An Approximation Algorithm for a Competitive Facility Location Problem with Network Effects

Ling-Chieh Kung* and Wei-Hung Liao†

Department of Information Management, National Taiwan University.

November 23, 2017

Abstract

When facilities are built to serve end consumers directly, it is natural that consumer demands are affected by the number of open facilities. Moreover, sometimes a facility becomes more attractive if other facilities around it are built. To capture these factors, in this study we construct a discrete location model for profit maximization with endogenous consumer demands and network effects. The effective demand is then a concave function of the sum of benefits of open facilities due to the diminishing marginal benefit effect. When the function is linear, we design a polynomial-time algorithm to find an optimal solution. When it is nonlinear, we show that the problem is NP-hard and develop an approximation algorithm based on demand function approximation, linear relaxation, decomposition, and sorting. It is demonstrated that the proposed algorithm has worst-case performance guarantees for some special cases of our problem. Numerical studies are conducted to demonstrate the average performance and general applicability of our algorithms.

Keywords: location, competitive facility location problem, network effect, approximation algorithm.

1 Introduction

The number and locations of facilities often affect consumers' willingness of buying a product or using a service. In fact, in many cases consumer demands may be driven up only if suf-

*Corresponding author: lckung@ntu.edu.tw

†r03725035@ntu.edu.tw

ficient facilities are built. Vehicle sharing systems provide a good illustration. As one of the most famous United States car sharing system nowadays, Zipcar is well known due to its new technologies and flexible renting plans. Nevertheless, one would sign up as a member of Zipcar only if she may easily find available cars nearby her location when she needs one. This can be achieved only if Zipcar sets up enough parking spaces in a city. A similar situation happens to public bicycle sharing system around the world. Almost all these systems can become a success only after enough rental sites have been built and enough bicycles have been supplied. As a non-transportation example, when a consumer wants to buy a personal computer or laptop, she would also evaluate how easy it is to find a warranty station when the product is broken. The distribution of warranty stations thus affects demands. Similar stories also exist for convenience stores, grocery stores, and charging or battery swapping stations for electronic vehicles.

Interestingly, opening a facility not only affects consumers directly but also changes how other facilities affect consumer demands. The first factor to consider is the interdependence among facilities. Consider the vehicle sharing business as an example. Many users of public bikes travel from subway stations to their offices, schools, and home. One would not find the service attractive even if there is one rental site just at the front door of her home. Instead, she will find it is much more attractive if there is another one at a subway station in the neighborhood. In general, while building a facility may attract consumers by itself, it may also create additional attractiveness together with other existing facilities. This is the upside of building a new facility. Nevertheless, there is also a downside: 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 altogether when one makes facility construction decisions.

In this study, we investigate a profit-maximizing service provider's facility location problem. Given a subset of locations, the service provider chooses a subset of locations to build facilities by considering two major types of effects: (1) the *stand-alone benefit* of a single facility and (2) the *network benefit* between a pair of facilities. To capture the diminishing marginal benefit property, we model a consumer's willingness to use the service as a nondecreasing concave function of the sum of all benefits. Therefore, the sum of all benefits is converted into the consumer demand (and thus the service revenue) by a nondecreasing concave *effective demand function*. By considering the total service revenue and total cost of building facilities, the service provider decides where to build facilities to maximize her profit. The problem is thus

formulated as a nonseparable nonlinear integer program, whose objective function is to maximize a nonmonotone submodular function.

We start our analysis by considering the solvability of this problem and find that the shape of the effective demand function plays an important role. When the function is linear, the problem (which is still a nonlinear integer program with the presence of products of decision variables) can be solved in polynomial time. We solve this problem by reducing it to the maximum flow problem. When the function is nonlinear, however, the problem is at least weakly NP-hard even if there is no network benefit. This is shown by a reduction from the partition problem.

As one of the most common procedures to approach NP-hard problems is to develop heuristic algorithms, we propose one based on approximating the effective demand function, relaxing integer constraints, decomposing the problem into subproblems, and sorting facilities to make the construction decisions. The algorithm is thus named the approximation-relaxation-sorting-aggregation algorithm (ARSA). We prove that ARSA exhibits worst-case performance guarantees in a few special cases of our problem. This makes ARSA an approximation algorithm for the special cases (Williamson and Shmoys, 2011). To investigate its average-case performance, we compare ARSA with an exponential-time exact algorithm and a genetic algorithm in various scenarios through numerical experiments. We demonstrate that ARSA's average-case performance is much better than its performance guarantees and the genetic algorithm. In many cases, it generates a solution that is close to an optimal solution.

Facility location problems with diminishing marginal benefit of building facilities are called competitive facility location problems (Karakitsiou, 2015). In these problems, facilities "compete" with each other in winning consumers. The competitive facility location problem has been studied from many perspectives. However, to the best of our knowledge, we are the first to develop an approximation algorithm while taking network effects into consideration. This is the major contribution of our study. Our study certainly has its limitations. In particular, we do not explicitly investigate how to quantify the form of the diminishing marginal benefit and estimate the stand-alone and network benefits. While this issue is ignored by almost all works in the literature of competitive facility location problems, we note that there is another stream of literature of developing data analytics methods to estimate consumer demands (some recent works include Hsieh et al. (2015), Tiwari and Kaushik (2014), among others). Interested readers may want to study these works. In Section 6, we briefly suggest how one may estimate parameters and apply our model and algorithm in practice. We hope one day we may connect

the two streams to make further contributions.

The remainder of this study is organized as follows. In Section 2, we discuss related works. In Section 3, we give a model formulation for our problem and conduct solvability analysis. The description of the ARSA algorithm and proofs of the worst-case performance guarantees are in Section 4. In Section 5, a numerical study is conducted to demonstrate the average-case performance and general applicability of our proposed algorithm. Concluding remarks and future works are provided in Section 6.

2 Literature review

Facility location problems have been widely studied in the past decades (Owen and Daskin, 1998; Daskin, 2013). Most traditional works are conducted by assuming that consumer demands are exogenous, i.e., not affected by the facility location decision. As the interrelationship between location decisions and demand endogeneity cannot be ignored in some situations, researchers are motivated to study competitive facility location problems (Karakitsiou, 2015).

In a static competitive facility location problem, a firm builds facilities to attract consumers (Aboolian et al., 2007a,b; Berman and Krass, 1998, 2002; Wu and Lin, 2003). Under the assumption that a consumer will only be served by one facility, facilities compete with each other. 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. Nevertheless, our model include the network effect between a pair of facilities, which is missing in most of the past works. In another stream of literature, researchers study sequential competitive facility location problems with a leader and a follower take turns to build facilities (Beresnev, 2014; Küçükayadın et al., 2011, 2012; Mel'nikov, 2014). Each firm is assumed to be a profit maximizer, and building facilities is the key to win consumers from the competitor. In this study, we investigate a static facility location problem.

Most competitive facility location problems require one to maximize a submodular objective function. Beside competitive facility location problems, submodular function maximization appears in many kinds of problems, e.g., maximum coverage, expected utility maximization with discrete choices of a risk-averse decision maker, and combinatorial auctions with submodular utilities (Ahmed and Atamtürk, 2011). As submodular function maximization is NP-hard,

many studies appear to propose polynomial-time approximation algorithms, which are heuristic algorithms that possess worst-case performance guarantees (Williamson and Shmoys, 2011). For a general submodular function maximization problem, Nemhauser and Wolsey (1978) combine exhaustion and greedy search while Nemhauser et al. (1978) adopt greedy, local search, and linear programming relaxation to develop approximation algorithms. Unfortunately, their results only apply to nondecreasing submodular functions. As our problem has a nonmonotone objective function, it cannot be solved by their algorithms with a performance guarantee.

For nonmonotone submodular function maximization, a special case is the uncapacitated facility location problem. Several researchers have developed approximation algorithms for this problem, and the best performance guarantee known so far is 0.828 (Ageev and Sviridenko, 1999; Cornuejols et al., 1977a,b). For general nonmonotone submodular function maximization, Feige et al. (2011) develop several deterministic and randomized approximation algorithms. Established approximation factors include $\frac{2}{5}$ for non-symmetric cases and $\frac{1}{2}$ for symmetric cases. For our facility location problem, we show that our algorithm achieves an above $\frac{1}{2}$ approximation factor under some explicit conditions.

To the best of our knowledge, the problem and method studied by Aboolian et al. (2007b) is the closest to our work. In their problem, a decision maker chooses several locations to build facilities to attract consumers at several markets. Each open facility gives each market a utility, which is negatively affected by the distance. The amount of realized demand at a market is a negative exponential function of the total utilities that the market obtains from all open facilities. The proportion of demand that goes to a facility is then the proportion of the total utility that comes from the facility. The decision maker's problem is to maximize the total revenue subject to a fixed budget of opening facility. To solve the problem, they first show that the objective function is smoothly concave. They then approximate the smooth concave objective function by a piecewise linear function, which is larger than the original one by at most $\alpha\%$.[1] This allows them to transform the original problem to an integer linear program, whose exact solution deviates from an optimal solution to the original problem by at most $\alpha\%$. Though this procedure can achieve any performance guarantee, it does not run in polynomial time. Therefore, they also developed a greedy algorithm having a performance guarantee $1 - \frac{1}{e}$ by applying a result in Nemhauser et al. (1978). Because the objective function in their study is nondecreasing, both these two algorithms do not apply to our problem (whose objective function

[1] A similar strategy is adopted by Li et al. (2002) for solving an assortment problem.

is nonmonotone). Nevertheless, their strategy of approximating a given objective function by a piecewise linear one turns out to be a critical step in our proposed algorithm.

3 Model and solvability

3.1 The service provider's problem

Suppose that there is a set of locations $I = \{1, 2, ..., n\}$, $|I| = n$, to build facilities. 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 facility location i, 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 $g(0) = 0$. Note that the sum of benefits collected from open facilities is input into a concave function to generate the demand volume. This setting is also adopted by Aboolian et al. (2007b) (they adopt the term "utility" rather than "benefit"). To facilitate discussion, we refer to the function $g(\cdot)$ as the *effective demand 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 at location i."

The stand-alone benefit s_i measures how a facility at location i may affect the demand by itself. When $s_i = 0$, building a facility at location i does not drive up the sales itself; when $s_i > 0$, however, the demand volume will be affected even if there exists no other facility. For the case of building warranty stations for PCs and laptops, we should have $s_i > 0$ for all i. On the contrary, for a public bike sharing system, most s_i would be close to 0.[2] The network benefit t_{ij} measures how building facilities i and j together may further bring up the demand. For a bike sharing system, if locations i and j are not too far away, we should have $t_{ij} > 0$. In this case, the distance between i and j can be one factor (but not the only factor) determining t_{ij}. For warranty stations, however, it should be reasonable to set $t_{ij} = 0$. Note that t_{ij} is measuring how attractive it is to have both facilities i and j open; it has nothing to do with

[2]It may still be positive, as some consumers may find it useful to rent and return a bike at the same location.

the competition among facilities and splitting consumer demand to multiple facilities. The diminishing marginal benefit of building facilities is modeled by the effective demand function $g(\cdot)$, not t_{ij}. This is why in our problem $t_{ij} \geq 0$ and should not be negative.

The service provider's problem is to make the facility location decision so that the total profit is maximized. Her complete problem is

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

where $\gamma > 0$ is the price of the service and $h_i > 0$ is the cost for building a facility at location i. The objective function consists of two parts, the sales revenue and the construction cost. In this market, 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 benefits of those open facilities. Multiplying the effective demand by the unit price γ results in the sales revenue. The service provider's profit is then the sales revenue minus the construction costs $\sum_{i \in I} h_i x_i$. The only constraint is the binary constraint for building facilities. Without loss of generality, we normalize γ to 1 to obtain

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

as our facility location problem in this study. For ease of exposition, we use $z(x)$ to denote the objective value associated with a solution $x = (x_1, x_2, ..., x_n)$. Moreover, for location i we define $s_i - h_i$ as its *net stand-alone benefit*. It turns out to play a critical role in the analysis of our algorithms.

Table 1 lists all the notations mentioned above.

3.2 Effective demand function and solvability

The solvability of our facility location problem in (1) critically depends on the shape of the effective demand function $g(\cdot)$. For a special case that $g(\cdot)$ is negative exponential, Ahmed and Atamtürk (2011) prove in their Theorem 1 that the problem is (weakly) NP-hard even if $t_{ij} = 0$. Below in Proposition 1 we extend their proof to any $g(\cdot)$ such that $g(w) - kw$ has a unique maximizer in $(0, \infty)$ for some $k > 0$, which is true for most concave $g(\cdot)$ that is nonlinear. Note that if $g(\cdot)$ is linear, $g(w) - kw$ will not have a unique maximizer over the open interval $(0, \infty)$. Therefore, whether our problem is NP-hard with a linear $g(\cdot)$ is not answered by Proposition 1.

Parameters	
I	the set of candidate facility locations
E	the set of all undirected edges connecting a pair of locations
n	The number of candidate facility 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
$g(B)$	the effective demand given the sum of all benefits B
Decision variables	
x_i	1 if a facility is built at location i or 0 otherwise

Table 1: Notations

Proposition 1. *Consider the problem defined in (1). Suppose that the function g satisfies the following condition: There exists some $k > 0$ such that the optimization problem*

$$\max_{w \geq 0} \; g(w) - kw$$

has a unique positive optimal solution. In this case, the problem is NP-hard.

Proof. Let the unique optimal solution be w^*. Given a partition problem of integers a_1, a_2, ..., and a_n that looks for a set S such that $\sum_{i \in S} a_i = \sum_{i \notin S} a_i$, we first normalize the integers so that $\sum_{i \in S} a_i = w^*$. Then we construct an instance of our problem as

$$z^* = \max_{x_i \in \{0,1\}} \; g\left(\sum_{i \in I} a_i x_i \right) - \sum_{i \in I} k a_i x_i,$$

i.e., we let $s_i = a_i$, $h_i = ka_i$, and $t_{ij} = 0$, where $k > 0$ is the value that makes $g(w) - kw$ having a unique maximizer in $(0, \infty)$. We now claim that such a partition exists if and only if $z^* = g(w^*) - kw^*$. If such a partition S exists, we may set $x_i = 1$ for $i \in S$ and 0 otherwise. This results in the objective value $g(\sum_{i \in S} a_i) - \sum_{i \in S} k a_i = g(w^*) - kw^*$. If our facility location problem is solved to achieve z^*, as the maximizer w^* is unique, it must be the case that we have selected a set S of facilities such that $\sum_{i \in S} a_i = w^*$. Such a set S then gives us a partition. □

The above result states that our facility location problem is NP-hard as long as the problem exhibit diminishing marginal benefit. However, it leaves the case of constant marginal benefit

unanswered. Below we assume that $g(w) = aw$ for some constant $a > 0$ and show a way to solve our problem in polynomial time.[3]

Consider the problem of maximizing

$$a\left(\sum_{i\in I} s'_i x_i + \sum_{[i,j]\in E} t'_{ij} x_i x_j\right) - \sum_{i\in I} h'_i x_i,$$

where $a > 0$ is a given constant. For this problem, we may without loss of generality normalize a to 1 by setting $s_i = as'_i$, $t_{ij} = at'_{ij}$, and $h_i = h'_i$. We then face an equivalent problem of maximizing

$$\sum_{i\in I} s_i x_i + \sum_{[i,j]\in E} t_{ij} x_i x_j - \sum_{i\in I} h_i x_i = \sum_{i\in I} (s_i - h_i) x_i + \sum_{[i,j]\in E} t_{ij} x_i x_j$$

Suppose that for facility i_1 we have $s_{i_1} - h_{i_1} > 0$, obviously $x_{i_1} = 1$ in any optimal solution (as $t_{ij} \geq 0$). Therefore, we may without loss of generality assume that $s_i - h_i < 0$ for all $i \in I$.

For our facility location problem in (1) where $g(w) = w$ and $s_i - h_i < 0$ for all i, we now demonstrate how to reduce our problem to a maximum flow problem. To do so, we define a directed graph $G = (V, A)$ as follows. The set of nodes $V = \{s\} \cup V_1 \cup V_2 \cup \{t\}$, where s is the source node, $V_1 = E$ contains the set of location pairs, $V_2 = I$ contains the set of locations, and t is the terminal node. The set of directed arcs $A = A_1 \cup A_2 \cup A_3$, where $A_1 = \{(s, [i, j]) \mid [i, j] \in V_1\}$ is the set of links from s to V_1, $A_2 = \{([i, j], i) \mid [i, j] \in V_1, i \in V_2\} \cup \{([i, j], j) \mid [i, j] \in V_1, j \in V_2\}$ contains links from V_1 to V_2, and $A_3 = \{(i, t) \mid i \in V_2$ contains links from V_2 to t. Note that each node $[i, j] \in V_1$ results in exactly two arcs in A_2 directing to the two locations i and j in V_2. Finally, the capacity is t_{ij} for arc $(s, [i, j]) \in A_1$, $h_i - s_i$ for arc (i, t) in A_3, and infinity for each arc in A_2. Note that all arc capacities are nonnegative. We then simply apply any maximum flow algorithm to obtain a minimum s-t cut (S, T), where S and T are the source and destination sets. The nodes in $V_2 \cap S$ are the optimal set of facilities to build.

Proposition 2. *Suppose that $g(w) = w$ and $s_i - h_i < 0$ for all $i \in I$. Let (S, T) be a minimum s-t cut of the graph G generated as above. An optimal solution x^* for the problem in (1) can be obtained by setting $x^*_i = 1$ if $i \in V_2 \cap S$ and 0 otherwise.*

Proof. For (S, T) to be a minimum cut, it cannot contain any link in A_2. Therefore, S cannot

[3]In practice, $g(\cdot)$ may be approximately linear when one is at the initial stage of building facilities when the diminishing marginal benefit property is still weak. This is why in this study most of the efforts are devoted to the problem of a nonlinear $g(\cdot)$. The analysis for the linear case is included for completeness.

contain $[i,j] \in V_1$ while either $i \in V_2$ or $j \in V_2$ is in T. Given this, the capacity of the cut is

$$\sum_{[i,j]\in V_1\cap T} t_{ij} + \sum_{i\in V_2\cap S}(h_i - s_i) = \sum_{[i,j]\in V_1} t_{ij} - \left[\sum_{[i,j]\in V_1\cap S} t_{ij} + \sum_{i\in V_2\cap S}(s_i - h_i)\right]$$
$$= \sum_{[i,j]\in E} t_{ij} - \left[\sum_{[i,j]\in E} t_{ij}x_i^* x_j^* + \sum_{i\in I}(s_i - h_i)x_i^*\right].$$

Therefore, a minimum cut gives an optimal solution to our facility location problem. □

In summary, our facility location problem is polynomially solvable if the marginal benefit is constant but NP-hard if it is diminishing. Below we will concentrate on the NP-hard case. In the next section, we design a polynomial-time heuristic algorithm for our problem and develop its worst-case performance guarantees in some special cases.

4 Algorithms and worst-case performance analysis

In this section, we will develop an algorithm for obtaining a feasible solution to our facility location problem defined in (1). We will introduce our proposed algorithm in Section 4.1, provide an illustrative example for it in Section 4.2, and prove its worst-case performance guarantees in some special cases in Section 4.3.

4.1 Approximation-relaxation-sorting-aggregation algorithm (ARSA)

Our algorithm has four major steps. First, regardless the functional form of the $g(\cdot)$ function, we *approximate* it by a kink function from above. The resulting problem can be transformed to an equivalent linear integer program. Second, we decompose the linear integer program into n subproblems (n is the number of candidate locations), *relax* the integer constraint, and solve the linear relaxation of each subproblems. Third, we construct a feasible solution to a subproblem through a simple *sorting* procedure. We finally *aggregate* all solutions from all subproblems to generate our solution to the original problem. Therefore, we call this algorithm the approximation-relaxation-sorting-aggregation algorithm (ARSA). Below we explain each step in details.

Approximation. Let the original problem defined in (1) be problem (P^{ori}). Given the

original $g(\cdot)$ function, ARSA first uniquely defines a kink function

$$g^K(w) = \begin{cases} aw & \text{if } w \leq \frac{B}{a} \\ B & \text{otherwise} \end{cases}, \tag{2}$$

where $a = g'(0)$ is the slope of $g(\cdot)$ at $x = 0$ and $B = g(\sum_{i \in I} s_i + \sum_{[i,j] \in E} t_{ij})$ is the maximum possible value of $g(\cdot)$ obtained by building all facilities. Note that an equivalent expression of $g^K(w)$ is $g^K(w) = \min\{aw, B\}$. It is clear that $g(x) \leq g^K(x)$ for all $x \in [0, B]$, i.e., $g^K(\cdot)$ approximates $g(\cdot)$ from the above.[4] We then replace $g(\cdot)$ by $g^K(\cdot)$ to convert (P^{ori}) to another nonlinear program.

Figure 1 provides a simple example of the approximation. Suppose that $g(w) = 4(1-e^{-w/4})$ (plotted in Figure 1 as the solid curve). Suppose also that the maximum benefit that one may collect is 13, then the maximum possible revenue is $B = g(13) \approx 3.845$. We then have $a = g'(0) = 1$ and $g^K(w) = \min\{w, g(13)\}$. In Figure 1, $g^K(w)$ is plotted as the dashed curve. We can see that it is indeed an approximation of $g(w)$ from the above.

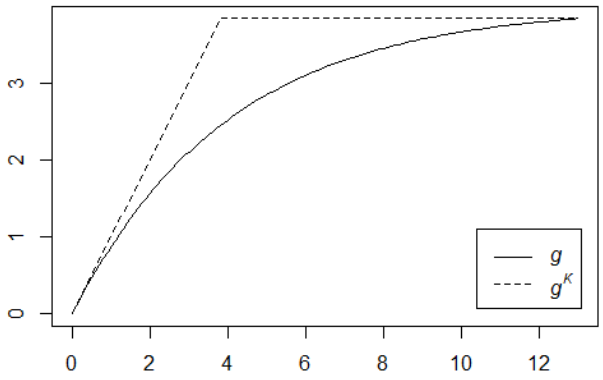

Figure 1: Approximating $g(w) = 4(1 - e^{-w/4})$ by $g^K(w) = \min\{w, g(13)\}$

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 of decision variables. For any slope a, we can get an equivalent problem by setting the slope a to 1 and replace s_i and t_{ij} by as_i and at_{ij}, respectively. Therefore, we

[4] Aboolian et al. (2007b) adopt the same from-the-above approximation to establish their α-approximate algorithm. Unfortunately, while their approach works for their monotone objective function (which includes revenues only), it cannot be applied to our nonmonotone one (which includes revenues and costs).

normalize a to 1 without loss of generality. The equivalent linear integer program is

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

We call this linear integer program (P).

Relaxation. For the integer program (P), ARSA then creates n subproblems by adding a constraint

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

to the kth subproblem, $k = 1, 2, ..., n$. We call the kth subproblem (P_k). The optimal solution of (P_k) builds exact k facilities. Obviously, the best among the optimal solutions to (P_k), $k = 1, ..., n$, is an optimal solution to (P). Our strategy is to obtain a near-optimal solution for each (P_k) and show that the best among our near-optimal solutions for (P_k)s is indeed be near-optimal to (P). To generate an integer solution for (P_k), ARSA first relaxes the integer constraints 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 the linear relaxation of (P_k).

Sorting. For each subproblem (P_k), ARSA now sorts candidate locations by x_i^k in the descending order. Let $x_{i_1}^k \ge x_{i_2}^k \ge \cdots \ge x_{i_n}^k$, where ties are broken arbitrarily, ARSA then compares two solutions $\bar{x}$ and $\hat{x}$, constructed in the following way. The first solution $\bar{x}$ is constructed by simply choosing facilities i_1, i_2, ..., and i_k. More precisely, we have $\bar{x}_{i_1} = \cdots = \bar{x}_{i_k} = 1$ and 0 otherwise. The second solution $\hat{x}$ is constructed by choosing facilities i_1, i_2, ..., i_{k-1}, and i_{k+1}. By doing so, we have $\hat{x}_{i_1} = \cdots = \hat{x}_{i_{k-1}} = \hat{x}_{i_{k+1}} = 1$ and 0 otherwise. 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 $x^{\text{ARSA-}k}$ to (P_k). By repeating this for all subproblems, we obtain n candidate solutions that are feasible to (P) as well as (P^{ori}).

Aggregation. The last step is simple: Choose the best solution among these n candidates.

Figure 2 sketches the flow of ARSA (when $n = 4$). Algorithm 1 is the pseudocode describing ARSA.

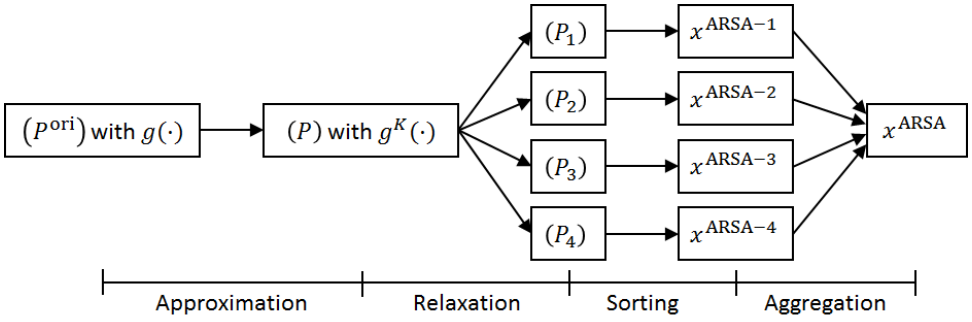

Figure 2: Basic flow of ARSA

4.2 An illustrative example

To better explain ARSA, in this section we provide a simple example for the illustration purpose. Suppose that a decision maker is given $n = 3$ locations to build facilities, as shown in Figure 3. The three locations are labeled as 1, 2, and 3. For the three locations, the stand-alone benefits are $s_1 = s_2 = s_3 = 3$, construction costs are $h_1 = h_3 = 1$ and $h_2 = 2$, and network benefits are $t_{12} = t_{23} = 2$ and $t_{13} = 5$. We also have $g(w) = 4(1 - e^{-w/4})$, the one depicted in Figure 1, to convert benefit w into revenue $g(w)$. The optimal solution is to build facilities 1 and 3.

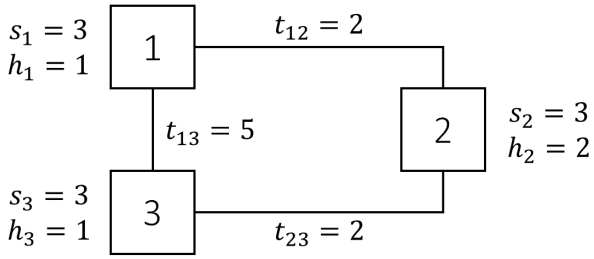

Figure 3: An illustrative example

To solve this instance by ARSA, we do the four steps as follows.

- **Step 1: Approximation.** As we have done in Figure 1, the kink function that we should adopt is $g^K(w) = \min\{w, 3.845\}$ (cf. Figure 1). By replacing $g(\cdot)$ by $g^K(\cdot)$, the original problem (P^{ori})

$$\max_{x_i \in \{0,1\}} \; 4 - e^{-(3x_1+3x_2+3x_3+2x_1x_2+2x_2x_3+5x_1x_3)/4} - (x_1 + 2x_2 + x_3)$$

Algorithm 1 approximation-relaxation-sorting algorithm (ARSA)

Find the kink function $g^K(\cdot)$ that approximate $g(\cdot)$. Replace $g(\cdot)$ by $g^K(\cdot)$ to obtain the problem (P).
Split the problem to n subproblems, one with an additional constraint $\sum_{i\in I} x_i = k$, $k = 1, ..., n$. Let the subproblems be (P_1), ..., and (P_n).
for k from 1 to n **do**
    Relax the integer constraints in (P_k).
    Solve the relaxation of (P_k). Let x_i^k be the value of x_i in the optimal solution.
    Sort locations so that $x_{i_1}^k \geq \cdots \geq x_{i_n}^k$, where ties are broken arbitrarily.
    Construct a solution $\bar{x}$ such that $\bar{x}_{i_1} = \cdots = \bar{x}_{i_k} = 1$ and 0 otherwise.
    Construct a solution $\hat{x}$ such that $\hat{x}_{i_1} = \cdots = \hat{x}_{i_{k-1}} = \hat{x}_{i_{k+1}} = 1$ and 0 otherwise.
    if $z(\bar{x}) > z(\hat{x})$ **then**
        Report $\bar{x}$ as the proposed solution $x^{\text{ARSA-}k}$ for (P_k).
    else
        Report $\hat{x}$ as the proposed solution $x^{\text{ARSA-}k}$ for (P_k).
    end if
end for
Report the best solution among $x^{\text{ARSA-1}}$, $x^{\text{ARSA-2}}$, ..., and $x^{\text{ARSA-}n}$.

can then be transformed to (P)

$$\begin{aligned} \max \quad & p - (x_1 + 2x_2 + x_3) \\ \text{s.t.} \quad & p \leq 3.845, p \leq 3x_1 + 3x_2 + 3x_3 + 2y_{12} + 2y_{23} + 5y_{13} \\ & y_{12} \leq x_1, y_{12} \leq x_2, y_{23} \leq x_2, y_{23} \leq x_3, y_{13} \leq x_1, y_{13} \leq x_3 \\ & x_1 \in \{0,1\}, x_2 \in \{0,1\}, x_3 \in \{0,1\}, y_{12} \in \{0,1\}, y_{23} \in \{0,1\}, y_{13} \in \{0,1\}. \end{aligned}$$

Note that (P) becomes a linear integer program.

- **Step 2: Relaxation.** The second step is the relaxation step. We create three subproblems (P^1), (P^2), and (P^3), where (P^k) is created by adding the constraint $x_1 + x_2 + x_3 = k$. We then relax the binary constraints on x_is and y_{ij}s to obtain the linear relaxation of (P^k).

More precisely, the relaxation of (P^k) is

$$\begin{aligned}
\max \quad & p - (x_1 + 2x_2 + x_3) \\
\text{s.t.} \quad & x_1 + x_2 + x_3 = k \\
& p \le 3.845, p \le 3x_1 + 3x_2 + 3x_3 + 2y_{12} + 2y_{23} + 5y_{13} \\
& y_{12} \le x_1, y_{12} \le x_2, y_{23} \le x_2, y_{23} \le x_3, y_{13} \le x_1, y_{13} \le x_3 \\
& x_1 \in [0,1], x_2 \in [0,1], x_3 \in [0,1], y_{12} \in [0,1], y_{23} \in [0,1], y_{13} \in [0,1].
\end{aligned}$$

Note that each of (P^k) is now just a linear program.

- **Step 3: Sorting.** The third step is the sorting step. For each subproblem (P^k), we solve its linear relaxation by any linear programming algorithm. Then we sort locations based on the values of x_i in the obtained optimal solution and select either the first k locations or the first $k-1$ locations and the $(k+1)$th location, depending on which solution gives us a higher profit (based on $g(\cdot)$, not $g^K(\cdot)$).

  - For (P^1), an optimal solution to its relaxation is $x^1 = (0.1207, 0, 0.8793)$, and the sorting results in the order 3, 1, and 2. Two candidate solutions to (P^1) are then $\bar{x} = (0,0,1)$ and $\hat{x} = (1,0,0)$. As $z(\bar{x}) = z(\hat{x}) = 1.1105$, we break tie arbitrarily and report $x^{\text{ARSA-1}} = (0,0,1)$ as the solution to (P^1).
  - For (P^2), an optimal solution to its relaxation is $x^2 = (1,0,1)$, and the sorting results in the order 1, 3, and 2 (ties are broken arbitrarily). Two candidate solutions to (P^2) are then $\bar{x} = (1,0,1)$ and $\hat{x} = (1,1,0)$. As $z(\bar{x}) = 1.7443$ and $z(\hat{x}) = 0.4587$, we report $x^{\text{ARSA-2}} = (1,0,1)$ as the solution to (P^2).
  - For (P^3), an optimal solution to its relaxation is obviously $x^3 = (1,1,1)$, the only feasible solution. We therefore report $x^{\text{ARSA-3}} = (1,1,1)$ as the solution to (P^3), even though its associated profit $z(x^{\text{ARSA-3}}) = -0.0444$ is negative.

- **Step 4: Aggregation.** We now compare the three candidate solutions $(0,0,1)$, $(1,0,1)$, and $(1,1,1)$. As $(1,0,1)$ results in the highest profit ($1.7443 > 1.1105 > -0.0444$), ARSA reports $(1,0,1)$ as the proposed solution to the original problem (P^{ori}). For this instance, ARSA finds an optimal solution.

4.3 Worst-case performance analysis of ARSA

Obviously, ARSA cannot always find an optimal solution. Therefore, it is worthwhile to investigate how badly ARSA may perform. While we cannot analytically prove a worst-case performance guarantee for ARSA in general, in this section we will present two performance guarantees when ARSA is applied to two special cases of our facility location problem. The average performance of ARSA in the general case will be examined in Section 5.

To describe the first special case, we state three assumptions below.

Assumption 1. *$g(\cdot)$ is the kink function $g^K(\cdot)$ in (2) for $a = 1$ and some $B > 0$. Moreover, $s_i < B$ for all $i \in I$.*

This assumption restricts our attention to effective demand functions that are kink functions. In this case, the approximation step of ARSA can be omitted, and there is no precision loss due to the approximation. Moreover, as we mentioned above, a can be normalized to 1 without loss of generality. Note that under this assumption, it is without loss of generality to assume that there is no location whose $s_i > B$. 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.

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

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

This assumption states that there is at least one facility which can bring profit itself. When this is true, a profitable option is to build only one facility. Under Assumption 2, we define a ratio that directly affects our worst-case performance guarantee.

Definition 1. *For a given instance of our facility location problem in (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 and Assumption 2, 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 $s_i - h_i < 0$. It is clear that $r \geq -1$, where $r = -1$ 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 all drives/rides between any two locations are equally possible. The next assumption formalizes 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 follow the idea of Caprara et al. (2000) to 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^ks that are fractional.

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

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

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

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

where the total network benefit is exactly $\frac{k(k-1)}{2} t$ regardless of which k locations are selected; the variable y_{ij} is therefore not needed anymore. For the linear relaxation of (P_k), the $n+1$ variables require that at least $n+1$ constraints are binding at an optimal extreme point solution. If at least three x_is 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_is 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. □

Note that the relaxation of (P_k) is obtained by first adding the constraint $\sum_{i=1}^{n} x_i = k$ and then relaxing the integer constraints, not the opposite. If we reverse the order, we will be unable to obtain the result in Lemma 1.

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

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

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

$$\max_{k=1,...,n} \{z_k^{LP}\} \geq \max_{k=1,...,n} \{z_k^*\} = z^*. \tag{3}$$

Let x^{LP-k} and x^{ARSA-k} be an optimal solution to the relaxation of (P_k) and the solution reported by ARSA, our plan is to show that

$$\begin{aligned} z' = \max_{k=1,...,n} \{z(x^{ARSA-k})\} &\geq \max\left\{z(x^{ARSA-k}), z(x^{ARSA-1})\right\} \\ &\geq \frac{1}{2+r} z(x^{LP-k}) = \frac{1}{2+r} z_k^{LP} \end{aligned} \tag{4}$$

for each $k = 1, 2, ..., n$, which implies that $z' \geq \frac{1}{2+r} \max_{k=1,...,n}\{z_k^{LP}\}$. Combining this and (3) then completes the proof.

We now prove the second inequality in (4) for a given k. From Lemma 1, we know x^{LP-k} has either two or no fractional value. If it has no fractional value, ARSA obviously selects an optimal solution to make $z(x^{ARSA-k}) = z(x^{LP-k})$. Now suppose that there are two fractional variables. Without loss of generality, let the two variables be x_1^{LP-k} and x_2^{LP-k} such that $s_1 \geq s_2$ and $x_1^{LP-k} = c = 1 - x_2^{LP-k}$ for some $c \in (0, 1)$. Let L_0 be the set of $k-1$ locations with $x_i^{LP-k} = 1$. For the linear relaxation, from Lemma 1 we know

$$\begin{aligned} z_k^{LP} = z(x^{LP-k}) &= \sum_{i \in I} s_i x_i^{LP-k} + \frac{k(k-1)}{2} t - \sum_{i \in I} h_i x_i^{LP-k} \\ &= \sum_{i \in L_0} (s_i - h_i) + c(s_1 - h_1) + (1-c)(s_2 - h_2) + T_k, \end{aligned} \tag{5}$$

where the constant $T_k = \frac{k(k-1)}{2} t$ is the total network benefit. 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

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

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

$$z(x^{ARSA-k}) \geq \sum_{i \in L_0} (s_i - h_i) + (s_2 - h_2) + T_k, \tag{6}$$

where the right-hand-side value is the objective value resulted from L_2, which is just one of the two candidates of x^{ARSA-k}. Combining (5) and (6), we then have the difference

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

Obviously, $s_1 - h_1 \leq \max_{i \in I}\{s_i - h_i\}$. Moreover, we have $-(s_2 - h_2) \leq r \max_{i \in I}\{s_i - h_i\}$ according to Assumption 2. Therefore, we have $z(x^{ARSA-k}) + (1 + r) \max_{i \in I}\{s_i - h_i\} \geq z_k^{LP}$. As $\max_{i \in I}\{s_i - h_i\}$ 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 $\max\{z(x^{ARSA-k}), z(x^{ARSA-1})\} \geq \frac{1}{2+r} z(x_k^{LP})$. □

Note that if there is at least one location such that $s_i - h_i < 0$, we have $r > 0$ and thus the performance guarantee $\frac{1}{2+r} < \frac{1}{2}$. On the contrary, if $s_i - h_i > 0$ for all $i \in I$, we have $r < 0$, which then implies that the performance guarantee is above one half. Roughly speaking, ARSA performs better (in the worst case) when the variability of net stand-alone benefit $s_i - h_i$ decreases. Finally, note that if all $s_i - h_i$ are identical, one may easily evaluate the profit of opening 1, 2, ..., and n facilities and find an optimal solution in polynomial time. When $s_i - h_i$ are all the same, we have $r = -1$, which implies that the performance guarantee is $\frac{1}{2+r} = 1$. In other words, ARSA does not fail to find an optimal solution for such a trivial case.

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

Assumption 4. *The network benefits $t_{ij} = 0$ for all $[i, j] \in E$.*

We now prove that ARSA has the performance guarantee $\max\{\frac{1}{2}, \frac{1}{2+r}\}$ under Assumptions 1 and 4. In other words, the performance guarantee is at least $\frac{1}{2}$ but may be better if $r < 0$ (i.e., $\min_{i \in I}\{s_i - h_i\} > 0$). Note that now Assumption 2 is not needed.

Proposition 4. *Suppose that Assumptions 1 and 4 hold. For the problem defined in (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. Because $t = 0$, the optimal solution to the linear relaxation does not contain any facility satisfying $s_i - h_i < 0$. This immediately implies

$$\begin{aligned} z(x_k^{LP}) - z(x^{ARSA-k}) &\leq c\Big[(s_1 - h_1) - (s_2 - h_2)\Big] \\ &< (s_1 - h_1) + \max\{0, r\} \max_{i \in I}\{s_i - h_i\} \\ &\leq (1 + \max\{0, r\}) \max_{i \in I}\{s_i - h_i\}, \end{aligned}$$

where the second inequality comes from the fact that $s_2 - h_2 \geq 0$. It then follows that

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

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

It is indeed unfortunate that the performance guarantees do not apply to our problem in general. Nevertheless, even if we only consider our algorithm as a heuristic algorithm for the general problem, we are still able to demonstrate its good average-case performance in Section 5 through numerical experiments. This shows the general applicability of our algorithm. Our analytical results in Section 4.3 and numerical results in Section 5 thus complement each other.

5 Numerical study

To understand how ARSA performs, we compare it with two other algorithms, an exact algorithm (either branch and bound or complete enumeration) and the genetic algorithm. To test ARSA's performance under different circumstances, the numerical study is done by incorporating four factors. Below we will describe the experiment setting in Section 5.1, explain the exact and genetic algorithms in Section 5.2, and demonstrate the results of the numerical experiments in Section 5.3. A discussion about computation time is in Section 5.4.

5.1 Experiment setting

The first factor is the instance scale, measured by the number of candidate locations n, with three levels 20, 50, and 100. These three levels are labeled as small, medium, and large.

The second factor is the net stand-alone benefit $s_i - h_i$. We first let $h_i \sim \mathrm{U}(1, 40)$, i.e., h_i follows the uniform distribution between 1 and 40. We then consider two relationships between s_i and h_i: either $s_i = \sqrt{h_i}$ or $s_i \sim \mathrm{U}(1, 40)$. The deterministic relationship $s_i = \sqrt{h_i}$ is to generate cases where all locations have negative net stand-alone benefit.[5] The alternative setting $s_i \sim \mathrm{U}(1, 40)$ is to generate cases that these two parameters are unrelated. The net stand-alone benefits may thus be of either sign. These two levels are labeled as negative and random.

The third factor is the distribution of network benefits. We consider three settings: $t_{ij} = 0$, $t_{ij} = t$, where $t \sim \mathrm{U}(0, 24)$, and $t_{ij} \sim \mathrm{U}(0, 24)$. These three levels, which are labeled as zero, identical, and random, are adopted to test the situations when Assumption 4 holds, Assumption 3 holds, or neither holds, respectively. The upper bound of t_{ij} when t_{ij} is not restricted to 0 is chosen to be 24 intentionally. To see the reason, note that the expected value of s_i is

$$\int_1^{40} \sqrt{x}\left(\frac{1}{39}\right) dx \approx 4.307$$

if $s_i = \sqrt{h_i}$ or

$$\int_1^{40} x\left(\frac{1}{39}\right) dx = 20.5$$

if $s_i \sim \mathrm{U}(1, 40)$. To generate both situations where the stand-alone benefits dominate network benefits and the opposite, we should set the expected value of the network benefits to be around the average of 4.307 and 20.5. Therefore, we set the upper bound of t_{ij} to be 24.

Finally, the last factor is the shape of the $g(\cdot)$ function. The first shape we consider is a kink function, i.e., $g(w) = \min\{w, K\}$. The second shape we consider is a smooth function, i.e., $g(w) = K(1 - e^{-w/K})$. These two levels are labeled as kink and smooth. In either case, the parameter K is set so that it is optimal to build around 60% of facilities. Let μ be the expected value of s_i (which is 20.5 if $s_i \sim \mathrm{U}(1, 40)$ or 4.307 if $s_i = \sqrt{h_i}$), we set

$$K = \begin{cases} 0.6n\mu & \text{if } t_{ij} = 0 \\ 0.6n\mu + \frac{(0.6n)(0.6n-1)}{2} t & \text{if } t_{ij} = t \\ 0.6n\mu + 12\left(\frac{(0.6n)(0.6n-1)}{2}\right) & \text{if } t_{ij} \sim \mathrm{U}(0, 24) \end{cases}.$$

The four factors together generate 36 scenarios. However, for the scenarios with $s_i = \sqrt{h_i}$ and zero or identical t_{ij}, we will have $h_i > h_j$ if and only if $s_i - h_i < s_j - h_j$. In this case, it

[5] When $s_i = \sqrt{h_i}$, to make $s_i - h_i = \sqrt{h_i} - h_i < 0$ for all i, we need $h_i > 1$. This is why we set the lower bound of the two uniform distributions to be 1 rather than 0.

is always optimal to select facilities in the ascending order of h_i until building one more facility decreases the objective value. Therefore, we remove the corresponding twelve scenarios and discuss the remaining 24 ones below.

For each scenario, we generate 100 instances to test ARSA's performance. Both the average performance across the 100 instances and the worst performance out of the 100 instances are recorded.

5.2 An exact algorithm and the genetic algorithm

To test the performance of ARSA, the first target for comparison is an exact integer solution (or its upper bound). Consider the situation when $n = 20$ first. For instances with a kink $g(\cdot)$, we linearize the piecewise-linear objective function and then apply CPLEX to find an optimal solution for the linear integer program. For those with a negative exponential $g(\cdot)$, we apply complete enumeration. Unfortunately, CPLEX and complete enumeration both fail to find an optimal solution when $n = 50$ and 100. In this case, we *relax the integer constraints* to generate a linear program if $g(\cdot)$ is kink or a convex program if $g(\cdot)$ is negative exponential. We then use CPLEX or MINOS to solve the linear or convex program respectively to obtain a deterministic *upper bound* of an optimal integer solution's objective value.

We also compare ARSA with the genetic algorithm (GA), which is a meta-heuristic algorithm that is widely applied for solving NP-hard combinatorial problems (Hillier and Lieberman, 2014). We encode a solution to our facility location problem as a chromosome with n genes, one for each location. Gene i is 1 if a facility is built at location i and 0 otherwise. We adopt the objective value of a solution as the fitness value of a chromosome.

To start GA, we first generate 500 chromosomes to form the initial population. To randomly generate a chromosome, we first randomly select a probability p within 0.1, 0.2, ..., and 0.9. Then each gene is set to 1 with probability p. If a generated chromosome's fitness value is negative, we redo the generation process until we obtain one with a positive fitness value. By doing so, all chromosomes in the population are guaranteed to have positive fitness values throughout the solution process.

With the initial population, we start to iterate. In each iteration, we select the two chromosomes with the highest fitness values as parents. Then a number $k \in \{1, 2, ..., n-1\}$ is randomly generated to perform crossover. We generate two children, where child 1's first k genes are from

parent 1, child 1's last $n-k$ genes are from parent 2, child 2's first k genes are from parent 2, and child 2's last $n-k$ genes are from parent 1. Then each gene of each child is mutated to its opposite value with a mutation probability 0.02. Among the two parents and two children (after mutation), we select the best two to replace the two parents in the population. To conduct a fair comparison between ARSA and GA, we equalize the running times of GA and ARSA by having GA run 10000, 40000, and 100000 iterations for $n = 20$, 50, and 100.

The experiments are conducted on a personal computer with Windows 7, 12G RAM, and Intel i5-4570 3.2 GHz CPU. We use the Java programming language to implement complete enumeration, ARSA, and GA. The same language is used to invoke CPLEX and MINOS.

5.3 ARSA's performance

Table 2 lists the complete result of our numerical experiments. The first four columns label the levels of the four factors. For instance scale, S, M, and L stand for small ($n = 20$), medium ($n = 50$), and large ($n = 100$); for stand-alone benefit, R and N stand for random ($s_i \sim \mathrm{U}(1, 40)$) and negative ($s_i = \sqrt{h_i}$); for network benefit, Z, I, and R stand for zero ($t_{ij} = 0$), identical ($t_{ij} = t$, $t \sim \mathrm{U}(0, 20)$), and random ($t_{ij} \sim \mathrm{U}(0, 20)$); finally, for the effective demand function $g(\cdot)$, K and S stand for kink ($g(w) = \min\{w, K\}$) and smooth ($g(w) = K(1 - e^{-w/K})$).

For each of the 24 scenarios, we run ARSA and GA on the 100 cases, and for each algorithm we record the performance ratios z^{ARSA}/z^* and z^{GA}/z^*, where z^{ARSA} is the objective values of the solutions obtained by ARSA and z^{GA} is that obtained by GA. Regarding z^*, it is the objective value of an optimal solution when $n = 20$ or an *upper bound* of that obtained by relaxing the integer constraints when $n = 50$ or 100. Each of the fifth and sixth columns then records the average of the 100 ratios for ARSA and GA, and each of the seventh and eighth columns records the minimum of the ratios. From Table 2, as the numbers in the fifth column are mostly close to 1, we conclude that ARSA typically obtains a near-optimal solution. Moreover, by comparing the fifth and sixth columns or the seventh and eighth columns, it is suggested that ARSA in general performs better than GA, both in the average and worst cases.

To understand how ARSA performs under different scenarios, we conduct further summaries with respect to the four factors. The summaries are provided in Tables 3 to 6. Because the main reason of conducting the numerical study is to investigate the average-case performance of ARSA, below we will focus on the average of performance ratios. The worst-case performances are provided for completeness.

Instance scale	Net stand-alone benefit	Network benefit	Effective demand	Average		Minimum	
				z^{ARSA}/z^*	z^{GA}/z^*	z^{ARSA}/z^*	z^{GA}/z^*
S	R	Z	K	0.983	0.758	0.891	0.620
S	R	Z	S	0.971	0.817	0.807	0.664
S	R	I	K	0.986	0.920	0.926	0.636
S	R	I	S	0.980	0.925	0.877	0.618
S	R	R	K	0.986	0.940	0.938	0.893
S	R	R	S	0.922	0.970	0.832	0.936
S	N	R	K	0.981	0.922	0.921	0.868
S	N	R	S	0.857	0.947	0.714	0.868
M	R	Z	K	0.991	0.504	0.952	0.373
M	R	Z	S	0.623	0.333	0.533	0.224
M	R	I	K	0.989	0.912	0.939	0.491
M	R	I	S	0.836	0.811	0.452	0.264
M	R	R	K	0.994	0.959	0.982	0.948
M	R	R	S	0.897	0.895	0.876	0.877
M	N	R	K	0.990	0.952	0.967	0.933
M	N	R	S	0.889	0.889	0.865	0.867
L	R	Z	K	0.993	0.349	0.961	0.258
L	R	Z	S	0.621	0.220	0.556	0.149
L	R	I	K	0.996	0.944	0.953	0.416
L	R	I	S	0.877	0.846	0.462	0.206
L	R	R	K	0.998	0.977	0.992	0.973
L	R	R	S	0.926	0.915	0.923	0.906
L	N	R	K	0.997	0.975	0.989	0.969
L	N	R	S	0.930	0.918	0.924	0.907

Table 2: The average and minimum performances of ARSA and GA in all scenarios

First, we examine the impact of instance scale. By looking at Table 3, we realize that the good performance of ARSA is not prone to the instance scale. Note that while the performance ratios are calculated using an optimal solution when $n = 20$, they are calculated using an upper bound of an optimal solution when $n = 50$ or 100. This explains the drop in performance ratios from the small scale to the medium scale.

Instance Scale	Average		Minimum	
	z^{ARSA}/z^*	z^{GA}/z^*	z^{ARSA}/z^*	z^{GA}/z^*
Small	0.958	0.900	0.714	0.618
Medium	0.901	0.782	0.452	0.224
Large	0.917	0.768	0.462	0.149

Table 3: Numerical results of problem scale

We next consider the impact of the relationship between the stand-alone benefit and construction cost. Table 4 shows that ARSA performs better when the relationship is deterministic rather than random. While this seems to be intuitive, we would like to emphasize that the deterministic setting $s_i = \sqrt{h_i}$ we set makes $s_i - h_i < 0$ for all locations. As the net stand-alone benefits are all negative, a consideration on network benefit is critical when searching for an optimal solution. It is important to see that ARSA, which does take network benefit into account, performs well in this case. When s_i and h_i becomes unrelated, naturally ARSA's performance becomes worse. Nevertheless, while the performance of GA drops significantly, that of ARSA only drops slightly. We conclude that ARSA performs much better than GA under the random relationship.

Net stand-alone benefit	Average		Minimum	
	z^{ARSA}/z^*	z^{GA}/z^*	z^{ARSA}/z^*	z^{GA}/z^*
Random	0.921	0.778	0.452	0.149
Negative	0.941	0.934	0.714	0.867

Table 4: Impact of net stand-alone benefit

The impact of network benefit is illustrated in Table 5. Again, we see that ARSA outperforms GA for all the three environments. It is somewhat surprising that ARSA performs the worst when there is no network benefit. However, note that in average an ARSA-solution is above 85% as good as an optimal solution if the network benefits are identical and zero. This

is much better than the worst-case performance guarantees $\frac{1}{2+r}$ and $\max\{\frac{1}{2}, \frac{1}{2+r}\}$ obtained in Proposition 3 and 4. This allows decision makers to be more self-confident in applying ARSA to problems in practice.

Network benefit distribution	Average		Minimum	
	z^{ARSA}/z^*	z^{GA}/z^*	z^{ARSA}/z^*	z^{GA}/z^*
Zero	0.864	0.497	0.533	0.149
Identical	0.944	0.893	0.452	0.206
Random	0.947	0.938	0.714	0.867

Table 5: Impact of network benefit

Finally, we examine the impact of the effective demand function with Table 6. As the first step of ARSA is to use a kink function to approximate the true one, it is intuitive that ARSA's performance is better when the function itself is a kink one. For a smooth $g(\cdot)$ function, we design ARSA by simply choosing a kink function to approximate the given function from above (cf. the definition of $g^K(\cdot)$ in (2)). While this is naive and simple to implement, ARSA can still achieve above 85% of the solution quality of an optimal solution in average. Whether a better way of approximation exists deserves future investigation.

Function form	Average		Minimum	
	z^{ARSA}/z^*	z^{GA}/z^*	z^{ARSA}/z^*	z^{GA}/z^*
Kink	0.99	0.843	0.891	0.258
Smooth	0.861	0.791	0.452	0.149

Table 6: Impact of functional form

5.4 Computation time

To analyze the time complexity of ARSA, note that it is mainly governed by solving the linear relaxation of the subproblems. To solve a general linear program with m functional constraints by the simplex method, the time needed is roughly proportional to m^3 (Hillier and Lieberman, 2014). For an n-location instance of our facility location, ARSA need to solve n LP problems, each has $n+3$ functional constraints. Therefore, the time complexity of ARSA with the simplex method embedded is $O(n^4)$.

To verify the above time complexity analysis and provide a comparison between ARSA and finding an exact solution, we conduct another numerical study. For each level of instance scale $n = 10$, 20, ..., and 100, we consider the two levels of net stand-alone benefits and three levels of network benefits to form six scenarios. For each scenario, we again generate 100 instances. While we set up each instance by assuming a kink $g(\cdot)$ function, we exclude smooth $g(\cdot)$ functions from this numerical study to allow CPLEX to solve for an optimal solution. The average computation times per instance by each method for each instance scale are then recorded in Table 7 and plotted in Figure 4. The computation times of GA are intentionally set to be roughly the same as those of ARSA and are thus omitted.

n	ARSA	CPLEX
10	123.92	23.82
20	268.5	77.97
30	514.6	112.6
40	991.57	459.47
50	1936.12	4450.53
60	3585.87	49754.48
70	6574.62	–
80	11662.45	–
90	20846.22	–
100	30944.2	–

Table 7: Computation time comparison (in milliseconds)

From Table 7 and Figure 4, it can indeed be verified that the computation time of ARSA is proportional to n^4 while that of CPLEX is proportional to 2^n. In fact, if we fit these data into regression models, each with n^4 and 2^n as the independent variable and the computation time as the dependent one, the R^2 values are both above 0.99. Though the $O(n^4)$ complexity of ARSA is not truly satisfactory, an instance with 100 candidate locations can be solved in around 30 seconds, and one with 300 locations is expected to be solved in around 40 minutes. As facility location problems are typically at the strategic level and do not need to be solved frequently, such a computation time should be acceptable in practice.

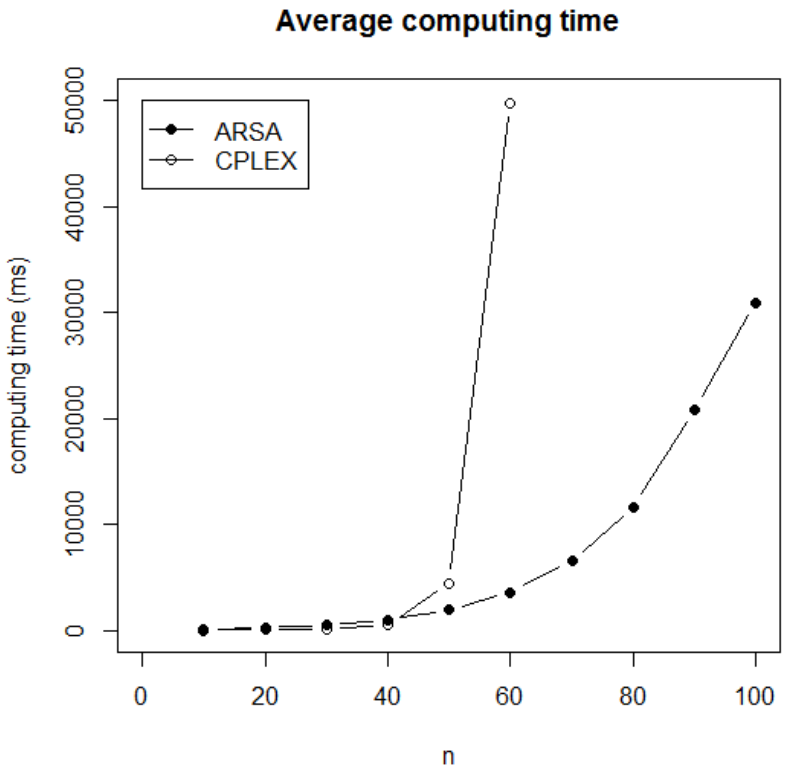

Figure 4: Average computing time of ARSA

6 Concluding remarks

In several service facility location problems such as building vehicle sharing sites, endogenous consumer demand and network effect play critical roles in finding an optimal way of building facilities. In this study, we capture these important effects into our facility location model. The solvability of our model depends on the shape of the concave effective demand function. When it is linear, we design a polynomial-time algorithm to find an optimal solution. When it is nonlinear, we show that it is NP-hard and design a polynomial-time heuristic algorithm to find a near-optimal solution. For two classes of special cases, our algorithm is proved to possess worst-case performance guarantees. Through a numerical study, our algorithm's average-case performance is shown to be much better than the guarantees in general. It is also demonstrated that our algorithm can indeed find near-optimal solutions and outperforms the genetic algorithms in most cases.

To apply the model and algorithm in practice, practitioners need to put efforts in parameter estimation for s_i and t_{ij} (h_j is the construction cost of building facility j and can be directly estimated). To estimate t_{ij}, one approach can be to monitor the current number of people

traveling between locations i and j through mass transportation and private car. If the number is high, t_{ij} can be set high accordingly. Similarly, a questionnaire-based survey can help reveal people's willingness to use the two facilities at a pair of locations. In the questionnaire, one may provide an interviewee all the potential locations and ask her/him to select, say, five pairs of locations that she/he travels in between the most. By combining the answers from a sufficiently large number of interviewees, we will discover the potential needs of each pair of facilities and thus suggest the values for t_{ij}. If there is an old system existing (e.g., when one copies a public bike system from one city to another city), the transaction data from the old system may certainly be used in estimating the attractiveness of the new system. The above approaches may all be applied to the estimation of s_i with some modifications. In fact, because one needs to have both facilities built to travel in between, for transportation systems typically t_{ij} dominates s_i. If a researcher has only limited budget/time to do parameter estimation for a transportation system, we suggest to set $s_i = 0$ and focus on the estimation of t_{ij} only.[6]

There are several ways to extend this research. First, it is unclear whether there can still be a worst-case performance guarantee if the assumptions made in this study are relaxed. In particular, the case with different network benefits among locations is definitely worth of investigation. This study can serve as a good starting point. It is also a promising direction to find a way of approximating a smooth effective demand function by a kink function without losing the performance guarantee. Second, as our problem is only proved to be weakly NP-hard, a pseudopolynomial algorithm which uses dynamic programming may be designed to find an exact solution, at least for some special cases. Such a pseudopolynomial exact algorithm may complement our polynomial approximation algorithm. Finally, most (if not all) competitive facility location studies ignore the issue of parameter estimation, and most (if not all) demand estimation studies ignore the competitive nature among facilities. We hope to help connect these two streams in the future.

[6]Some recent works adopt various data analytics techniques to do the estimation; see, e.g., Hsieh et al. (2015), Tiwari and Kaushik (2014), and the references therein. Interested readers may want to study these works. Nevertheless, most of these works only estimate the consumer demand at a location *before* a facility is built there. The endogeneity that new demand will be created *after* building a facility is widely considered difficult to estimate.

Acknowledgment

We thank the editor-in-chief José Oliveira and four anonymous reviewers for their detailed comments and many valuable suggestions that significantly enhanced the quality of this work. We also thank Zuo-Jun (Max) Shen for his helpful advising at the beginning stage of this work and Bertrand M.T. Lin and Kwei-Long Huang for their constructive comments at the final stage. Finally, we thank Po-Hsuan Chiang, Chien-Lin Chang, and Tzu-Hsiang Chien for helping us refining the numerical study. All remaining errors are our own.

References

Aboolian, R., O. Berman, D. Krass. 2007a. Competitive facility location and design problem. *European Journal of Operational Research* **182**(1) 40–62.

Aboolian, R., O. Berman, D. Krass. 2007b. Competitive facility location model with concave demand. *European Journal of Operational Research* **181**(2) 598–619.

Ageev, A., M. Sviridenko. 1999. An 0.828 approximation algorithm for uncapacitated facility location problem. *Discrete Applied Mathematics* **93**(2-3) 149–156.

Ahmed, S., A. Atamtürk. 2011. Maximizing a class of submodular utility functions. *Mathematical Programming* **128**(1) 149–169.

Beresnev, V.L. 2014. On the competitive facility location problem with free choice of supplier. *Automation and Remote Control* **75**(4) 668–676.

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.

Cornuejols, G., M. Fischer, G. Nemhauser. 1977a. Location of bank accounts to optimize float:

an analytic study of exact and approximation algorithms. *Management Science* **23**(8) 789–810.

Cornuejols, G., M. Fischer, G. Nemhauser. 1977b. On the uncapacitated location problem. *Annals of Discrete Mathematics* **1** 163–178.

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.

Hillier, F., G. Lieberman. 2014. *Introduction to Operations Research*. 10th ed. McGraw Hill, USA.

Hsieh, H.-P., C.-T. Li, S.-D. Lin. 2015. Estimating potential customers anywhere and anytime on location-based social networks. *The Proceedings of the 30th European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD'15)*. Porto, Portugal, 576–592.

Karakitsiou, A. 2015. *Modeling Discrete Competitive Facility Location*. Springer, Germany.

Küçükayadın, H., N. Aras, İ.K. Altinel. 2011. Competitive facility location problem with attractiveness adjustment of the follower: a bilevel programming model and its solution. *European Journal of Operational Research* **208**(3) 206–220.

Küçükayadın, H., N. Aras, İ.K. Altinel. 2012. A leader-follower game in competitve facility location. *Computers and Operations Research* **39**(2) 437–448.

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.

Mel'nikov, A.A. 2014. Randomized local search for the discrete competitive facility location problem. *Automation and Remote Control* **75**(4) 700–714.

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 functionsi. *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.

Tiwari, S., S. Kaushik. 2014. User category based estimation of location popularity using the road GPS trajectory databases. *Geoinformatica: An International Journal* **4**(2) 20–31.

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.