithm is better when there is only one scale. When facility has multiple scales, the problem is more complicated with more decision variables and more potential facilities. Moreover, facilities can only have one scale. Once a scale is picked, the location cannot be used in the future. So this increase the chances that our algorithm outputs a solution that is not as good as the solution from problems with single facility scale.

In table 5.3 we can observe that the distribution of facility location has no impact

doi:10.6342/NTU201703025

Facility scale	Average		Minimum	
	$\frac{z^{\mathrm{ALG}}}{z^*}$	$\frac{z^{\mathrm{GA}}}{z^*}$	$\frac{z^{\mathrm{ALG}}}{z^*}$	$\frac{z^{\mathrm{GA}}}{z^*}$
Single scale	0.970	0.595	0.543	0.349
Three scales	0.964	0.71	0.592	0.367

Table 5.2: Impact of the number of scale levels

in the performance of our algorithm. However, our algorithm performs better when customers are willing to go to more facility. This is because when customers have more options, even we select a sub-optimal facility, we can still collect some customers.

Preference and facility distribution	Average		Minimum	
	$\frac{z^{\mathrm{ALG}}}{z^*}$	$\frac{z^{\mathrm{GA}}}{z^*}$	$\frac{z^{\mathrm{ALG}}}{z^*}$	$\frac{z^{\mathrm{GA}}}{z^*}$
Dense connection with normal distribution	0.952	0.641	0.56	0.358
Dense connection with uniform distribution	0.951	0.632	0.54	0.35
Sparse connection with normal distribution	0.978	0.665	0.748	0.360
Sparse connection with normal distribution	0.979	0.666	0.794	0.364
random preference	0.976	0.660	0.764	0.353

Table 5.3: Impact of the distribution of preferences

Table 5.4 shows that performance of our algorithm is better when budget is tight. In this case, only a handful of facilities will be built. Every facility is more important than the facilities in the loose budget scenario. It is easier for our algorithm to come up with a better solution.

doi:10.6342/NTU201703025

Capacity and Budget	Average		Minimum	
	$\frac{z^{\text{ALG}}}{z^*}$	$\frac{z^{\text{GA}}}{z^*}$	$\frac{z^{\text{ALG}}}{z^*}$	$\frac{z^{\text{GA}}}{z^*}$
Loose capacity with tight budget	0.978	0.732	0.56	0.45
Tight capacity with loose budget	0.944	0.49	0.592	0.349
Tight capacity with tight budget	0.980	0.735	0.543	0.453

Table 5.4: Impact of the capacity and budget constraints

5.4 Time complexity

In this section we want to find out how the problem size influences computing time. All the other factors are set as follows: Scale levels are set to one. Performance are randomly distributed. For capacity and budget constraint, we randomly select form "L,T", "T,L" and "T,T" scenarios. For each problem size we produce 100 instance and calculate the average computation time.

Table 5.5 and Figure 5.1 are the computation time using different method to solve the problem with different customer location size m by setting $n = 10$. We can see that both IP solution and GA takes more time to solve problems than Algorithm 2. If we take a closer look on the computation time of Algorithm 2 with different m we can find out that as m gets larger, computation will take slightly more time. However, differences are not significant.

Table 5.6 and Figure 5.2 are the computation time using different methods in problem with different facility location candidate size n while setting $m = 200$. However, an integer solution can not be obtained in reasonable time so we use the linear relaxation as

doi:10.6342/NTU201703025

m	IP solution	Algorithm 2	GA
10	39.06	50.42	71.96
20	50.68	50.46	63.52
30	58.92	50.38	68.54
40	69.64	50.6	76.54
50	75.74	50.64	79.68
60	84.94	50.48	83.84
70	92.92	50.58	89.34
80	86.62	50.56	91.2
90	95.64	50.66	94.88
100	94.22	50.76	98.28

Table 5.5: Computation time of different m when $n = 10$ (in milliseconds)

an alternative baseline.

Computation time needed for getting a linear relaxation solution for problems with larger n grows rapidly. It is less likely to find an integer solution as the problem grows since integer solution is harder to find comparing to its linear relaxation. For a larger problem size, our algorithm can indeed find a solution in a short time.

Figure 5.3 shows the computation time of Algorithm 2. It fits a cubic polynomial function of n with $R^2 = 0.99$. In Chapter 4 we claim that Algorithm 2 has a time complexity of $O(smn^3)$. Figure 5.3 helps us verify this result.

doi:10.6342/NTU201703025

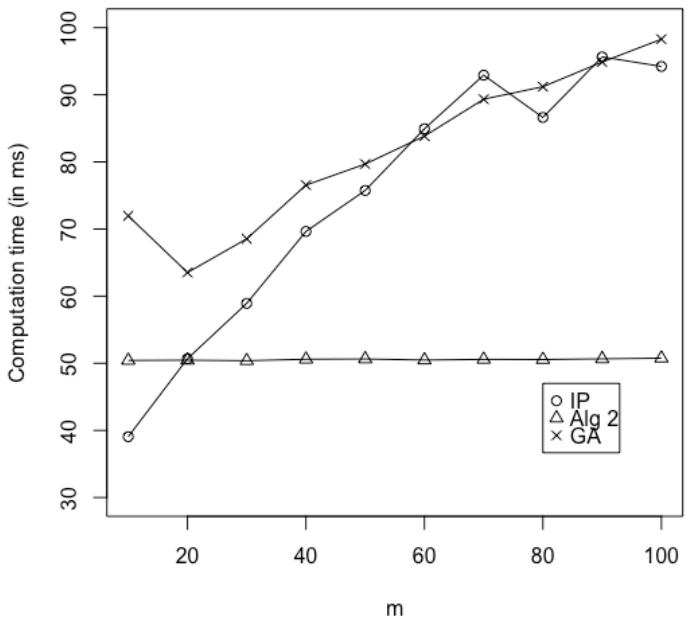


Figure 5.1: Computation time of different m when $n = 10$ (in milliseconds)

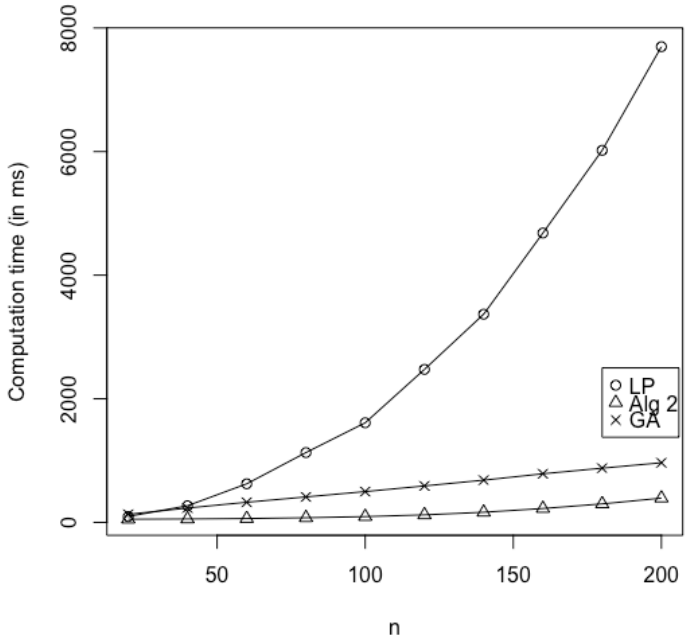


Figure 5.2: Computation time of different n when $m = 200$ (in milliseconds)

doi:10.6342/NTU201703025

n	LP solution	Algorithm 2	GA
20	94.7	50.74	130.48
40	269.08	55.18	232.22
60	623.92	62.22	325.48
80	1129.68	75.02	411.74
100	1611.46	93.26	498.22
120	2475.62	121.74	589.8
140	3368.6	163.88	683.26
160	4682.34	224.48	786.44
180	6020.5	298.98	877.1
200	7697.9	392.38	964.92

Table 5.6: Computation time of different n when $m = 200$ (in milliseconds)

doi:10.6342/NTU201703025

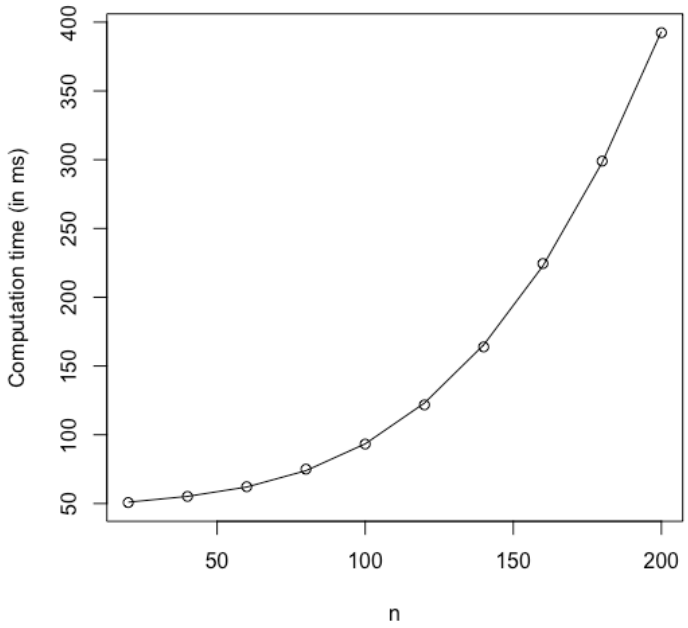


Figure 5.3: Computation time of Algorithm 2 of different n (in milliseconds)

doi:10.6342/NTU201703025

Chapter 6

Conclusions

In this study we describe a special facility location problem where customers moves according to their own will. The decision maker cannot decide how they moves. Afterwords, we propose a greedy based heuristic algorithm to solve this problem. Our algorithm gives performance guarantee in some special cases and we propose a more efficient way to estimate the number of customer that will be captured according to a given construction plan. To test the applicability, a numerical analysis is conducted. In numerical studies, we find out that our algorithm have good performance comparing to a genetic algorithm. Although our algorithm cannot give performance guarantee to all the cases, we still catch the most important effect about customer preference. The algorithm also helps us understand more about the problem.

doi:10.6342/NTU201703025

Appendix A

Results of Numerical Experiments

doi:10.6342/NTU201703025

Problem size	Scale levels	Preference distribution	Capacity and budget	Average $\frac{z^{\mathrm{ALG}}}{z^*}$	Average $\frac{z^{\mathrm{GA}}}{z^*}$	Minimum $\frac{z^{\mathrm{ALG}}}{z^*}$	Minimum $\frac{z^{\mathrm{GA}}}{z^*}$
L	1	H,N	L,T	0.999	0.549	0.973	0.492
L	1	H,N	T,L	0.977	0.391	0.955	0.359
L	1	H,N	T,T	0.999	0.545	0.967	0.474
L	1	H,U	L,T	0.999	0.547	0.971	0.477
L	1	H,U	T,L	0.977	0.388	0.957	0.350
L	1	H,U	T,T	0.999	0.546	0.981	0.476
L	1	L,N	L,T	0.999	0.545	0.963	0.479
L	1	L,N	T,L	0.977	0.390	0.953	0.361
L	1	L,N	T,T	0.997	0.547	0.919	0.456
L	1	L,U	L,T	0.998	0.549	0.928	0.475
L	1	L,U	T,L	0.975	0.394	0.958	0.364
L	1	L,U	T,T	0.999	0.550	0.973	0.480
L	1	R	L,T	0.998	0.549	0.978	0.490
L	1	R	T,L	0.977	0.387	0.954	0.354
L	1	R	T,T	0.999	0.545	0.967	0.484
L	3	H,N	L,T	0.989	0.770	0.911	0.709
L	3	H,N	T,L	0.927	0.412	0.889	0.382
L	3	H,N	T,T	0.988	0.764	0.934	0.700
L	3	H,U	L,T	0.999	0.763	0.966	0.708
L	3	H,U	T,L	0.923	0.413	0.881	0.367
L	3	H,U	T,T	0.998	0.761	0.978	0.704
L	3	L,N	L,T	0.998	0.765	0.996	0.684
L	3	L,N	T,L	0.924	0.412	0.898	0.378
L	3	L,N	T,T	1.000	0.767	1.000	0.682
L	3	L,U	L,T	1.000	0.768	1.000	0.714
L	3	L,U	T,L	0.927	0.411	0.896	0.377
L	3	L,U	T,T	1.000	0.769	1.000	0.708
L	3	R	L,T	1.000	0.768	1.000	0.712
L	3	R	T,L	0.927	0.413	0.899	0.384
L	3	R	T,T	1.000	0.769	1.000	0.712

Table A.1: Performances of our algorithm and GA in large problem size

doi:10.6342/NTU201703025

Problem size	Scale levels	Preference distribution	Capacity and budget	Average $\frac{z^{\mathrm{ALG}}}{z^*}$	Average $\frac{z^{\mathrm{GA}}}{z^*}$	Minimum $\frac{z^{\mathrm{ALG}}}{z^*}$	Minimum $\frac{z^{\mathrm{GA}}}{z^*}$
M	1	H,N	L,T	0.997	0.663	0.900	0.513
M	1	H,N	T,L	0.979	0.474	0.923	0.415
M	1	H,N	T,T	0.996	0.658	0.848	0.531
M	1	H,U	L,T	0.990	0.656	0.907	0.542
M	1	H,U	T,L	0.980	0.476	0.938	0.403
M	1	H,U	T,T	0.993	0.640	0.909	0.492
M	1	L,N	L,T	0.991	0.657	0.893	0.521
M	1	L,N	T,L	0.979	0.472	0.941	0.402
M	1	L,N	T,T	0.990	0.658	0.895	0.550
M	1	L,U	L,T	0.991	0.657	0.876	0.531
M	1	L,U	T,L	0.978	0.468	0.926	0.404
M	1	L,U	T,T	0.991	0.659	0.886	0.571
M	1	R	L,T	0.990	0.654	0.907	0.511
M	1	R	T,L	0.980	0.472	0.949	0.411
M	1	R	T,T	0.990	0.651	0.868	0.544
M	3	H,N	L,T	1.002	0.818	1.000	0.635
M	3	H,N	T,L	0.925	0.479	0.870	0.411
M	3	H,N	T,T	1.003	0.814	1.000	0.700
M	3	H,U	L,T	1.000	0.810	1.000	0.628
M	3	H,U	T,L	0.923	0.477	0.809	0.420
M	3	H,U	T,T	1.000	0.816	1.000	0.576
M	3	L,N	L,T	1.000	0.838	1.000	0.758
M	3	L,N	T,L	0.926	0.483	0.850	0.424
M	3	L,N	T,T	1.000	0.844	1.000	0.747
M	3	L,U	L,T	1.000	0.843	1.000	0.729
M	3	L,U	T,L	0.921	0.480	0.840	0.429
M	3	L,U	T,T	1.000	0.845	0.955	0.727
M	3	R	L,T	1.000	0.834	1.000	0.738
M	3	R	T,L	0.925	0.476	0.855	0.421
M	3	R	T,T	1.000	0.837	1.000	0.743

Table A.2: Performances of our algorithm and GA in medium problem size

doi:10.6342/NTU201703025

Problem size	Scale levels	Preference distribution	Capacity and budget	Average $\frac{z^{\mathrm{ALG}}}{z^*}$	Average $\frac{z^{\mathrm{GA}}}{z^*}$	Minimum $\frac{z^{\mathrm{ALG}}}{z^*}$	Minimum $\frac{z^{\mathrm{GA}}}{z^*}$
S	1	H,N	L,T	0.814	0.668	0.561	0.449
S	1	H,N	T,L	0.902	0.589	0.654	0.435
S	1	H,N	T,T	0.918	0.746	0.776	0.518
S	1	H,U	L,T	0.869	0.697	0.632	0.513
S	1	H,U	T,L	0.906	0.571	0.733	0.431
S	1	H,U	T,T	0.840	0.674	0.544	0.453
S	1	L,N	L,T	0.963	0.779	0.748	0.588
S	1	L,N	T,L	0.987	0.605	0.930	0.493
S	1	L,N	T,T	0.971	0.787	0.748	0.556
S	1	L,U	L,T	0.975	0.771	0.849	0.593
S	1	L,U	T,L	0.985	0.613	0.883	0.480
S	1	L,U	T,T	0.968	0.788	0.794	0.627
S	1	R	L,T	0.957	0.773	0.764	0.633
S	1	R	T,L	0.977	0.606	0.884	0.499
S	1	R	T,T	0.961	0.776	0.785	0.591
S	3	H,N	L,T	0.921	0.802	0.737	0.585
S	3	H,N	T,L	0.880	0.614	0.730	0.457
S	3	H,N	T,T	0.929	0.794	0.734	0.575
S	3	H,U	L,T	0.929	0.770	0.695	0.577
S	3	H,U	T,L	0.881	0.589	0.592	0.433
S	3	H,U	T,T	0.925	0.785	0.640	0.487
S	3	L,N	L,T	0.994	0.905	0.850	0.719
S	3	L,N	T,L	0.927	0.599	0.806	0.488
S	3	L,N	T,T	0.992	0.921	0.796	0.743
S	3	L,U	L,T	0.993	0.918	0.861	0.754
S	3	L,U	T,L	0.935	0.594	0.817	0.488
S	3	L,U	T,T	0.993	0.913	0.884	0.702
S	3	R	L,T	0.987	0.882	0.789	0.693
S	3	R	T,L	0.926	0.583	0.780	0.443
S	3	R	T,T	0.979	0.894	0.817	0.692

Table A.3: Performances of our algorithm and GA in small problem size

doi:10.6342/NTU201703025

Bibliography

Aboolian, R., O. Berman, D. Krass. 2007. Competitive facility location model with concave demand. *European Journal of Operational Research* **18**(2) 598–619.

Ahuja, Ravindra K., James B. Orlin, Clifford Stein, Robert E. Tarjan. 1994. Improved algorithms for bipartite network flow. *SIAM Journal on Computing* **23**(5) 906–933.

Camacho-Vallejo, J., A. Cordero-Franco, R.G. Gonzlez-Ramrez. 2014. Solving the bilevel facility location problem under preferences by a stackelberg-evolutionary algorithm. *Mathematical Problems in Engineering* **2014**.

Daskin, M.S. 2013. *Network and Discrete Location: Models, Algorithms, and Applications*. Wiley, USA.

Francis, R.L., J.A. White. 1974. *Facility layout and location: an analytical approach*. International Industrial and Systems Engineering Series, Prentice-Hall.

Goldberg, Andrew V., Robert E. Tarjan. 1988. A new approach to the maximum-flow problem. *Journal of the ACM* **35**(4) 921–940.

Hanjoul, P., D. Petters. 1987. A facility location problem with clients' preference orderings. *Regional Science and Urban Economics* **17** 451–473.

doi:10.6342/NTU201703025

Hochbaum, D.S, A. Pathria. 1994. Analysis of the greedy approach in covering problems. Unpublished thesis.

Khuller, Samir, Anna Moss, Joseph (Seffi) Naor. 1999. The budgeted maximum coverage problem. *Information Processing Letters* **70**(1) 39 – 45.

Lee, J.M., Y.H. Lee. 2012. Facility location and scale decision problem with customer preference. *Computers and Industrial Engineering* **63** 184–191.

Liao, W.H. 2016. A service facility location model with endogenous consumer demands. Master's thesis, National Taiwan University.

Miller, Brad L, David E Goldberg, et al. 1995. Genetic algorithms, tournament selection, and the effects of noise. *Complex systems* **9**(3) 193–212.

Nemhauser, G. L., L. A. Wolsey, M. L. Fisher. 1978. An analysis of approximations for maximizing submodular set functions—i. *Mathematical Programming* **14**(1) 265–294.

Wagner, J.L., L.M. Falkson. 1975. The optimal nodal location of public facilities with price-sensitive demand. *Geographical Analysis* **7** 69–83.

Williamson, D.P., D.B. Shmoys. 2011. *The Design of Approximation Algorithms*. Cambridge University Press, London, UK.

Wu, T.H., J.N. Lin. 2001. Sovling the competitive discretionary service facility location problem. *European Journal of Operational Research* **144** 366–378.

doi:10.6342/NTU201703025