

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



需求具內生性的服務設施位置問題

A Service Facility Location Model with
Endogenous Consumer Demands

廖偉宏

Wei-Hung Liao

指導教授：孔令傑 博士

Adviser: Ling-Chieh Kung, Ph.D.

中華民國 105 年 7 月

July, 2016

國立臺灣大學(碩、博)士學位論文
口試委員會審定書



題目：A Service Facility Location Model with
Endogenous Consumer Demands

本論文係廖偉宏君(學號 R03725035)在國立臺灣大學
資訊管理學系、所完成之(博、碩)士學位論文，於民國 105
年 6 月 21 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：

林妙聰

孔令傑

魯百鴻

所 長：

李善坤

謝辭

完成一篇論文永遠都不是一件簡單的事情，尤其是畢業論文，而要完成困難的事總是需要許多人的幫助，因此我要感謝所有幫助我順利完成論文的人。

首先，我要感謝我的指導教授，孔令傑教授，他帶領從零開始的我進入他的研究領域，補充了所有既專業卻又必須的知識，他帶著我學習做研究所需的技巧，更重要的是他帶給我面對人生該有的態度，他為了自己的夢想毅然決然放棄在國外更優渥的環境，選擇回來臺灣為教育與研究付出，他不只是我的論文指導教授，他也是我的人生導師。

接下來我要感謝我的實驗室夥伴們，不論是同屆的夥伴還是下一屆的學弟妹，都在我與論文奮鬥之時給予不同的幫助，不論是在口試前夕幫助我完善報告，又或是在我無法承受如山高的壓力之下陪伴我舒緩壓力，這些種種都是我能夠支撐到現在的重要心靈支柱，畢竟在研究所相處時間最多的就是一起在實驗室奮鬥的你們了。

再來要感謝同屆同學們，我們一起吃飯，一起奮鬥，一起參與各種大大小小的活動，一起分享彼此研究的困難，儘管我們在畢旅才正式認識彼此，但是一起奮鬥的革命情感是永遠無法抹去的。

最後我要感謝我的父母，支持我上研究所的決定，支持我在臺北的開銷，支持我所需要的任何東西，並且在家裡築起一道能夠擋住任何風雨的港灣，讓我在回到家時補充所需要的電力，回到工作崗位上能夠繼續面對接下來的挑戰。

感謝所有曾經幫助過我的人，少了你們的任何幫助我都無法完成這篇論文，感謝感謝再感謝。

廖偉宏 謹致

于台大資訊管理研究所

民國一百零五年九月

中文摘要



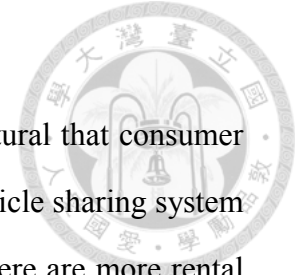
在顧客透過設施據點取得服務或商品的商業模式下，設施據點數量與位置影響消費者需求是一個很正常的現象，消費者周遭的設施據點越多；代表消費者能夠越便利的取得服務或商品，同時有意願使用服務或產品的顧客數量也會增加。有趣的是，設置一處設施據點不只會影響需求，還會影響其他設施據點對需求的影響，以 YouBike 為例，在捷運發達的台北，很多使用者利用 YouBike 作為從捷運站點到實際目的地之間的短程交通工具，因此若在捷運站點與工作區、學校、居住地之間皆設置站點，將會帶來額外的網路效益。

在本研究中，我們研究一個供應商該選擇哪些位置設置站點才能夠使其利益最大化，他需要考慮到的主要因素有兩個，分別為單獨效益以及網路效益，我們將這兩個效益當作一個凹函數的參數使我們的模型具有邊際效益遞減的效果，最後考慮各站點的建造成本，我們得到一個不可分的非線性整數規劃問題。

之後我們證明背包問題為此問題的子問題得到此問題為弱 NP-困難問題，並提出 ARSA、NGA 兩個演算法。我們由著名的背包問題演算法中獲得靈感，設計出 ARSA 演算法，並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。NGA 演算法則是簡單的貪婪演算法。最後我們透過數值分析驗證了兩種演算法的表現、穩定性、計算時間。

關鍵字：設施位置、網路效應、內生需求、近似演算法

Thesis Abstract



For those facilities that serve end consumers directly, it is natural that consumer demands are affected by the number and locations of facilities. Vehicle sharing system provide a good illustration, as more users join the system when there are more rental sites. Interestingly, opening a facility not only affects customer demands directly but also changes how other facilities affect consumer demands. For example, for a typical bike sharing system, users often travel from the subway stations to their offices, schools, and home. When there are rental sites at both sides, an additional network effect emerges.

In this study, we investigate the problem of a profit maximizing retailer in selecting a set of facilities to build from a given set of locations. We assume that the retailer needs to consider two major types of effects: (1) the stand-alone benefit of a single facility and (2) the network benefit between two facilities. The sum of these two benefits will then be input into a nondecreasing concave function to capture the diminishing marginal benefit property. By considering the overall benefit and total cost of building facilities, the retailer decides where to build facilities to maximize her profit. The problem is then formulated as a nonseparable nonlinear integer program.

We first prove that the problem is weakly NP-hard. As one of the most common method to approach NP-hard problems is to develop heuristic algorithms, we propose two algorithms. The first one, which is based on relaxing the integer constraints, is called the approximation-relaxation-sorting algorithm (ARSA). The second one, which is called the naive greedy algorithm (NGA), is a straightforward greedy algorithm. We show that ARSA has different worst-case performance guarantees for some special cases of our general problem. We then study the average performance of the two algorithms in various scenarios through numerical experiments.

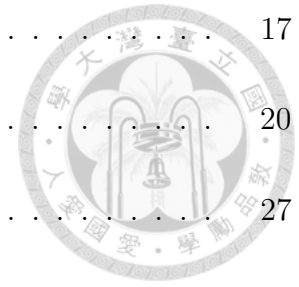
Keywords: Facility location, network effect, endogenous consumer demands, approximation algorithm.



Contents

1	Introduction	1
1.1	Background and motivation	1
1.2	Research objectives	3
1.3	Research plan	3
2	Literature Review	5
2.1	Facility location problems	5
2.2	Submodular function maximization	7
2.3	Approximation algorithms	8
3	Problem Description and Formulation	11
3.1	Model	11
3.2	NP-hardness	13
4	Analysis	17
4.1	Approximation-relaxation-sorting algorithm	17

4.1.1	The algorithm	17
4.1.2	Worst-case performance analysis	20
4.2	Naive greedy algorithm	27
5	Numerical Study	29
5.1	Solution performance	29
5.2	Time complexity	33
6	Conclusion and Future Work	37
6.1	Conclusion	37
6.2	Future work	38
A	Results of Numerical Experiments	39
	Bibliography	41





List of Figures

4.1	Flow chart of ARSA	18
5.1	Average computing time of ARSA	34





List of Tables

3.1	Notations	13
5.1	Numerical results of problem scale	31
5.2	Numerical results of stand-alone benefit distribution	32
5.3	Numerical results of network benefit distribution	32
5.4	Numerical results of function form	33
5.5	Numerical results of function form	35
A.1	The average and minimum performances of ARSA and NGA in all scenarios	40





Chapter 1

Introduction

1.1 Background and motivation

The world is full of situations that the number of facilities affect how consumers are willing to buy a product or use a service. In fact, in many cases the consumer demands may be driven up only if sufficient facilities are distributed.

Vehicle sharing systems provide a good illustration. Even though Zipcar, one of the most famous car sharing system nowadays, adopts new technologies and flexible renting plans, a consumer will become a member of Zipcar only if there are usually available cars nearby her. This happens only if there are enough Zipcar parking spaces in a city. A similar situation happened to the YouBike bicycle sharing system that is recently implemented in Taiwan. Even though renting a YouBike is completely free for the first thirty minutes when it was introduced, the system became a success only after enough rental sites have been built and enough bicycles have been supplied.

As another example, consider what consumers think when they want to buy a personal computer or laptop. While prices and functionalities are both important, a consumer would also evaluate how easy she may find a warranty station when the product is broken. The distribution of warranty stations thus affects demands. Similar stories range from opening convenience stores to compete for consumers and building charging or battery swapping stations for electronic vehicles to attract potential buyers.

Interestingly, opening a facility not only affects the consumers directly but also changes how other facilities affect consumer demands. The first factor to consider is the interdependence among facilities. The vehicle rental business is an illustrating example. Many users of public bikes travel from subway stations to their offices, schools, and home. When there are facilities at both places, e.g., a subway station and one's office, an additional source of attractiveness emerges beyond the stand-alone benefit from building each site individually.¹ This is the upside of building one facility. Nevertheless, there is also a downside of building the facility: the marginal benefit of building facilities is diminishing. When one's home/office has been surrounded by many public bike stations, building one more is not so attractive. All the aforementioned effects should be considered together when one makes the facility construction decisions.

¹It is arguably true that network benefit does not exist in any pair of two facilities, due to for example the far distance between them. In this case, the network benefit will simply be 0 between this pair of facilities.

1.2 Research objectives



In this study, we investigate the problem of a profit maximizing retailer or service provider (hereafter, retailer) in selecting a subset of facilities to build from a given set of locations.

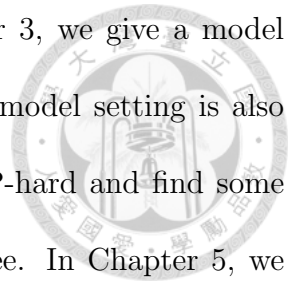
We assume that the retailer needs to consider two major types of effects: (1) the *stand-alone benefit* of a single facility and (2) the *network benefit* between two facilities. The sum of these two benefits will then be input into a nondecreasing concave function to capture the diminishing marginal benefit property. By considering the overall benefit and total cost of building facilities, the retailer decides where to build facilities to maximize her profit. The problem is then formulated as a nonseparable nonlinear integer program.

We first prove that the problem is weakly NP-hard. As one of the most common procedure to approach NP-hard problems is to develop heuristic algorithms, we propose two algorithms. The first one, which is based on relaxing the integer constraints, is called the approximation-relaxation-sorting algorithm (ARSA). The second one, which is called the naive greedy algorithm (NGA) in this study, is a straightforward greedy algorithm. We show that ARSA has different worst-case performance guarantees for some special cases of our general problem. We then study the average performance of the two algorithms in various scenarios through numerical experiments.

1.3 Research plan

In Chapter 2, we discuss some related works. In the studies that discuss the facility location problem, we focus on the works which address their problems on competitive facility location problems. We also review relevant works in the literature of submodular

function maximization and approximation algorithms. In Chapter 3, we give a model formulation for our problem. Moreover, the rationale behind our model setting is also explained. In Chapter 4, we prove that the problem is weakly NP-hard and find some conditions under which our algorithm has a performance guarantee. In Chapter 5, we present the results of a numerical study. Chapter 6 concludes.





Chapter 2

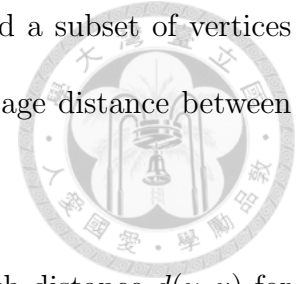
Literature Review

2.1 Facility location problems

Facility location problems have been widely studied in the past decades (Daskin, 2013; Owen and Daskin, 1998; Shen et al., 2003). Based on the formats of the objective functions and constraints, the problems are classified to several classes. Six most common classes are listed below.

1. (Set Cover) Given a set of elements U and a collection of subsets $S \subset 2^U$, the problem is to find a minimum cardinality or weighted collection of sets $C \subseteq S$ such that the union of C is U .
2. (Max Cover) Given a set of elements U and a collection of subsets $S \subset 2^U$, the problem is to find a collection of sets $C \subseteq S$ whose cardinality is no greater than k such that the weighted sum of the elements in the union of C is maximized.
3. (k -median) Given a complete undirected graph $G = (V, E)$ with distance $d(u, v)$ for

each pair of vertices $u \in V$ and $v \in V$, the problem is to find a subset of vertices $S \subset V$ with cardinality no greater than k such that the average distance between each vertex to its nearest vertex in S is minimized.



4. (k -center) Given a complete undirected graph $G = (V, E)$ with distance $d(u, v)$ for each pair of vertices $u \in V$ and $v \in V$, the problem is to find a subset of vertices $S \subset V$ with cardinality no greater than k such that the maximum distance between each vertex to its nearest vertex in S is minimized.
5. (Uncapacitated Facility Location) Given a set of potential facility sites F and a set of demand points D , the problem is to find a subset $S \subseteq F$ such that the sum of the distance between each demand point and its nearest facility in S weighted by the demand size and the total construction cost of facilities in S is minimized.
6. (Capacitated Facility Location) Given a set of potential facility sites F , a set of demand points D , the problem is to find a subset $S \subseteq F$ such that the sum of the distance between each demand point and its nearest facility in S weighted by the demand size and the total construction cost of facilities in S is minimized. Moreover, the total amount of demands assigned to a facility in S cannot exceed its capacity.

While most works in the above six categories are conducted with exogenous consumer demands, few studies have been devoted to problems with endogenous demands. As we mentioned in Chapter 1, there are situations in which the interrelationship between location decisions and demand realization cannot be ignored. This motivates the study of competitive facility location problems, see, e.g., Aboolian et al. (2007), Berman and Krass (1998), Berman and Krass (2002), and Wu and Lin (2003). In these studies, the

total market demand, as a function of the number of facilities, is generally assumed to be increasing for the market expansion effect and concave for the market cannibalization effect. In this study, we follow this stream and model the consumer demand by a concave function.



2.2 Submodular function maximization

Maximizing submodular functions is a main issue of many kinds of problems, e.g., the maximum coverage problem mentioned above. With the importance of the problem, many studies appear to propose solutions, e.g., Feige et al. (2011), Nemhauser and Wolsey (1978), and Nemhauser et al. (1978).

Nemhauser and Wolsey (1978) provide the best approximation algorithm for maximizing a submodular set function. More precisely, the problem is to choose k elements from a subset of given elements that can maximize a given submodular function. The algorithm combines the concept of exhaustion and greedy search to balance the performance and computation time. The algorithm first lists all the subsets that have $q < k$ elements. It next uses a greedy algorithm to select the remaining $k - q$ elements and takes the best one as the result. After a researcher chooses the parameter q , the algorithm's performance depends on q . Larger q makes the algorithm more precise but also more time-consuming. The changeable q give the algorithm more flexibility to fit different situations.

Nemhauser et al. (1978) give some definitions to the submodular functions and analyze different classes of approximation algorithms for submodular set function maximization, including greedy, R -step greedy, local search, linear programming, heuristic, and partial

enumeration. In particular, they prove a worst-case performance guarantee $1 - \frac{1}{e}$ of the greedy algorithm.

Feige et al. (2011) give examples to some important problems that has submodular function maximization involved. They also give approximation algorithms better than before. The algorithms have approximation factor $\frac{2}{5}$ for non-symmetric cases and $\frac{1}{2}$ for symmetric cases. They also prove that the performance on symmetric cases cannot be better unless $P = NP$. The most valuable part of these algorithms is that the algorithms are suitable for any submodular functions.

2.3 Approximation algorithms

We all know that there is no polynomial time algorithms that can solve NP-hard problems unless $P = NP$. Some researchers develop heuristic algorithms to find near-optimal solutions. Numerical studies are typically conducted to demonstrate the performance of the proposed algorithms. In most cases, researchers prefer to have a worst-case performance guarantee, i.e., a bound to the difference between the solution and the optimal solution, of a heuristic algorithm. If this is the case, the algorithm is called an approximation algorithm (Williamson and Shmoys, 2011).

Li et al. (2002) propose a new strategy to linearize the quadratic function in an assortment problems to improve the computational efficiency of the algorithm. The objective of an assortment problem is to find a big rectangle with minimum area into which a given set of small rectangles can be put without overlapping. The key step of the strategy is using a piecewise linear function to approximate the original quadratic

function which changes the original nonlinear programming problem to a mixed 0-1 linear programming problem. The new strategy is faster than those used in previous studies.

Arya et al. (2004) analyze the performance of local search on the metric k -median problem and uncapacitated facility location problem. For k -median, the traditional local search has a performance guarantee of 5. This study proposes swaps among p facilities to improve the performance guarantee to $3 + 2/p$. For the uncapacitated facility location problem, the authors prove that the bound of the local search algorithm is 3. The two results, $3 + 2/p$ and 3, are both better than the bounds in previous studies.

Aboolian et al. (2007) try to solve a competitive facility location problem with concave demand using approximation algorithms. After the formulation, they aware that the problem is an integer nonlinear problem. They replace the function with a piecewise linear function to transform the problem to an integer linear problem. They design three procedures to get the solution of the problem. The first procedure is an exact procedure, using branch and bound to directly get an exact answer of the original problem. The second procedure is “ α -approximate,” using an integer programming solver to solve the problem with the piecewise linear objective function. While the piecewise linear function differs from the original function by a relative error at most α , the procedure also has error α compare to the optimal solution. The third procedure is greedy. The procedure is shown to have a bound $1 + \frac{1}{e}$ by applying a result in Nemhauser et al. (1978). They also demonstrate that the bound is tight.





Chapter 3

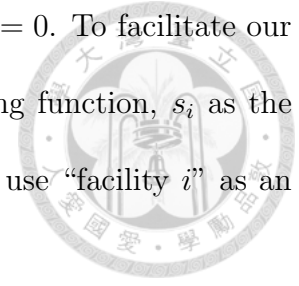
Problem Description and Formulation

3.1 Model

Suppose that there is a set of locations $I = \{1, 2, \dots, n\}$, $|I| = n$, at which service facilities may be built. Let E be the set of all undirected edges $[i, j]$ where $i \in I$, $j \in I$, and $i < j$. As $|I| = n$, we have $|E| = \frac{n(n-1)}{2}$. For each facility location i , the fixed construction cost of the facility is $h_i > 0$ and the decision variable $x_i = 1$ means there is a facility in location i and 0 otherwise. Once a facility is built at location i , it will not only increase the demands, but also affect the impact of other facilities. Let $s_i \geq 0$ be a coefficient for the facility in location i and $t_{ij} \geq 0$ be a coefficient for the facilities between location i and j , we assume the effective demand is

$$g\left(\sum_{i \in I} s_i x_i + \sum_{[i,j] \in E} t_{ij} x_i x_j\right),$$

where $g(\cdot)$ is a nondecreasing and concave function satisfying $g(0) = 0$. To facilitate our discussion, we shall refer to the function g as the demand-boosting function, s_i as the stand-alone benefit, t_{ij} as the network benefit, and at some times use “facility i ” as an abbreviation of “the facility in location i .”



The stand-alone benefit s_i measures how a facility at location i may affect the demand directly. When $s_i = 0$, building a facility at location i does not drive up the sales itself; when s_i is large, however, the demand volume will be largely determined by whether there is a facility at location i no matter there exists other facilities or not. The network benefit t_{ij} measures how building facilities i and j together may bring up the demand than there is only facility i or facility j . In some applications, e.g., the distance between i and j is one of the main factors determining t_{ij} . We assume that $s_i \geq 0$ for all $i \in I$ and $t_{ij} \geq 0$ for all $[i, j] \in E$.

The retailer’s question is to make the facility location decision so that the total profit is maximized. Her complete problem is

$$\max_{x_i \in \{0,1\}} rg \left(\sum_{i \in I} s_i x_i + \sum_{[i,j] \in E} t_{ij} x_i x_j \right) - \sum_{i \in I} h_i x_i$$

where $r > 0$ is the price for a product or one time of service. Without loss of generality, we normalize r to 1 to obtain

$$\max_{x_i \in \{0,1\}} g \left(\sum_{i \in I} s_i x_i + \sum_{[i,j] \in E} t_{ij} x_i x_j \right) - \sum_{i \in I} h_i x_i \quad (3.1)$$

In this study, we consider the problem in (3.1) as our facility location problem.

In our facility location problem, the objective function consists of two parts, the net sales revenue and the construction cost. In this region, the effective demand is $g(\sum_{i \in I} s_i x_i + \sum_{[i,j] \in E} t_{ij} x_i x_j)$, the demand brought by the stand-alone and network effects



Parameters	
I	the set of locations to build facilities at
E	the set of all undirected edges connecting a pair of locations
s_i	the stand-alone benefit of facility i
t_{ij}	the network benefit between facilities i and j
h_i	construction cost of facility i
$g(S)$	the effective demand given the sum of all effects S
Decision variables	
x_i	1 if a facility is built at location i or 0 otherwise

Table 3.1: Notations

of those open facilities. This is also the sales revenue after we normalize the price to 1. The retailer's problem in (3.1) is to maximize its total profit, which is the net sales revenue minus the construction costs $\sum_{i \in I} h_i x_i$.

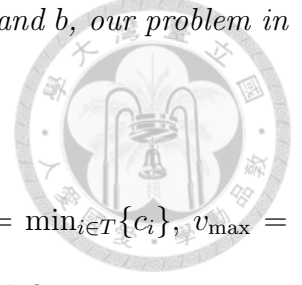
Table 3.1 lists all the notations mentioned above.

3.2 NP-hardness

We now show that our problem is weakly NP-hard by a reduction from the Knapsack problem:

Definition 1. *A knapsack instance consists of two positive numbers L' and K' and a set T of items where $|T| = n$. Item $i \in T$ has its value v_i and its size c_i . The question is whether there is a subset of items $S \subset T$ such that $\sum_{i \in S} c_i \leq L'$ and $\sum_{i \in S} v_i \geq K'$.*

Proposition 1. *If $g(\cdot)$ satisfies $\lim_{x \rightarrow \infty} \frac{g(ax)}{g(bx)} = 1$ for all positive a and b , our problem in (3.1) is NP-hard.*



Proof. For ease of exposition, we define $c_{\max} = \max_{i \in T} \{c_i\}$, $c_{\min} = \min_{i \in T} \{c_i\}$, $v_{\max} = \max_{i \in T} \{v_i\}$, and $C = \sum_{i \in I} c_i$. Given a knapsack instance, we first define

$$\rho(x) = \frac{g(Cx) - g((L' - c_{\max})x)}{g(L'x)}$$

for all $x > 0$. Since the assumption we impose on $g(\cdot)$ implies that $\rho(x)$ goes to 0 as x approaches infinity, we can find a positive number k such that $c_{\min} - \rho(k)L' > 0$. With our k and $\rho(k)$, we construct our location instance by making

$$I = T, \quad r = r_1 r_2, \quad r_1 = \frac{kL'}{g(kL')}, \quad r_2 = \frac{v_{\max}}{c_{\min} - \rho(k)L'}$$

$$s_i = kc_i \quad \forall i \in I, \quad h_i = k(r_2 c_i - v_i) \quad \forall i \in I, \quad t_{ij} = 0 \quad \forall [i, j] \in E, \quad \text{and} \quad K = kK'$$

Let $z(y)$ be the objective value of the location instance with the solution vector y . We now prove the equivalence of the two instances below.

Suppose there is a subset of items S such that $\sum_{i \in S} c_i \leq L'$ and $\sum_{i \in S} v_i \geq K'$. The former implies $\sum_{i \in S} s_i \leq kL'$, which further implies

$$g\left(\sum_{i \in S} s_i\right) \geq \frac{g(kL')}{kL'} \sum_{i \in S} s_i = \frac{1}{r_1} \sum_{i \in S} s_i = \frac{k}{r_1} \sum_{i \in S} c_i,$$

where the inequality comes from that g is concave and $g(0) = 0$. Let $y_i = 1$ if $i \in S$ and 0 otherwise. Then

$$z(y) = rg\left(\sum_{i \in S} s_i\right) - \sum_{i \in S} h_i \geq r_2 k \sum_{i \in S} c_i - k \sum_{i \in S} (r_2 c_i - v_i) = k \sum_{i \in S} v_i \geq kK' = K.$$

Therefore, y is a feasible solution of the location instance and $z(y) \geq K$.

Suppose there exists y satisfying $z(y) \geq K$, we now show if $\sum_{i \in I} c_i y_i > L'$, then for all i such that $y_i = 1$, we have $y' = y - e_i$ satisfying $z(y') \geq K$, where $e_i = (0, \dots, 0, 1, 0, \dots, 0)$, a vector with the i -th element being 1 and all others being 0. To see this, first note that since

$$\sum_{i \in I} s_i y_i \leq \sum_{i \in I} s_i = k \sum_{i \in I} c_i = kC,$$

we have $g(\sum_{i \in I} s_i y_i) \leq g(kC)$ since $g(\cdot)$ is nondecreasing. In addition, if $y_j = 1$, then

$$\sum_{i \in I} s_i y'_i = \sum_{i \in I} s_i y_i - s_j = k \sum_{i \in I} c_i y_i - k c_j > k(L' - c_j) \geq k(L' - c_{\max}),$$

which means $g(\sum_{i \in I} s_i y'_i) > g(k(L' - c_{\max}))$ since $g(\cdot)$ is nondecreasing. The two observations then implies

$$\begin{aligned} z(y') - z(y) &= r g\left(\sum_{i \in I} s_i y'_i\right) - \sum_{i \in I} h_i y'_i - r g\left(\sum_{i \in I} s_i y_i\right) + \sum_{i \in I} h_i y_i \\ &> r \left[g(k(L' - c_{\max})) - g(kC) \right] + h_j \\ &= r_2 \frac{kL'}{g(kL')} \left[g(k(L' - c_{\max})) - g(kC) \right] + k r_2 c_j - k v_j \\ &= k \left[r_2 (c_j - \rho(k)L') - v_j \right] \geq k \left[r_2 (c_{\min} - \rho(k)L') - v_j \right] \\ &= k(v_{\max} - v_j) \geq 0. \end{aligned}$$

Therefore, $z(y') > z(y) \geq K$. We may repeatedly remove selected facilities until we have a solution which satisfies the requirement of the knapsack instance. \square

Observing that the equivalent instance of our problem has no network effect, the problem with network effect is obviously more difficult.

While we have shown that our problem is weakly NP-hard in general, note that our assumption on $g(\cdot)$ in Proposition 1 is satisfied by many concave functions, such as the negative exponential function $g(z) = 1 - e^{-az}$ for all $a > 0$, the logarithm function

$g(z) = \log_a(1 + z)$ for all $a > 0$, and the kink function

$$g(z) = \begin{cases} az & \text{if } z < \frac{b}{a} \\ b & \text{otherwise} \end{cases} .$$



Whether our problem with $g(\cdot)$ violating the assumption in Proposition 1 is NP-hard remains open to us.



Chapter 4

Analysis

In this chapter, we will propose two algorithms for obtaining a feasible solution to our facility location problem defined in (3.1). For ease of exposition, let

$$z(x) = g\left(\sum_{i \in I} s_i x_i + \sum_{e \in E} t_{ij} x_i x_j\right) - \sum_{i \in I} h_i x_i$$

be the objective value associated with the solution x .

4.1 Approximation-relaxation-sorting algorithm

4.1.1 The algorithm

The first algorithm that we propose has three major steps: approximating the $g(\cdot)$ function by a kink function, solving the linear relaxation of the program, and sorting and selecting facilities according to the solution to the linear relaxation. Therefore, we call this algorithm the approximation-relaxation-sorting algorithm (ARSA).

Let the original problem defined in (3.1) be problem (P^{ori}). Given the original $g(\cdot)$

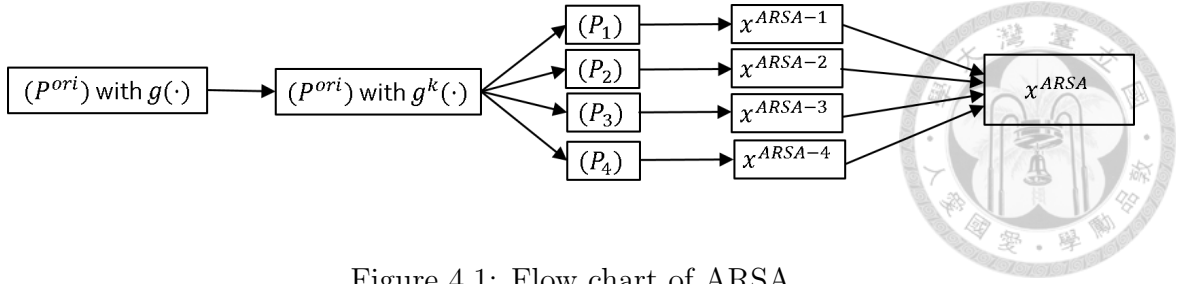


Figure 4.1: Flow chart of ARSA

function, ARSA first uniquely defines a kink function

$$g^K(x) = \begin{cases} ax & \text{if } x \leq \frac{B}{a} \\ B & \text{otherwise} \end{cases}, \quad (4.1)$$

where $a = g'(0)$, the slope of g at $x = 0$, and

$$B = g\left(\sum_{i \in I} s_i + \sum_{i \in I, i \in I, i \neq J} t_{ij}\right),$$

the maximum possible value of $g(\cdot)$ obtained by building all facilities. We then replace $g(\cdot)$ by $g^K(\cdot)$ to convert (P^{ori}) to another nonlinear program.

As $g^K(x) = \min\{ax, B\}$, the nonlinear objective function can now be linearized by introducing a new variable p and two new constraints. We also replace $x_i x_j$ by a new variable y_{ij} to eliminate products between decision variables. For any slope a , we can get an equivalent problem by setting the slope a to 1 and all stand-alone and network benefits to a times larger. Therefore, we normalize a to 1 without loss of generality.

The equivalent linear integer program is

$$\begin{aligned}
 \max_{x_i, p} \quad & p - \sum_{i \in I} h_i x_i \\
 \text{s.t.} \quad & p \leq B \\
 & p \leq \sum_{i \in I} s_i x_i + \sum_{[i, j] \in E} t_{ij} y_{ij} \\
 & y_{ij} \leq x_i \quad \forall [i, j] \in E \\
 & y_{ij} \leq x_j \quad \forall [i, j] \in E \\
 & x_i \in \{0, 1\} \quad \forall i \in I \\
 & y_{ij} \in \{0, 1\} \quad \forall [i, j] \in E.
 \end{aligned}$$



We call this integer program (P) . For this integer program, ARSA then creates n subproblems by adding the following constraint

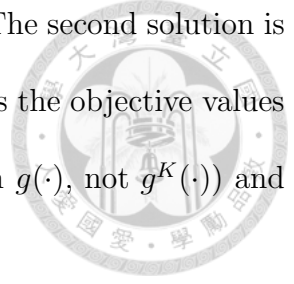
$$\sum_{i \in I} x_i = k$$

to the k th subproblem, $k = 1, 2, \dots, n$. We call the k th subproblem (P_k) . The optimal solution of (P_k) is the best set contains exact k locations. Obviously, the best among the optimal solutions to (P_k) , $k = 1, \dots, n$, is an optimal solution to (P) . Our strategy is to obtain a near-optimal solution for each (P_k) . Then the best among our near-optimal solutions for (P_k) s will also be near-optimal to (P) .

To find an integer solution for (P_k) , ARSA first relaxes the integer constraints on x_i s and y_{ij} s to obtain its linear relaxation, whose optimal solution can be found in polynomial time. Let x_i^k be the value of x_i in an optimal solution to (P^k) .

The last step is to sort candidate locations by x_i^k in the descending order. Let $x_{(1)}^k \geq x_{(2)}^k \geq \dots \geq x_{(n)}^k$, where ties are broken arbitrarily, ARSA then compares two solutions.

The first solution is simply choosing facilities (1), (2), ..., and (k). The second solution is to choose facilities (1), (2), ..., ($k - 1$), and ($k + 1$). ARSA compares the objective values achieved by the two solutions (with respect to the original function $g(\cdot)$, not $g^K(\cdot)$) and reports the better one to be a feasible integer solution to (P_k).



By repeating this for all subproblems, we obtain n solutions that are feasible to (P) as well as (P^{ori}). We finally choose the best solution among these n candidate solutions. Algorithm 1 is the pseudocode describing ARSA.

4.1.2 Worst-case performance analysis

In this section, we will prove two worst-case performance guarantees when ARSA is applied to two special cases of our facility location problem. To describe the two special cases, we state three assumptions below.

Assumption 1. $g(\cdot)$ is the kink function $g^K(\cdot)$ in (4.1) for some $a > 0$ and $B > 0$.

Note that under this assumption, it is without loss of generality to assume that there is no location whose $s_i > \frac{B}{a}$. If such a location exists, an optimal solution will either contain only this location or does not contain it. We may thus safely remove this location for a while, solve the remaining problem, and at the end check whether selecting only this location is actually the best option.

Another assumption we need for ARSA to have a worst-case performance guarantee is the following.

Assumption 2. The net stand-alone benefits $s_i - h_i$, $i \in I$, satisfy

$$\max_{i \in I} \{s_i - h_i\} > 0.$$



Algorithm 1 approximation-relaxation-sorting algorithm (ARSA)

- 1: Find the kink function $g^K(\cdot)$ that approximate $g(\cdot)$. Replace $g(\cdot)$ by $g^K(\cdot)$ to obtain the problem (P) .
 - 2: Split the problem to n subproblems, one with an additional constraint $\sum_{i \in I} x_i = k$, $k = 1, \dots, n$. Let the subproblems be (P_1) , ..., and (P_n) .
 - 3: **for** k from 1 to n **do**
 - 4: Relax the integer constraints in (P_k) .
 - 5: Solve the relaxation of (P_k) . Let x_i^k be the value of x_i in the optimal solution.
 - 6: Sort locations so that $x_{(1)}^k \geq \dots \geq x_{(n)}^k$, where ties are broken arbitrarily.
 - 7: Construct a solution \bar{x} such that $\bar{x}_{(1)} = \dots = \bar{x}_{(k)} = 1$ and 0 otherwise.
 - 8: Construct a solution \hat{x} such that $\hat{x}_{(1)} = \dots = \hat{x}_{(k-1)} = \hat{x}_{(k+1)} = 1$ and 0 otherwise.
 - 9: **if** $z(\bar{x}) > z(\hat{x})$ **then**
 - 10: Report \bar{x} as the proposed solution for (P_k) .
 - 11: **else**
 - 12: Report \hat{x} as the proposed solution for (P_k) .
 - 13: **end if**
 - 14: **end for**
 - 15: Report the best solution among the proposed solutions for the n subproblems.
-

This assumption says that the most positive net stand-alone benefit is greater than zero. In other words, there must be some facility which can bring profit itself. With this assumption, we know that the maximum benefit of building only one facility is positive.

Under Assumption 2, we define a ratio that directly affects our worst-case performance guarantee.

Definition 2. *For an instance of our facility location problem in (3.1), we define its critical ratio as*

$$r = -\frac{\min_{i \in I}\{s_i - h_i\}}{\max_{i \in I}\{s_i - h_i\}}.$$

Note that r is readily available when an instance of our facility location problem is given. Also note that according to our definition, r is negative if $s_i - h_i > 0$ for all $i \in I$ or positive if there is at least one location whose net stand-alone benefit $s_i - h_i < 0$. Finally, we have $r \geq -1$. $r = -1$ if and only if $s_i - h_i$ are identical for all $i \in I$.

Finally, we need all network benefits to be equal. One example in which the assumption may be valid is to build car/bike sharing stations where a drive/ride between any two locations are equally possible. The next assumption formalize this requirement.

Assumption 3. *There is a constant $t \geq 0$ such that the network benefits t_{ij} , $[i, j] \in E$, satisfy*

$$t_{ij} = t \quad \forall [i, j] \in E.$$

Before proving that ARSA has a worst-case performance guarantee under our three assumptions, we first prove Lemma 1. It states that, under Assumption 3, an optimal solution to the relaxation of subproblem (P_k) has either zero or two x_i^k s that are fractional.

Lemma 1. *Suppose that Assumption 3 is satisfied. An optimal solution to the relaxation of subproblem (P_k) has either zero or two fractional components. If there are two fractional components, we have*

$$\sum_{i \in I} s_i x_i + t \left(\frac{k(k-1)}{2} \right) = B.$$

Proof. First, note that Assumption 3 and the additional constraint $\sum_{i \in I} x_i = k$ together make the relaxation of subproblem (P_k) become

$$\begin{aligned} \max_{x_i, p} \quad & p - \sum_{i \in I} h_i x_i \\ \text{s.t.} \quad & p \leq \sum_{i \in I} s_i x_i + \left(\frac{k(k-1)}{2} \right) t \\ & p \leq B \\ & \sum_{i \in I} x_i = k \\ & x_i \in [0, 1] \quad \forall i \in I, \end{aligned}$$

where the variable y_{ij} is not needed anymore, and the total network benefit is exactly $\left(\frac{k(k-1)}{2} \right) t$ regardless of which k locations are selected. We then follow the idea of Caprara et al. (2000) to prove this result. For our linear program with $n + 1$ variables, at least $n + 1$ constraints are binding at an optimal extreme point solution. If at least three x_i s are fractional, we will have at most $n - 3$ constraints binding in the set of constraints $x_i \in [0, 1]$. The maximum number of binding constraints is thus n , which is not enough. If exactly two x_i s are fractional, the first three constraints must all be binding, and the right-hand-side values of the first two constraints must be identical. Because $\sum_{i \in I} x_i$ must be an integer, there is no solution with only one fractional x_i . Finally, no fractional variable is also a possible outcome. □

With Lemma 1, we are now ready to prove that ARSA is a $\frac{1}{2+r}$ -approximation algorithm for our facility location problem when our three assumptions hold.

Proposition 2. *Suppose that Assumptions 1, 2, and 3 hold. For the problem defined in (3.1), let z^* and z' be the objective values of an optimal solution and the solution reported by ARSA, respectively. We then have $\frac{z'}{z^*} \geq \frac{1}{2+r}$.*

Proof. Let z_k^* denotes the objective value of an optimal solution to subproblem (P_k) (with constraint $\sum_{i \in I} x_i = k$) and z_k^{LP} denote that to its relaxation. Note that

$$\max_{k=1, \dots, n} \{z_k^{LP}\} \geq \max_{k=1, \dots, n} \{z_k^*\} = z^*. \quad (4.2)$$

Our plan is to prove that, for each subproblem (P_k) , at least one of ARSA's n solutions will be at least one-third as good as an optimal solution to the relaxation. More precisely, let x^{LP-k} and x^{ARSA-k} be an optimal solution to the relaxation of (P_k) and the solution reported by ARSA, we will show that

$$\max_{k=1, \dots, n} \left\{ z(x^{ARSA-k}) \right\} \geq \frac{1}{2+r} z(x^{LP-k}) = \frac{1}{2+r} z_k^{LP} \quad \forall k = 1, \dots, n. \quad (4.3)$$

Combining (4.2) and (4.3), we will be able to complete the proof.

We now prove (4.3). From lemma 1, we know x^{LP-k} has either two or no fractional value. If it has no fractional value, ARSA obviously selects the optimal solution to make $x^{ARSA-k} = x^{LP-k}$ and achieves z_k^{LP} . Now suppose that there are two fractional variables, say, x_1^{LP-k} and x_2^{LP-k} . Without loss of generality, let $x_1^{LP-k} = c = 1 - x_2^{LP-k}$ for some $c \in (0, 1)$ and $s_1 \geq s_2$. Let L_0 be the set of $k-1$ locations with $x_i^{LP-k} = 1$. For the relaxation, we know

$$z_k^{LP} = \sum_{i \in L_0} (s_i - h_i) + c(s_1 - h_1) + (1 - c)(s_2 - h_2).$$

ARSA will select the $k - 1$ locations in L_0 and either location 1 or location 2 to form a solution. Let $L_1 = L_0 \cup \{1\}$ be the former solution and $L_2 = L_0 \cup \{2\}$ be the latter. If location 2 is selected, because $s_1 \geq s_2$, we have

$$\sum_{i \in L_2} s_i + T_k < \sum_{i \in I} s_i^{LP-k} x_i + T_k = B,$$

i.e., the total benefit obtained in the solution L_2 does not exceed B . Therefore, we have

$$z(x^{ARSA-k}) = \sum_{i \in L_0} (s_i - h_i) + (s_2 - h_2).$$

The difference is

$$z_k^{LP} - z(x^{ARSA-k}) = c((s_1 - h_1) - (s_2 - h_2)) \leq (s_1 - h_1) - (s_2 - h_2).$$

Obviously, $s_1 - h_1 \leq \max_{i \in I} \{x_i - h_i\}$. Moreover, according to Assumption 2, we have $-(s_2 - h_2) \leq r \max_{i \in I} \{x_i - h_i\}$. Therefore, we have $z(x^{ARSA-k}) + (1+r) \max_{i \in I} \{x_i - h_i\} \geq z_k^{LP}$. As $\max_{i \in I} \{x_i - h_i\}$ is the solution reported by ARSA for (P_1) , we have

$$z(x^{ARSA-k}) + (1+r)z(x^{ARSA-1}) \geq z_k^{LP},$$

which implies that $\max\{z(x^{ARSA-k}), z(x^{ARSA-1})\} \geq \frac{1}{2+r} z_k^{LP}$. This implies (4.3) and thus completes the proof. \square

Note that if $s_i - h_i > 0$ for all $i \in I$, we have $r < 0$ and thus the performance guarantee $\frac{1}{2+r} > \frac{1}{2}$. In this case, ARSA is guaranteed to perform pretty well. If there is at least one location such that $s_i - h_i < 0$, we have $r > 0$. If that negative $s_i - h_i$ is far below 0, our r will be large and the guarantee would be small. Finally, note that $r = -1$ if and only if all $s_i - h_i$ are identical. In this case, indeed ARSA will obtain the optimal solution (and that is why the performance guarantee is 1).

We next prove that ARSA is a $\frac{1}{2}$ -approximation algorithm for our facility location problem in another special case. We first state another assumption, which is a stronger one than Assumption 3.



Assumption 4. *The network benefits t_{ij} , $[i, j] \in E$, satisfy*

$$t_{ij} = 0 \quad \forall [i, j] \in E.$$

We now start proving ARSA has performance greater than $\max\{\frac{1}{2}, \frac{1}{2+r}\}$

under Assumptions 1 and 4. Note that now Assumption 2 is not needed.

Proposition 3. *Suppose that Assumptions 1 and 4 hold. For the problem defined in (3.1), let z^* and z' be the objective values of an optimal solution and the solution reported by ARSA, respectively. We then have $\frac{z'}{z^*} \geq \max\{\frac{1}{2}, \frac{1}{2+r}\}$.*

Proof. Due to the similarity between this proof and the previous one, we only describe the different part. Let $F_N = \{i : s_i - h_i < 0\}$ while N is the abbreviation of “negative.” Since $t = 0$, the optimal solution of the integer problem does not contain any facility in F_N .

Suppose the optimal solution contains k facilities. For the subproblem (P_k) , we know that $z_k^{LP} > z^*$. And the optimal solution x^k also contains no facility in F_N .

We now have a new bound of the difference between $z_k^{LP} - z(x^{ARSA-k})$, which is

$$\begin{aligned} z_k^{LP} - z(x^{ARSA-k}) &= c((s_1 - h_1) - (s_2 - h_2)) \\ &\leq (s_1 - h_1) + \max\{0, r\} \max_{i \in I} \{x_i - h_i\} \\ &\leq (1 + \max\{0, r\}) \max_{i \in I} \{x_i - h_i\}, \end{aligned}$$

where the first inequality comes from the fact that $s_2 - h_2 \geq 0$; otherwise, it will not be set to be a positive value when solving the relaxation of (P_k) . It then follows that

$$z(x^{ARSA-k}) + (1 + \max\{0, r\})z(x^{ARSA-1}) \geq z_k^{LP}$$

and thus $\max\{z(x^{ARSA-k}), z(x^{ARSA-1})\} \geq \max\{\frac{1}{2}, \frac{1}{2+r}\}z_k^{LP}$. This completes the proof. □

4.2 Naive greedy algorithm

We here present another algorithm, the naive greedy algorithm (NGA). In each iteration, NGA selects the facility which brings most profit at that moment. It terminates when there is no facility bringing a positive profit in that turn. Different from ARSA, We are unable to show that NGA has a worst-case performance guarantee. The pseudocode of NGA is listed in Algorithm 2.

Algorithm 2 Naive greedy algorithm

- 1: let $R = \phi$, $P(R) = rg(\sum_{j \in R} s_j + \sum_{i \in I, j \in I, i \neq j} t_{ij}x_{ij}) - \sum_{j \in R} h_j$.
 - 2: **repeat**
 - 3: choose a facility j with the highest positive $P(R \cup \{j\}) - P(R)$.
 - 4: $R \leftarrow R \cup \{j\}$
 - 5: **until** each remaining facility k satisfies $P(R \cup \{k\}) - P(R) < 0$
 - 6: **return** R
-





Chapter 5

Numerical Study

5.1 Solution performance

To understand how the algorithm performs in the problem, we compare the result given by the algorithm to the naive greedy algorithm. Different factors are adopted to observe the performances under different circumstances. The first factor is the problem scale. We consider four scenarios, each with $n = 10$, $n = 20$, $n = 50$, $n = 100$. The second factor is the stand-alone benefit distribution. The two scenarios are random distribution and deterministic. The third factor is the network benefit distribution. The three scenarios are uniform value, zero, and random. The last factor is the form of the g function with two scenarios, kink and smooth. The four factors generate 48 scenarios together, each of which has 100 instances. We adopt the following settings for each scenario:

- problem scale: scenario small: $n = 10$; scenario medium: $n = 20$; scenario large: $n = 50$; scenario extremely large: $n = 100$.

- stand-alone benefit distribution: scenario random: $s \sim U(0, 100)$; scenario deterministic: $s = \sqrt{h}$.
- network benefit distribution: scenario uniform value: $t_{ij} = t \forall [i, j] \in E, t \sim U(0, 20)$; scenario random: $t_{ij} \sim U(0, 20) \forall [i, j] \in E$; scenario zero: $t_{ij} = 0 \forall [i, j] \in E$.
- function form: scenario kink: $g(x) = \min\{x, K\}$; scenario smooth: $g(x) = K(1 - e^{-\frac{x}{K}})$ where K is a parameter we set according to the scenarios.

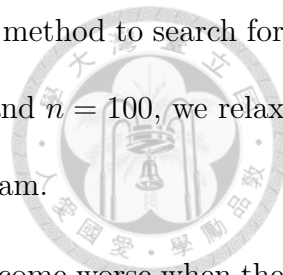
We discuss the parameter K further more. The kink function is a piece-wise linear function which has form $g(x) = \max\{x, K\}$. If K is too small, the optimal solution won't choose any facilities. So we set K at the average of the sum of the stand-alone benefit and the network benefit when choosing $0.6n$ facilities.

After generating those 48 scenarios, some scenarios are trivial to solve. For the scenarios with $s = \sqrt{h}$, $t_{ij} = t$ or $t_{ij} = 0$, and g is kink, we observe that $h_i > h_j$ iff $s_i - h_i < s_j - h_j$. Since there is no different when choosing different facility when the number of facilities is fixed, choose the facility with smaller h is always better. So we simply remove those 8 scenarios.

We solve the problems with the solver CPLEX. The analysis is run on a personal computer with Windows 7, 12G RAM, and Intel i5-4570 3.2 GHz CPU. We use Java to implement our algorithm and invoke CPLEX.

There is one thing that should also be aware that cplex cannot solve the scenarios with $n = 100$ and $n = 50$ within a tolerable time when the problem is ILP (integer linear programming problem). We linear relax the problem instead to get the upper bound of the optimal value. The scenarios with the g function is smooth are nonlinear integer

programs which cannot be solved by any solver. We use exhaustive method to search for optimal solutions instead when $n = 10$ and $n = 20$. When $n = 50$ and $n = 100$, we relax the integer constraint and use minos to solve the exponential program.




In table 5.1, we realize that the performance of our algorithm become worse when the problems become larger, but back to a high level performance when it comes to extremely large. The performance of NGA has the same pattern, too. The main reason is that when the problems become larger, their is more combination of facilities to build which makes ARSA and NGA difficult to give a better solution. On the other hand, when it comes to extremely large scenario, choosing one facility wrong does not affect the performance so much. It is also the reason that the minimum performance of ARSA goes up when the problem is larger.

Problem Scale	Average		Minimum	
	$\frac{z^{ARSA}}{z^*}$	$\frac{z^{NGA}}{z^*}$	$\frac{z^{ARSA}}{z^*}$	$\frac{z^{NGA}}{z^*}$
Small	0.9030	0.8741	0.5247	0.6106
Medium	0.8455	0.7745	0.6834	0.6709
Large	0.8253	0.6961	0.6965	0.5448
Extremely Large	0.8957	0.7977	0.7713	0.5793

Table 5.1: Numerical results of problem scale

Table 5.2 shows that ARSA performs a little worse than NGA on random scenario, while it performs much better on deterministic case. When the stand-alone benefit is deterministic, the network benefit is the only matter thing. Since NGA cannot take the network benefit into consideration, it is the reason that it performs so bad on deterministic

case.



Stand-alone benefit distribution	Average		Minimum	
	$\frac{z^{ARSA}}{z^*}$	$\frac{z^{NGA}}{z^*}$	$\frac{z^{ARSA}}{z^*}$	$\frac{z^{NGA}}{z^*}$
Random	0.8912	0.9286	0.6982	0.8419
Deterministic	0.8317	0.5711	0.6252	0.2406

Table 5.2: Numerical results of stand-alone benefit distribution

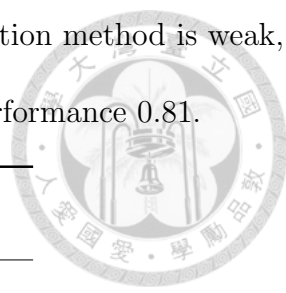
In table 5.3, we can see that NGA is better than ARSA when there is no network benefit, while the other two is not. The reason is similar to last table. NGA cannot consider the network effect while ARSA is designed for this case. The two algorithms both performs worst in the random scenario is reasonable. Randomness is always the reason to the errors.

Network benefit distribution	Average		Minimum	
	$\frac{z^{ARSA}}{z^*}$	$\frac{z^{NGA}}{z^*}$	$\frac{z^{ARSA}}{z^*}$	$\frac{z^{NGA}}{z^*}$
Uniform value	0.9055	0.7775	0.6803	0.5362
Random	0.7933	0.6758	0.5262	0.4288
Zero	0.9280	0.9401	0.8480	0.8967

Table 5.3: Numerical results of network benefit distribution

While ARSA has one more process finding the kink function to approximate the smooth function, the decrease on the performance is acceptable. Note that the kink function to approximate the smooth function is weak because we approximate it from above which generates much difference. We can easily get a better performance if we

have a stronger approximate kink function. Though the approximation method is weak, ARSA performs so well on the smooth scenario, having average performance 0.81.



Function form	Average		Minimum	
	$\frac{z^{\text{ARSA}}}{z^*}$	$\frac{z^{\text{NGA}}}{z^*}$	$\frac{z^{\text{ARSA}}}{z^*}$	$\frac{z^{\text{NGA}}}{z^*}$
Kink	0.9545	0.8379	0.8020	0.6733
Smooth	0.8093	0.7511	0.5803	0.5534

Table 5.4: Numerical results of function form

5.2 Time complexity

The time complexity needed to solve a LP is $O(n^3)$ where n is the number of the constraints. For a problem with n candidate location, ARSA need to solve n LP problems, each of which has $O(n)$ constraints. The time complexity of ARSA is then $O(n^4)$. Figure 5.1 and table 5.5 help us verify it.

We have all the factors the same as the last section but the problem scale and the function form. The problem scale is set to be 10, 20, ..., 100. We compare the two computing time of ARSA and solving the ILP with cplex. Because the smooth function form is not solvable with cplex, we remove the scenario. The average computing time among all the scenarios is then calculated. Since the computing time is too large to solve a ILP using cplex when $n > 60$, we did not collect the data of this part.

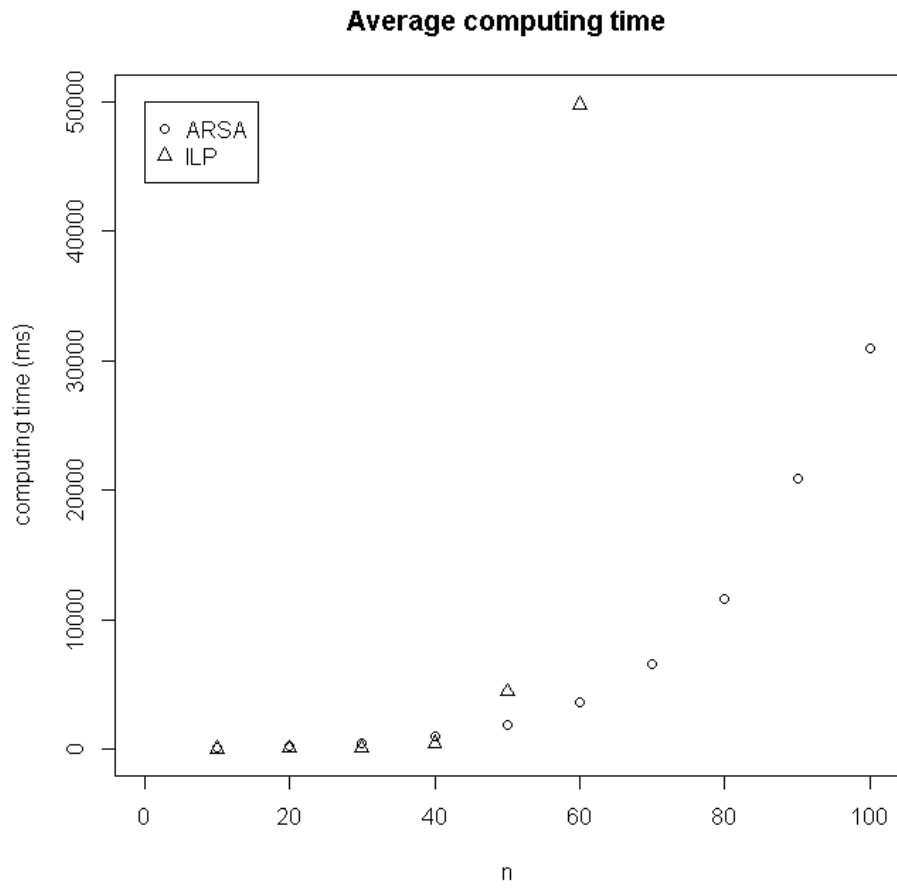


Figure 5.1: Average computing time of ARSA



n	10	20	30	40	50
ARSA (ms)	123.95	273.0917	527.5417	1031.55	2036.3917
ILP (ms)	23.8167	77.9667	112.6	459.4667	4450.5333
n	60	70	80	90	100
ARSA (ms)	3830.3417	7026.9167	12446.1417	22501.9583	29251.1
ILP	49754.4833	-	-	-	-

Table 5.5: Numerical results of function form





Chapter 6

Conclusion and Future Work

6.1 Conclusion

In vehicle renting systems, network effect has a critical impact to the value of facilities. We capture this important effect into our facility location model. After showing that the problem is weakly NP-hard, we set our target to developing an approximation algorithm. In general intuitions, a simple greedy cannot take the network effect into consideration. A special designed algorithm is then needed. Our algorithm gives performance guarantee in a special case. To test the applicability, a numerical analysis is conducted. In numerical analysis, network effect increase the difficulty of getting the better solution to algorithms. The minimum performance is low because of the difficulty. Although our algorithm cannot give performance guarantee to all the cases, we still capture the important network effect into our model. The algorithm also helps us understand more about the problem.

6.2 Future work



Though our algorithm has been showed to be effective here, there are still some directions to improve it. First, the most difficult and valuable part that we can research in future is to change the algorithm from an approximation algorithm under special conditions to all cases. We make some assumption to prove the bound of this a little complicated algorithm. There may be a bound that can be proved with a more reasonable assumption or just no assumption. Second, we have some numerical study here, but the algorithm is not applied to any practical data. The performance may not be that excellent when it comes to real. Third, our algorithm first find the piece-wise linear function to approximate the original function, which makes our algorithm not accurate enough. We wonder if there is some other way to change the non-linear function to a linear form which does not loss lots of accuracy. Fourth, due to the market cannibalization, the network effect may be negative. We do not consider this situation while it may exist in real world. Under this case, we don not know our algorithm has a performance guarantee or not. Finally, the problem is proved to be weakly NP-hard by the reduction from knapsack. A pseudo polynomial algorithm which uses dynamic programming technique may be designed to find an exact solution.



Appendix A

Results of Numerical Experiments

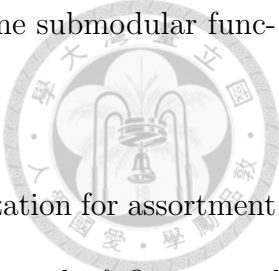
Problem scale	Stand-alone benefit	Network benefit	Function form	Average		Minimum	
				$\frac{z^{ARSA}}{z^*}$	$\frac{z^{NGA}}{z^*}$	$\frac{z^{ARSA}}{z^*}$	$\frac{z^{NGA}}{z^*}$
S	R	U	K	0.9612	0.9862	0.7905	0.8612
S	R	U	S	0.9577	0.9981	0.4403	0.8075
S	R	Z	K	0.9746	0.9955	0.8141	0.8724
S	R	Z	S	0.9792	1	0.7807	1
S	R	R	K	0.9051	0.981	0.4212	0.5645
S	R	R	S	0.8456	1	0	1
S	D	U	S	1	0.71	1	0
S	D	Z	S	1	1	1	1
S	D	R	K	0.705	0.4502	0	0
S	D	R	S	0.7019	0.62	0	0
M	R	U	K	0.9738	0.9837	0.8868	0.9099
M	R	U	S	0.966	0.9998	0.8419	0.9791
M	R	Z	K	0.9851	0.9987	0.8972	0.9671
M	R	Z	S	0.9702	0.9998	0.833	0.9876
M	R	R	K	0.9553	0.9756	0.834	0.8857
M	R	R	S	0.6126	0.9997	0	0.9799
M	D	U	S	1	0.37	1	0
M	D	Z	S	1	1	1	1
M	D	R	K	0.8959	0.1874	0.5411	0
M	D	R	S	0.0959	0.2302	0	0
L	R	U	K	0.9858	0.9754	0.9284	0.9474
L	R	U	S	0.7111	0.7229	0.4343	0.4636
L	R	Z	K	0.9869	0.994	0.9373	0.9556
L	R	Z	S	0.6235	0.6477	0.5259	0.5572
L	R	R	K	0.9875	0.9672	0.9539	0.9405
L	R	R	S	0.6953	0.7471	0.6134	0.6742
L	D	U	S	0.6916	0.261	0.2542	0
L	D	Z	S	0.9978	0.9978	0.9091	0.9091
L	D	R	K	0.9777	0.3942	0.9373	0
L	D	R	S	0.5957	0.2539	0.4713	0
E	R	U	K	0.995	0.9807	0.9633	0.9365
E	R	U	S	0.8281	0.8335	0.4837	0.5292
E	R	Z	K	0.9924	0.9964	0.9657	0.9717
E	R	Z	S	0.627	0.6523	0.5718	0.5982
E	R	R	K	0.9967	0.975	0.9894	0.9602
E	R	R	S	0.8726	0.8768	0.8498	0.8556
E	D	U	S	0.7962	0.5091	0.141	0
E	D	Z	S	0.999	0.999	0.9412	0.9412
E	D	R	K	0.9934	0.5556	0.9722	0
E	D	R	S	0.8567	0.5986	0.8352	0

Table A.1: The average and minimum performances of ARSA and NGA in all scenarios



Bibliography

- Aboolian, R., O. Berman, D. Krass. 2007. Competitive facility location model with concave demand. *European Journal of Operational Research* **181**(2) 598–619.
- Arya, V., N. Garg, R. Khandekar, A. Meyerson, K. Munagala, V. Pandit. 2004. Local search heuristics for k-median and facility location problems. *SIAM Journal on Computing* **33**(3) 544–562.
- Berman, O., D. Krass. 1998. Flow intercepting spatial interaction model: a new approach to optimal location of competitive facilities. *Location Science* **6**(1–4) 41–65.
- Berman, O., D. Krass. 2002. Locating multiple competitive facilities: spatial interaction models with variable expenditures. *Annals of Operations Research* **111**(1–4) 197–225.
- Caprara, A., H. Kellerer, U. Pferschy, D. Pisinger. 2000. Approximation algorithms for knapsack problems with cardinality constraints. *European Journal of Operational Research* **123**(2) 333–345.
- Daskin, M.S. 2013. *Network and Discrete Location: Models, Algorithms, and Applications*. Wiley, USA.

- 
- Feige, U., V.S. Mirrokni, J. Vondrk. 2011. Maximizing non-monotone submodular functions. *SIAM Journal on Computing* **40**(4) 1133–1153.
- Li, H.L., C.T. Chang, J.F. Tsai. 2002. Approximately global optimization for assortment problems using piecewise linearization techniques. *European Journal of Operational Research* **140**(3) 584–589.
- Nemhauser, G.L., L.A. Wolsey. 1978. Best algorithms for approximating the maximum of a submodular set function. *Mathematics of operations research* **3**(3) 177–188.
- 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.
- Owen, S.H., M.S. Daskin. 1998. Strategic facility location: A review. *European Journal of Operational Research* **111**(3) 423–447.
- Shen, Z.-J., C. Coullard, M. S. Daskin. 2003. A joint location-inventory model. *Transportation Science* **37**(1) 40–55.
- Williamson, D.P., D.B. Shmoys. 2011. *The Design of Approximation Algorithms*. Cambridge University Press, London, UK.
- Wu, T.H., J.N. Lin. 2003. Solving the competitive discretionary service facility location problem. *European Journal of Operational Research* **144**(2) 366–378.