Park-and-Ride Facility Location Selection under Nested Logit Demand Function

Sang Hyun Kim¹
Korea Aerospace University, Goyang-si, Gyeonggi-do, 10540, Republic of Korea
Sangho Shim²∗
Robert Morris University, Moon Township, PA 15108, USA

Abstract
Park-and-ride (P&R) facilities are car parks where users can transfer to public transportation, by which they reach their final destinations. Commuters can use P&R facilities or choose to travel by car to their destinations, and individual choice behavior is assumed to follow a logit model. The P&R facility location problem identifies locations for a fixed number of P&R facilities from among potential locations such that the number of users of the P&R facilities is maximized. This problem has previously been formalized under a multinomial logit (MNL) demand function. However, as it imposes the strong condition of the independence of irrelevant alternatives (IIA), the MNL model is unable to represent the real-world P&R facility location problem exactly. Respecting the nested structure of individual choice behavior, we generalize the MNL model to a nested logit (NL) model and develop two computational methods—neighborhood search and randomized rounding—to solve large-scale P&R facility location problems under the NL demand function. The neighborhood search method first finds one feasible solution and then improves the feasible solution to the next one along an edge of the polyhedron whose vertices are the feasible solutions. The neighborhood search method is randomized to create an adaptive randomized rounding procedure. Computational experiments verified that the computational methods were able to solve the nonlinear optimization problem under the NL demand function on 1,000 medium-scale instances to exact optimality rapidly. Specifically, the methods were verified to solve the MNL model to exact optimality 10,000 times faster than the mixed integer linear programming formulation described in the literature. We performed additional computational experiments to assess the performance of our computational methods on a variety of large-scale instances. Our computational analysis also elucidates the difference between the MNL and NL models.

Keywords: traffic, park-and-ride facility location problem, decision-dependent demand, nested logit model, randomized rounding

*Corresponding author
Email address: shim@rmu.edu (Sangho Shim)
¹Assistant Professor, School of Air Transport, Transportation, and Logistics
²Associate Professor, School of Engineering, Mathematics and Science
1. Introduction

A transit facility is a place that provides access to transit services [12]. This paper focuses on park-and-ride (P&R) facilities, which include bus, train, and air mobility stations; generally, people drive to reach these facilities, park their cars, and then transfer to public transportation. As the cost of construction for transit facilities is high and it is difficult to change their locations once they are built, the location and number of new transit facilities must be carefully determined through quantitative and qualitative analyses of relevant factors such as cost, demand, and level of service. The problem examined in this paper assumes that a large number of new transit facilities are to be built because of the introduction of a new mode of transportation such as urban air mobility. This park-and-ride facility location problem (P&R FLP) finds a fixed number of optimal P&R locations, maximizing the total transportation demand served by the P&R facilities (e.g., the total number of P&R users or the modal share of P&R facilities).

The P&R FLP is a $p$-hub location problem whose objective is to identify locations for a fixed number ($p$) of facilities (in this paper, $p$ is denoted by $N$ to avoid confusion with a probability $p$). Goldman [13] first addressed the network hub location problem and O’Kelly [36] presented the first mathematical formulation of a hub location problem through a study of airline passenger networks. Later, Campbell [10] presented a mixed integer linear programming (MILP) formulation of the $p$-hub median problem. Based on the $p$-hub formulation, Aros-Vera et al. [3] proposed a $p$-hub approach that utilizes a spatial optimization model taking into account origin–destination (O-D) trips and considers transportation demand to determine the proportion of users patronizing the facility. In this study, we solve the P&R FLP introduced by Aros-Vera et al. [3] by generalizing the objective function of the multinomial logit demand to the nested logit demand function.

Typically, transportation demand is analyzed using a four-step model that comprises trip generation, trip distribution, mode choice, and route assignment [33]. The first step generates the total number of origin and destination trips, and the second step determines the O-D trips by matching origins with destinations.
(For this study, we assumed that the O-D trips are publicly known from previous studies such as those using the Korea Transport Database [26] or from other published works.) Then, given the O-D trips, the modal share of a particular transportation mode (i.e., the probability that that mode is chosen) is analyzed, usually using a multinomial logit (MNL) model because of its flexibility and computational efficiency [32]. A distinguishing property of the MNL model is the independence of irrelevant alternatives (IIA): the ratio of the probabilities that two respective alternatives are chosen is independent of the existence or characteristics of other alternatives [30]. However, the IIA assumption is invalid if some alternatives share unobservable attributes [7]. As a result, the probabilities for choosing such alternatives (those sharing some (unmodeled) attributes) are overestimated by the MNL model. The well-known red/blue bus problem provides an example of such overestimation of probabilities when buses are being chosen [30].

To resolve the IIA property issue, the nested logit (NL) model was proposed, which reflects correlations between alternatives [6, 31]. The NL model is represented by a tree structure that groups alternatives that have unobserved attributes in common. Fig. 1 shows a nested structure of private car and P&R options, where the P&R facilities are grouped in the same subtree. When there exists such a nested structure of transit options based on common unobserved attributes, the NL model estimates modal shares better than the MNL model. Fig. 2 shows that the MNL model underestimates the private car demand, which is indicated by the red bars [3]. A commuter first chooses between using a private car or public transportation; the commuter who decides to use public transportation then chooses one of the P&R facilities. Ignoring this nested structure of individual choice behavior, the MNL model treats all nine options (the private car and the eight P&R options) as being on the same level.

Together with certain constraints, logit models can be used for the objective function of FLPs. Benati and Hansen [8] used an MNL model to formulate the objective function of a competitive FLP and computed its upper bounds. Aros-Vera et al. [3] utilized another MNL model to calculate decision-dependent demand for P&R facilities, which constitutes their objective function. They developed MILP formulations that assume the IIA property for MNL models. Jokar [19] modified the linearized model of Aros-Vera et al. [3] to formulate a facility-and-hub location problem. López-Ospina et al. [29] estimated customer demand with a constrained MNL model and solved an FLP by considering costs and quality of service.

However, as Haase and Müller [15] pointed out, there are limitations associated with the IIA property of the MNL model, specifically in spatial choice applications. They utilized a mixed MNL model for a free school-choice problem without assuming any particular distribution for random utility. Using the general random utility model, they simulated students based on population characteristics and solved the problem for the simulated students. Basciftci et al. [5] considered the effect of facility location on demand uncertainty and proposed a distributionally robust optimization model that minimizes worst-case costs. Other transportation

---

Footnote: Fig. 2 compares the modal shares estimated by NL and MNL models for 40 commuters that have a private car option plus the eight transit options marked by asterisks. The locations of these eight P&R options have been optimally selected from among 30 candidates. (See Experiment 1 in Section 5.)
researchers [40, 21, 37] reached a consensus that the NL model is more appropriate than the MNL model for the passenger travel data from the Korea Transport Database [26], especially when the dependence between alternatives cannot be ignored.

As López-Ospina et al. [29] stated, the more accurate the estimation of transportation demand, the better the decision regarding facility locations. Because it is apparent that P&R facilities share common unobserved attributes, an NL model can estimate transportation demand more accurately than an MNL model. With this in mind, we propose an approach for solving the P&R FLP introduced by Aros-Vera et al. [3] by generalizing the objective function of the MNL demand to the NL demand function. Based on the geometry of the polyhedron whose vertices are the feasible solutions, this study developed two computational methods, neighborhood search and randomized rounding, to solve the P&R FLP under the NL demand function.

Like the simplex method, the neighborhood search method starts at one vertex of the polyhedron whose vertices are the feasible solutions and moves along an edge to the next vertex (called a neighbor), thereby improving the demand function. The neighborhood search method iterates the process of moving from one vertex to the next along an edge until there is no better neighbor remaining. Based on the same polyhedral geometry, the adaptive randomized rounding procedure randomizes the neighborhood search method, searching for a better random solution (an approximate neighbor) near the best solution found so far.

In this study, computational experiments were performed to verify that the two computational methods solve the nonlinear optimization problem for medium-scale instances under the NL demand function to exact optimality extremely rapidly. It was also verified that the methods solve P&R FLPs under the MNL demand function to exact optimality much faster than the MILP formulation of the MNL model introduced.
by Aros-Vera et al. [3]. We performed additional computational experiments to assess the performance of the proposed computational methods on a variety of large-scale instances for which solutions have not been efficiently found in the literature. Our computational experiments also examine the difference between the MNL and the NL models.

In Section 2, we develop an NL model as a generalization of the MNL demand function of the P&R FLP introduced by Aros-Vera et al. [3]; then, we discuss the relationship between the MNL model and the NL model and revisit the MILP formulation developed by Aros-Vera et al. [3] for the MNL model. Section 3 explores the geometry of the polyhedron whose vertices are the feasible solutions, and a neighborhood search method is developed for solving the P&R FLP under the NL demand function. In Section 4, the neighborhood search method is randomized, and the adaptive randomized rounding procedure is developed. Section 5 describes the conditions and reports the results of the computational experiments conducted to assess the performance of the proposed methods; it also presents a sensitivity analysis of the correlation between P&R locations. Finally, Section 6 summarizes the findings of the study and presents our conclusions.

2. Nested Logit Demand Function

2.1. Nested Logit Model

The NL model can estimate modal shares better than the MNL model can when travelers choose a transit option from among multiple transit options that share common unobserved attributes. The NL model is represented by a tree structure, in which alternatives having unobserved attributes in common are grouped. For example, bus and subway options can be grouped together as public transportation, and their P&R facilities may be grouped in the same subtree. Fig. 1 shows an example of a nested structure that consists solely of private car and P&R options. For simplicity of modeling, we assume that each traveler either uses a private car or chooses to park and ride and that no other mode of transportation is available. (The proposed model can easily be extended to a general model with multiple transportation modes, such as one consisting of multiple subtrees representing a private car, P&R facilities, buses, and subways.) Private cars move travelers directly from origin to destination. In contrast, travelers who wish to use the P&R mode move from their origin to a P&R facility and then transfer to public transportation to reach their final destination. In our model, all P&R facilities share some common unobserved attributes other than location.

MNL and NL models are widely used to analyze transportation demand. Both models calculate the demand using the observed utility of alternatives \( V \), which generally consists of travel time and cost. Holguín et al. [17] proposed a generalized utility for car and P&R modes as follows:

\[
V_c^j = c_{\text{time}}TT_c^j + c_{\text{cost}}TC_c^j + c_{\text{dist}}TD_c^j
\]

\[
V_p^{ij} = c_{\text{time}}TT_p^{ij} + c_{\text{wait}}WT_p^{ij} + c_{\text{cost}}TC_p^{ij} + c_{\text{dist}}TD_p^{ij}.
\]

\( V_c^j \) and \( V_p^{ij} \) denote the (observed) utility of using a private car and P&R facility \( i \), respectively, for O-D trip \( j \); and \( c_{\text{time}}, c_{\text{cost}}, c_{\text{dist}}, \) and \( c_{\text{wait}} \) are coefficients for the (in-vehicle) travel time, travel cost, travel distance,
and wait \( (i.e., \text{out-of-vehicle}) \) time, respectively. \( TT^j_c, TC^j_c, \text{and} TD^j_c \) denote the travel time, cost, and distance of using a private car for O-D trip \( j \), and \( TT^p_{ij}, WT^p_{ij}, TC^p_{ij}, \text{and} TD^p_{ij} \) denote the travel time, wait time, cost, and distance, respectively, of using P&R facility \( i \) for O-D trip \( j \).

The P&R FLP investigated in this study determines a fixed number \( (N) \) of optimal locations for P&R facilities to be built. The value of \( N \) is determined from the budget available for P&R by dividing the available budget by the average cost of building a P&R facility. The \( N \) locations selected for the P&R facilities will be indicated by binary variables \( (x_i \in \{0, 1\} : i \in P) \) satisfying

\[
\sum_{i \in P} x_i = N.
\]

That is, \( x_i = 1 \) if facility \( i \in P \) is selected from among all of the candidates \( P \), and \( x_i = 0 \) otherwise.

The objective function of the P&R FLP maximizes the demand for P&R \( (i.e., \text{the total number of P&R users}) \). For an O-D trip \( j \in T \), the total number \( R_j \) of travelers is given, and multiple P&R facilities are available. The number of P&R users for the O-D trip is the product of the total number of travelers on the O-D trip \( (R_j) \) and the probability of their choosing P&R options for this O-D trip \( (p^p_j) \). This study uses the NL model to calculate the probability of their choosing P&R options, as the NL model considers the correlation between P&R facilities. If there were no limits on the number of P&R facilities to be built, the optimal combination of P&R facilities would include all of the candidates \( P \). Because of the budget limitation, however, only a subset of the candidate P&R facilities can eventually be built \( (i.e., N < |P|) \); thus, the demand for unselected P&R candidate facilities must be excluded from the demand function for the expected total number of P&R users.

In this study, the NL demand function uses the nested structure shown in Fig. 1, where all P&R options are in the same subtree. The first decision of a commuter for an O-D pair \( j \in T \) is whether to drive the car or take public transportation. It is represented by

\[
p^c_j + p^p_j = \frac{e^{V^c_j}}{e^{V^c_j} + e^{\Gamma^p_j}} + \frac{e^{\lambda \Gamma^p_j}}{e^{V^c_j} + e^{\lambda \Gamma^p_j}} = 1, \forall j \in T,
\]

where \( T \) denotes the set of all O-D trips, \( \lambda \) is a logsum parameter, and \( \Gamma^p_j \) is the logsum of the exponential of the P&R utilities for O-D trip \( j \); i.e.,

\[
\Gamma^p_j = \ln \sum_{i \in P} e^{V^p_{ij} / \lambda x_i}, \forall j \in T.
\]

For O-D trip \( j \), the two terms of \( (3) \) are, respectively, the probability that the private car mode of transportation is chosen and the probability that the P&R mode of transportation is chosen:

\[
p^c_j = \frac{e^{V^c_j}}{e^{V^c_j} + e^{\lambda \Gamma^p_j}}, \forall j \in T,
\]

\[
p^p_j = \frac{e^{\lambda \Gamma^p_j}}{e^{V^c_j} + e^{\lambda \Gamma^p_j}}, \forall j \in T.
\]

If P&R facility \( i \) is not selected by the P&R FLP, the utility of P&R facility \( i \) \( (i.e., V^p_{ij}) \) is not included for the calculation of \( \Gamma^p_j \). Specifically, the probability that a P&R facility is chosen for O-D trip \( j \) \( (i.e., p^p_j) \) excludes the utilities of the candidates \( i \in P \) for which \( x_i = 0 \).
Decomposing the probability \( p_{ij}^P \) that the P&R option is chosen for O-D trip \( j \in T \), the decision-dependent probability that a specific P&R facility \( i \in P \) is chosen is then represented by

\[
p_{ij}^P = p_j^P \frac{e^{V_{ij}/\lambda x_i}}{\sum_{k \in P} e^{V_{kj}/\lambda x_k}} = \frac{e^{\lambda \Gamma_j} e^{V_{ij}/\lambda x_i}}{e^{V_j} + e^{\lambda \Gamma_j} \sum_{k \in P} e^{V_{kj}/\lambda x_k}}, \quad \forall i \in P, \forall j \in T.
\]  
(6)

Thus, \( p_{ij}^P \) is nonzero if and only if \( x_i \) is not zero, and the objective function of the P&R FLP is the demand function

\[
\sum_{i \in P} \sum_{j \in T} R_{ij} p_{ij}^P = \sum_{j \in T} R_j p_j^P,
\]
which is the expected total number of P&R users.

Now, the P&R FLP under the NL demand function is formalized as a nonlinear mathematical programming formulation as follows:

Maximize \( \sum_{i \in P} \sum_{j \in T} R_{ij} p_{ij}^P \)  
(7)

subject to

\[
\sum_{i \in P} x_i = N
\]  
(8)

\[
\Gamma_j^P = \ln \sum_{i \in P} e^{V_{ij}/\lambda x_i}, \quad \forall j \in T
\]  
(9)

\[
p_{ij}^P = \frac{e^{\lambda \Gamma_j} e^{V_{ij}/\lambda x_i}}{e^{V_j} + e^{\lambda \Gamma_j} \sum_{k \in P} e^{V_{kj}/\lambda x_k}}, \quad \forall i \in P, \forall j \in T
\]  
(10)

\[
x_i \in \{0, 1\}, \forall i \in P.
\]  
(11)

Equation (8) ensures that only \( N \) park-and-ride facilities are to be built. As (9) and (10) are functions of \( x = (x_i : i \in P) \) and can be substituted for \( p_{ij}^P = p_{ij}^P(x) \) in the objective function, Equation (8) is the only constraint of the formulation. If \( \lambda \neq 1 \), Equations (9) and (10) do not allow linearization of the nonlinear program.

2.2. Relationship to Multinomial Logit Model

The MNL model is a special case of our NL model. The logsum parameter \( \lambda \) of the NL model is bounded by zero and one. When \( \lambda = 1 \), P&R facilities are uncorrelated (i.e., there is no nested structure), and

\[
e^{\lambda \Gamma_j^P} = \sum_{i \in P} e^{V_{ij} x_i}.
\]

Then, (4) and (6) are

\[
p_j^* = \frac{e^{V_j}}{e^{V_j} + \sum_{i \in P} e^{V_{ij} x_i}}, \quad \forall j \in T
\]  
(12)

\[
p_{ij}^P = \frac{e^{V_{ij} x_i}}{e^{V_j} + \sum_{k \in P} e^{V_{kj} x_k}}, \quad \forall i \in P, \forall j \in T
\]  
(13)
Then, the IIA condition between private car and P&R facility $i$ for O-D trip $j$ is

$$\frac{p_{ij}^p}{p_j^c} = \frac{e^{V_j x_i}}{e^{V_j}}, \quad \forall i \in P, \quad \forall j \in T,$$

(14)

by dividing (13) by (12). The IIA condition means that the ratio of $p_{ij}^p$ to $p_j^c$ depends only on the utilities of P&R facility $i$ with $x_i = 1$ (i.e., a facility to be built) and of the private car. Note that (12) and (13) are equivalent to (14) taken together with

$$p_j^c + \sum_{i \in P} p_{ij}^p = 1.$$

Similarly, the IIA condition between P&R facilities $i$ and $k$ for O-D trip $j$ is

$$\frac{p_{kj}^p}{p_j^c} = \frac{e^{V_j x_k}}{e^{V_j x_i}}, \quad \forall i, k \in P, \quad \forall j \in T.$$

(15)

Unlike the P&R FLP with the NL model presented in (7)–(11), the P&R FLP with the MNL model can be expressed as an MILP formulation. Of the three linearized formulations presented by Haase and Müller [16], the formulation of Aros-Vera et al. [3] is adopted for this study because it is quickly solved. The MILP formulation of the P&R FLP with the MNL model is

Maximize \( \sum_{i \in P} \sum_{j \in T} R_j p_{ij}^p \)  

subject to

$$\sum_{i \in P} x_i = N$$

(17)

$$p_{ij}^p \leq x_i, \quad \forall i \in P, \quad \forall j \in T$$

(18)

$$p_j^c + \sum_{i \in P} p_{ij}^p = 1, \quad \forall j \in T$$

(19)

$$p_{ij}^p \leq \frac{e^{V_j x_i}}{e^{V_j}}, \quad \forall i \in P, \quad \forall j \in T$$

(20)

$$p_j^c \leq \frac{e^{V_j}}{e^{V_j}} p_{ij}^p + (1 - x_i), \quad \forall i \in P, \quad \forall j \in T$$

(21)

$$p_{ij}^p \leq \frac{e^{V_j x_k}}{e^{V_j x_i}} p_{kj}^p + (1 - x_k), \quad \forall i, k \in P, \quad \forall j \in T$$

(22)

$$x_i \in \{0, 1\}, \quad \forall i \in P$$

(23)

$$p_j^c \geq 0, \quad \forall j \in T$$

(24)

$$p_{ij}^p \geq 0, \quad \forall i \in P, \quad \forall j \in T.$$  

(25)

Equations (12) and (13) are formalized by the linear constraints (18)–(21). Equation (18) forces $p_{ij}^p = 0$ if $x_i = 0$, as given in (13). Equation (19) ensures that the sum of all probabilities (e.g., $p_j^c$ and $p_{ij}^p$) is equal to one for each O-D trip. The IIA condition of (14) with $x_i = 1$ is the equation induced by (20) and (21). Similarly, the IIA condition of (15) is ensured by (22). Note that Equation (22) is redundant as (12) and
are implied by (18)–(21). In a similar manner, we may alternatively delete Equations (20) and (21) instead of Equation (22) (see Haase and Müller [16]). To further strengthen the formulation, Krohn et al. [28] have introduced a tighter bound on $p_{ij}$ in (18).

3. Neighborhood Search Method

In this section, we develop a neighborhood search (NS) method to solve the P&R FLP by considering the polyhedron whose vertices are the feasible solutions $x^{\text{INT}} \in \{0, 1\}^{|P|}$ satisfying (8). We do not consider the entire polyhedron but rather the adjacency graph defined by the vertices and edges of the polyhedron. Like the simplex method, the neighborhood search method starts at one vertex of the polyhedron and moves along an edge to the next vertex (called a neighbor), improving the demand function. The method iterates until there is no better neighbor at a local maximum.

Let $\text{INT}(N)$ denote the set of $x^{\text{INT}} \in \{0, 1\}^{|P|}$ satisfying (8), which are the feasible solutions to an $N$-hub facility location problem. Let $\text{DEMAND}_\lambda(x)$ denote the objective function (7) in which $p^p_{ij} = p^p_{ij}(x)$ are functions of $x$ defined by (9) and (10). More precisely, $p^p_{ij}(x)$ is determined by variable vector $x$ along with parameter $\lambda$, and therefore $\text{DEMAND}_\lambda(x)$ is determined by $x$ and $\lambda$. We will denote $\text{DEMAND}_\lambda(x)$ simply as $\text{DEMAND}(x)$ when $\lambda$ does not need to be specified. Then, the P&R FLP can be simply written as

$$\max \left\{ \text{DEMAND} \left( x^{\text{INT}} \right) : x^{\text{INT}} \in \text{INT}(N) \right\}.$$ (26)

The relaxation of $\text{INT}(N)$ is

$$\text{REL}(N) = \left\{ \tilde{x} \in [0, 1]^{|P|} : \tilde{x} \text{ satisfying (8)} \right\},$$

where $[0, 1]$ is the interval $\{ \tilde{x}_i \in \mathbb{R} : 0 \leq \tilde{x}_i \leq 1 \}$ between 0 and 1. It is easy to see that $\text{INT}(N)$ is the set consisting of the vertices of the polyhedron $\text{REL}(N)$:

**Proposition 1.** The feasible solutions $x^{\text{INT}} \in \text{INT}(N)$ are the vertices of the polyhedron $\text{REL}(N)$.

Considering $\text{INT}(N)$ as a subset of $\text{REL}(N)$, we will refer to the feasible solutions $x^{\text{INT}}$ interchangeably as the integer solutions, particularly when we distinguish the integer solutions from the fractional solutions $\tilde{x} \in \text{REL}(N) \setminus \text{INT}(N)$.

It is also easy to see that two vertices $x^{\text{INT}(1)}$ and $x^{\text{INT}(2)}$ are adjacent (i.e., connected by an edge of the polyhedron $\text{REL}(N)$) if their 1-norm distance is 2:

**Proposition 2.** Two vertices $x^{\text{INT}(1)}$ and $x^{\text{INT}(2)}$ are adjacent on the polyhedron $\text{REL}(N)$ if and only if

$$\left\| x^{\text{INT}(1)} - x^{\text{INT}(2)} \right\|_1 = \sum_{i \in P} \left| x_i^{\text{INT}(1)} - x_i^{\text{INT}(2)} \right| = 2.$$
Figure 3: Adjacency graph of REL($N$) selecting $N = 2$ locations among $|P| = 4$ candidates

When two vertices $x^{\text{INT}(1)}$ and $x^{\text{INT}(2)}$ are adjacent, one is said to be a neighbor of the other. Note that each vertex has $N \times (|P| - N)$ neighbors. On the adjacency graph $G = (V, E)$ of the polyhedron REL($N$), where $V$ is the set of the vertices and $E$ is the set of the edges of the polyhedron, a local optimum $x^{\text{LOPT}}$ is defined to have all inferior neighbors $x^{\text{NBR}(k)}$; i.e.,

$$\text{DEMAND}(x^{\text{LOPT}}) \geq \text{DEMAND}(x^{\text{NBR}(k)}) \quad \text{for} \quad k = 1, \ldots, N \times (|P| - N).$$

For further details regarding the vertices and edges of a polyhedron, the reader may refer to Bertsimas and Tsitsiklis [9].

Since $\dot{x} = (\dot{x}_i = N/|P| \quad \text{for} \quad i \in P)$ is an inner point that satisfies Equation (8) and the strict inequalities $0 < \dot{x}_i = N/|P| < 1$ for $i \in P$, the dimension of the polyhedron REL($N$) is $|P| - 1$. In particular, $\dot{x}$ is the average of the integer solutions $x^{\text{INT}} \in \text{INT}(N)$:

**Proposition 3.** The average of the integer solutions $x^{\text{INT}} \in \text{INT}(N)$ is $\dot{x}$; i.e.,

$$\dot{x} = \frac{\sum \{x^{\text{INT}} \in \text{INT}(N)\}}{|\text{INT}(N)|}.$$

**Proof.** Each facility $i \in P$ is a member of the same number $\binom{|P|-1}{N-1}$ of combinations of $N$ facilities. Then,

$$\dot{x}_1 = \frac{N}{|P|} = \frac{\binom{|P|-1}{N-1}}{\binom{|P|}{N}} = \frac{\binom{|P|-1}{N-1}}{|\text{INT}(N)|} = \frac{\sum_{x^{\text{INT}} \in \text{INT}(N)} x_1^{\text{INT}}}{|\text{INT}(N)|}.$$

We refer to $\dot{x}$ as the centroid of REL($N$) as it is the center of weight of the polyhedron REL($N$). For further details regarding inner points and the dimensionality of a polyhedron, the reader may refer to Nemhauser and Wolsey [35].

Fig. 3 depicts the adjacency graph $G = (V, E)$ of REL($N$) selecting $N = 2$ locations from among $|P| = 4$ candidates. The graph has $|V| = 6$ vertices and $|E| = 12$ edges. Each vertex is adjacent to
$N \times (|\mathcal{P}| - N) = 4$ neighbors. Two vertices are adjacent whenever exactly two components differ. The centroid is $\hat{x} = (0.5, 0.5, 0.5, 0.5)$. The polyhedron $\text{REL}(N)$ is three-dimensional as $|\mathcal{P}| - 1 = 3$.

**Algorithm 1** Neighborhood Search Method

1: Start with any integer solution $x^{\text{INT}}$
2: Set $\text{is\_local\_optimum} = \text{False}$
3: while $\text{is\_local\_optimum} = \text{False}$ do
4: Set $\text{is\_local\_optimum} = \text{True}$
5: for $i \in \text{supp}(x^{\text{INT}})$ and $k \in \mathcal{P} \setminus \text{supp}(x^{\text{INT}})$ do
6: Set $x^{\text{NBR}} \leftarrow x^{\text{INT}}$
7: $x^{\text{NBR}}_i \leftarrow 0$
8: $x^{\text{NBR}}_k \leftarrow 1$
9: if DEMAND($x^{\text{INT}}$) $<$ DEMAND($x^{\text{NBR}}$) then
10: Update $x^{\text{INT}} \leftarrow x^{\text{NBR}}$
11: Set $\text{is\_local\_optimum} = \text{False}$
12: break
13: end if
14: end for
15: end while
16: return the local maximum $x^{\text{LOPT}} = x^{\text{INT}}$

Pseudocode for the neighborhood search method is given in Algorithm 1. In Line 1, the search starts with an integer solution $x^{\text{INT}}$, which is the initial best known integer solution. In Lines 3–15, the method moves the best known integer solution $x^{\text{INT}}$ to its neighbor, improving DEMAND($x^{\text{INT}}$), until $x^{\text{INT}}$ reaches a local optimum $x^{\text{LOPT}}$. During this process, $\text{is\_local\_optimum} = \text{False}$ indicates that $x^{\text{INT}}$ is not guaranteed to be a local optimum, in which case it continues moving. In Lines 5–14, the method enumerates the neighbors $x^{\text{NBR}}$ of $x^{\text{INT}}$. If a superior neighbor is found (Line 9), the method updates the best known integer solution with the superior neighbor by setting $x^{\text{INT}}$ to $x^{\text{NBR}}$ (Line 10) and sets $\text{is\_local\_optimum}$ to $\text{False}$ so that the best known integer solution will continue moving (Line 11). If there is no superior neighbor (i.e., $\text{is\_local\_optimum} = \text{True}$), the method returns the local optimum $x^{\text{LOPT}} = x^{\text{INT}}$. Note that the 1-norm distance between $x^{\text{INT}}$ and $x^{\text{NBR}}$ equals 2; i.e., $x^{\text{NBR}}$ is given by swapping a location $i \in \text{supp}(x^{\text{INT}}) = \{i \in \mathcal{P} : x^{\text{INT}}_i = 1\}$ of the support of $x^{\text{INT}}$ and a non-support location $k \in \mathcal{P} \setminus \text{supp}(x^{\text{INT}})$.

As will be shown in Section 5, computational experiments verified that the neighborhood search method solves the medium-scale P&R FLP to exact optimality extremely rapidly. Specifically, it solves the MNL model of the medium-scale instances to exact optimality 10,000 times faster than the MILP formulation introduced by Aros-Vera et al. [3]. On the large-scale instances, which the MILP formulation cannot solve, the neighborhood search method solves the MNL model within 30 s.
4. Adaptive Randomized Rounding Procedure

In this section, the neighborhood search method described in the previous section is randomized and developed into an adaptive randomized rounding (ARR) procedure. Given a fractional solution \( \tilde{x} \), which we will refer to as a seed, randomized rounding finds an integer solution \( x^{\text{INT}} \) near the seed. A randomized rounding procedure finds multiple integer solutions and selects the best known integer solution \( x^{\text{BKI}} \). An adaptive randomized rounding procedure moves the seed \( \tilde{x} \) toward the best known integer solution \( x^{\text{BKI}} \) to find a better integer solution near \( x^{\text{BKI}} \) (i.e., a better integer solution in the approximate neighborhood of \( x^{\text{BKI}} \)).

The traditional randomized rounding procedure first solves the relaxation

\[
\text{DEMAND}(\tilde{x}^*) = \max \{ \text{DEMAND}(\tilde{x}) : \tilde{x} \in \text{REL}(N) \} \tag{27}
\]

and then finds an integer solution \( X^{\text{INT}} = x^{\text{INT}} \) based on the fractional optimal solution \( \tilde{x}^* \) such that

\[
P [X^i_{\text{INT}} = 1] = \tilde{x}^*_i \text{ for } i \in \mathcal{P}, \tag{28}
\]

where \( X \) denotes the random variable for an integer solution \( x \). (For further details regarding (traditional) randomized rounding, the readers may refer to Motwani and Raghavan [34] and Vazirani [38].) A randomized rounding procedure conducts multiple trials to find integer solutions \( x^{\text{INT}(t)}, t = 1, \ldots, T \), and selects the solution found (BFS); i.e.,

\[
\text{BSF} = \arg \max \{ \text{DEMAND} \left( x^{\text{INT}(t)} \right) : t = 1, \ldots, T \}.
\]

As it is not easy to solve the relaxation (27), the initial seed for our randomized rounding procedure will be the uniform distribution \( \hat{x} = (\hat{x}_i = N/|\mathcal{P}| : i \in \mathcal{P}) \), and the seed will move to converge to the best known integer solution \( x^{\text{BKI}} \) so far, searching for better integer solutions in the vicinity of the best known integer solution. Because \( \hat{x} \) is interpreted as the likelihood of \( x^{\text{INT}} \), we consider only \( \tilde{x} \in [0, 1]|\mathcal{P}| \subseteq \mathbb{R}|\mathcal{P}| \), with the components \( \hat{x}_i \) being between 0 and 1, throughout this study. Section 4.1 describes the three main components of our adaptive randomized rounding technique, and Section 4.2 shows the construction of the full adaptive randomized rounding procedure.

4.1. Three Main Components

4.1.1. Rounding

A rounding function \( \text{ROUND} : \mathbb{R}|\mathcal{P}| \rightarrow \text{INT}(N) \) maps a fractional solution \( \tilde{x} \in \mathbb{R}|\mathcal{P}| \) to the nearest integer solution \( x^{\text{INT}} = \text{ROUND}(\tilde{x}) \) indicating the \( N \) largest components; i.e., \( x^{\text{INT}}_i = 1 \), if and only if \( \hat{x}_i \) is one of the \( N \) largest components of \( \tilde{x} \). (A tie may be broken by a linear ordering \( \prec \) of \( \mathcal{P} \). That is, the same \( N \)-th largest components \( \hat{x}_i = \hat{x}_k \) of \( i \prec k \in \mathcal{P} \) may be rounded to \( x^{\text{INT}}_i = 0 \) and \( x^{\text{INT}}_k = 1 \).) Then, an integer solution \( x^{\text{INT}} \) is rounded to itself; i.e.,

\[
\text{ROUND}(x^{\text{INT}}) = x^{\text{INT}}.
\]
The hyperplane $H = \{ \tilde{x} \in \mathbb{R}^3 : \tilde{x}_1 + \tilde{x}_2 + \tilde{x}_3 = 2 \}$ is partitioned into three regions $H \cap \text{ROUND}^{-1}(x^{\text{INT}})$ corresponding to the three integer solutions $x^{\text{INT}} \in \text{INT}(N)$ where $N = 2$ and $|\mathcal{P}| = 3$. The rounding function maps a fractional solution $\tilde{x}$ in the region of $\text{ROUND}^{-1}(x^{\text{INT}}) \subseteq [0, 1]^{|\mathcal{P}|}$ to $x^{\text{INT}}$.

The inverse image of the rounding function partitions the entire space $\mathbb{R}^{|\mathcal{P}|}$ into regions $\text{ROUND}^{-1}(x^{\text{INT}})$ for $x^{\text{INT}} \in \text{INT}(N)$. In Fig. 4, the regions $\{ \text{ROUND}^{-1}(x^{\text{INT}}) : x^{\text{INT}} \in \text{INT}(N) \}$ partition the hyperplane $H$ defined by (8), where $N = 2$ and $\mathcal{P} = \{1, 2, 3\}$. The rounding function maps a fractional solution $\tilde{x}$ lying in the region $\text{ROUND}^{-1}(x^{\text{INT}})$ to $x^{\text{INT}}$.

4.1.2. Randomization

Our randomized rounding procedure perturbs a given seed $\tilde{x}$ randomly so that the perturbed seed $\tilde{x}^{\text{PTB}}$ may lie in $\text{ROUND}^{-1}(x^{\text{INT}}) \subseteq [0, 1]^{|\mathcal{P}|}$, satisfying (32). A random perturbation increases each component $\tilde{x}_i$ by uniform distribution $U[\tilde{x}, 1]$ in the interval $[\tilde{x}, 1]$; i.e.,

$$\tilde{x}_i^{\text{PTB}} = \tilde{x}_i + (1 - \tilde{x}_i) \tilde{\pi}_i \text{ with } \tilde{\pi}_i \in U[0, 1] \text{ for } i \in \mathcal{P}. \quad (29)$$

If $\tilde{x}_i$ equals 1, $\tilde{x}_i^{\text{PTB}}(\tilde{x}) = 1$ is always the highest component, and facility $i$ is selected (i.e., $x_i^{\text{INT}} = 1$). Therefore, if the seed is an integer solution $\tilde{x} = x^{\text{INT}}$, the perturbed solution lies in $\tilde{x}^{\text{PTB}} \in \text{ROUND}^{-1}(x^{\text{INT}})$; i.e.,

$$\text{ROUND} (\tilde{x}^{\text{PTB}}) = x^{\text{INT}} \text{ whenever } \tilde{x} = x^{\text{INT}}. \quad (30)$$

In contrast, $x_i^{\text{INT}}$ does not need to be zero for a facility candidate $i$ with $\tilde{x}_i = 0$, giving the facility candidate a positive opportunity (i.e., $\tilde{x}_i^{\text{PTB}} > 0$) to be selected, whenever $\tilde{x}$ is not an integer solution. In the remainder of this paper, we denote $x^{\text{INT}} = \text{ROUND} (\tilde{x}^{\text{PTB}})$ by $x^{\text{INT}}(\tilde{x})$.

4.1.3. Moving Seed

As it is not easy to solve the relaxation (27), the initial seed for our randomized rounding procedure will be the uniform distribution $\tilde{x}^{(0)} = \tilde{x} = (\tilde{x}_i = N/|\mathcal{P}| : i \in \mathcal{P})$, and the seed will move to converge to
the best known integer solution \( x^{\text{BKI}} \) as better integer solutions are sought in the vicinity of \( x^{\text{BKI}} \). The procedure first finds an integer solution \( x^{\text{INT}(0)} = \text{ROUND} \left( \tilde{x}^{\text{PTB}(0)} \right) \), where \( \tilde{x}^{\text{PTB}(0)} \) is given by perturbing \( \tilde{x}^{(0)} = \hat{x} \). The first integer solution is the best known integer solution \( x^{\text{BKI}} = x^{\text{INT}(0)} \). At each trial \( t \geq 1 \), the procedure then moves the last seed \( \tilde{x}^{(t-1)} \), converging to the best known integer solution \( x^{\text{BKI}} \) by exponential smoothing:

\[
\tilde{x}^{(t)} = (1 - \alpha(t)) \tilde{x}^{(t-1)} + \alpha(t)x^{\text{BKI}},
\]

where \( 0 < \alpha(t) < 1 \). When \( \text{DEMAND} \left( x^{\text{BKI}} \right) < \text{DEMAND} \left( x^{\text{INT}(t)} \right) \), the procedure updates the best known integer solution \( x^{\text{BKI}} \) to \( x^{\text{INT}(t)} \).

The initial seed \( \tilde{x}^{(0)} = \hat{x} \) is the centroid, which is the probability distribution of \( X^{\text{INT}}(\hat{x}) \):

**Proposition 4.** At the centroid \( \hat{x} \), the probability distribution is \( \hat{x} \) itself; i.e.,

\[
P \left[ X_i^{\text{INT}}(\hat{x}) = 1 \right] = \hat{x}_i.
\]

**Proof.** On the seed of \( \hat{x} \), each candidate \( i \in P \) can be selected equally likely. Therefore,

\[
P \left[ X_i^{\text{INT}}(\hat{x}) = 1 \right] = \frac{\sum_{i \in P} P \left[ X_i^{\text{INT}}(\hat{x}) = 1 \right]}{|P|} = \sum_{i \in P} E \left[ X_i^{\text{INT}}(\hat{x}) \right] = \frac{E \left[ \sum_{i \in P} X_i^{\text{INT}}(\hat{x}) \right]}{|P|} = \frac{N}{|P|} = \hat{x}_i.
\]

That is, the centroid \( \hat{x} \) satisfies Equation (28) in replace of the optimal solution \( \hat{x}^* \) to the relaxation (27). At the centroid, the randomized rounding procedure finds all integer solutions equally likely.

Beginning with the centroid \( \tilde{x}^{(0)} = \hat{x} \), the seed \( \tilde{x}^{(t)} \) (at trial \( t \geq 1 \)) will move to converge to the best known integer solution by exponential smoothing (31), searching for better integer solutions in the vicinity of the best known integer solution \( x^{\text{BKI}} \). Instead of strictly satisfying (28), the probability distribution will be proportional to the seed \( \tilde{x} \); more precisely,

\[
\tilde{x}_i < \tilde{x}_k \Rightarrow P \left[ X_i^{\text{INT}} = 1 \right] < P \left[ X_k^{\text{INT}} = 1 \right] \text{ for } i \neq k \in P.
\]

Converging toward \( x^{\text{BKI}} \), the seed \( \tilde{x}^{(t)} \) finds the same integer solution \( x^{\text{INT}(t)} = x^{\text{INT}}(\tilde{x}^{(t)}) = x^{\text{BKI}} \) increasingly frequently. When the seed is too close to the best known integer solution, our randomized rounding procedure cannot find a better integer solution and will reset the seed back to the centroid \( \hat{x} \).

### 4.2. Full Procedure

In this section, the full adaptive randomized rounding (ARR) procedure is developed. For this, we will define the rule for slowing the convergence speed (i.e., smoothing constant) \( \alpha(t) \) in (31) and the rule for resetting the seed.

The initial seed for our adaptive randomized rounding procedure is the centroid \( \tilde{x}^{(0)} = \hat{x} \). The centroid finds the initial integer solution \( x^{\text{BKI}} = x^{\text{INT}}(\hat{x}) \). Each trial \( t \geq 1 \) moves the seed, converging to the best known integer solution \( x^{\text{BKI}} \) by exponential smoothing (31). When \( \tilde{x}^{(t)} \) is too close to \( x^{\text{BKI}} \), a run of many
trials will find the same integer solution \( x^{\text{INT}(t)} = x^{\text{BKI}} \), in which case we reset the seed \( \tilde{x}^{(t)} \) back to the centroid \( \hat{x} \).

For computational convenience, we further relax the relaxation \( \text{REL}(N) \) to \([0, 1]^{|\mathcal{P}|}\) and consider its centroid \( \hat{x} \) given by

\[
\hat{x}_i = 0.5 \text{ for } i \in \mathcal{P}.
\]  

(33)

This is equivalent to \( \hat{x} = (\hat{x}_i = N/|\mathcal{P}|) \) in that

\[
P \left[ X_i^{\text{INT}} (\hat{x}) = 1 \right] = P \left[ X_i^{\text{INT}} (\hat{x}) = 1 \right] = N/|\mathcal{P}|.
\]

In general, \( \tilde{x}' \in [0, 1]^{|\mathcal{P}|} \) is projected to \( \tilde{x} \in [0, 1]^{|\mathcal{P}|} \), satisfying \( \sum_{i \in \mathcal{P}} \tilde{x}_i = N \) as follows.

\textbf{Theorem 1.} Let \( \tilde{x}' \in [0, 1]^{|\mathcal{P}|} \), and let

\[
\tilde{x} = 1 - \frac{|\mathcal{P}| - N}{\|1 - \tilde{x}'\|_1} (1 - \tilde{x}').
\]  

(34)

Then

\[
P \left[ X_i^{\text{INT}} (\tilde{x}') = x^{\text{INT}} \right] = P \left[ X_i^{\text{INT}} (\tilde{x}) = x^{\text{INT}} \right] \text{ for } x^{\text{INT}} \in \text{INT}(N).
\]  

(35)

\textbf{Proof.} An integer solution \( x^{\text{INT}} (\tilde{x}') \) selects the largest \( N \) values \( \tilde{l}'_i \) of

\[
\tilde{l}'_i = \tilde{x}'_i + (1 - \tilde{x}'_i) \tilde{\pi}_i
\]

with \( \tilde{\pi}_i \in U [0, 1] \) for \( i \in \mathcal{P} \). This is equivalent to selecting the smallest \( N \) values of

\[
1 - \tilde{l}'_i = (1 - \tilde{x}'_i) (1 - \tilde{\pi}_i) = \left( \frac{1 - \tilde{x}'_i}{|\mathcal{P}| - N} \right) (1 - \hat{x}_i) (1 - \tilde{\pi}_i) \text{ for } i \in \mathcal{P}.
\]  

(36)

Since \( \left( \frac{1 - \tilde{x}'_i}{|\mathcal{P}| - N} \right) \) is a constant factor across \( i \in \mathcal{P} \) in (36), selecting the smallest \( N \) values of \( 1 - \tilde{l}'_i \) is equivalent to selecting the smallest \( N \) values of \( 1 - \tilde{l}_i = (1 - \hat{x}_i) (1 - \tilde{\pi}_i) \), which is equivalent to \( x^{\text{INT}} (\tilde{x}) \) selecting the largest \( N \) values of \( \tilde{l}_i = \tilde{x}_i + (1 - \tilde{x}_i) \tilde{\pi}_i \). This completes the proof. \( \square \)

During the convergence to the best known integer solution, a decelerator slows the speed of convergence. To determine \( \alpha(t) \), we define the root-mean-square deviation (RMSD) of \( \tilde{x}^{(t)} \) with respect to 0.5, i.e.,

\[
\text{RMSD}(\tilde{x}^{(t)}) = \sqrt{\frac{\sum_{i \in \mathcal{P}} (\tilde{x}_i^{(t)} - 0.5)^2}{|\mathcal{P}|}},
\]  

(37)

which ranges from 0 at \( \tilde{x} \) to 0.5 at an integer solution \( x^{\text{INT}} \). Thus, as \( \tilde{x}^{(t)} \) converges to the best known integer solution \( x^{\text{BKI}} \), the RMSD converges to 0.5. The smoothing constant \( \alpha(t) \) is determined by decelerator (DEC) which slows \( \alpha = \frac{1}{2} \) (at \( \tilde{x} \)) to \( \alpha = \frac{1}{1 + e^{4 \text{RMSD}(\tilde{x}^{(t-1)})}} \) (at \( x^{\text{BKI}} \)):

\[
\alpha(t) = \text{DEC}(t - 1) = \frac{1}{1 + e^{4 \text{RMSD}(\tilde{x}^{(t-1)})}}.
\]  

(38)
Algorithm 2 Adaptive Randomized Rounding

1: Initialize $\tilde{x}^{(0)} \leftarrow \tilde{x} = (\tilde{x}_i = 0.5$ for $i \in \mathcal{P}$); $x^{\text{BKI}} \leftarrow \text{INT}(\tilde{x})$; $t \leftarrow 1$; $n^{\text{local}} \leftarrow 0$

2: while all termination criteria are not met do

3: Adjust smoothing constant $\alpha(t)$ by Decelerator (38)

4: Move $\tilde{x}^{(t)}$ by exponential smoothing (31) with the smoothing constant $\alpha(t)$

5: Find $x^{\text{INT}}(t) = \text{INT}(\tilde{x}^{(t)})$

6: if DEMAND ($x^{\text{INT}}(t)$) $>$ DEMAND ($x^{\text{BKI}}$) then

7: $x^{\text{BKI}} \leftarrow x^{\text{INT}}(t)$

8: $n^{\text{local}} \leftarrow 0$

9: else if $x^{\text{INT}}(t)$ $=$ $x^{\text{BKI}}$ then

10: $n^{\text{local}} \leftarrow n^{\text{local}} + 1$

11: $p^R \leftarrow \min \left( \frac{n^{\text{local}}}{20}, 1 \right) \times \text{RMSD}(\tilde{x}^{(t)})$

12: Reset $\tilde{x}^{(t)} \leftarrow \tilde{x}$ and $n^{\text{local}} \leftarrow 0$ with probability of $p^R$

13: else

14: $n^{\text{local}} \leftarrow 0$

15: end if

16: $t \leftarrow t + 1$

17: end while

18: Return the best known integer solution $x^{\text{BKI}}$
Pseudo-code for this ARR procedure is given in Algorithm 2. The number of consecutive trials in which the same solution $x^{BK1}$ is found is denoted by $n_{\text{local}}$. As the same solution $x^{BK1}$ is found in the run of trials (i.e., $n_{\text{local}}$ increases), the probability that the seed will be reset to $\bar{x}$ increases. In particular, the probability of reset $p^R$ reaches RMSD ($\tilde{x}^{(t)}$) after a run of 20 trials that find the same $x^{BK1}$. A computational time limit is set as the termination criterion for the full procedure. In the computational experiments on medium-scale instances, finding the global optimal solution was set as an additional termination criterion, as described in the next section.

5. Numerical Experiments

5.1. Experimental Design

Two numerical experiments, on medium- and large-scale instances, respectively, were conducted to evaluate the performance of the proposed algorithms (the neighborhood search (NS) method and the adaptive randomized rounding (ARR) procedure) and to compare the P&R FLPs as modeled using the MNL and NL demand functions. In the experiments, the logsum parameter $\lambda$ of the NL demand function was set to 0.5 except when the $\lambda$ sensitivity analysis was performed. As references for the performance comparison, three other algorithms were implemented on the large-scale instances: the conventional ARR procedure introduced by Chopra et al. [11], denoted as Con-ARR; an ARR with an accelerator (ACC) (instead of a decelerator (DEC)), denoted as Acc-ARR; and a genetic algorithm (GA).

Acc-ARR is the adaptive randomized rounding procedure of Algorithm 2 with the smoothing constant $\alpha(t)$ in Line 3 determined by accelerator ACC:

$$\alpha(t) = \text{ACC}(t-1) = \frac{1}{1 + e^{-8(\text{RMSD}(\tilde{x}^{(t-1)}) - 0.5)}}.$$  \hfill (39)

The smoothing constant increases from $\frac{1}{1+e^4}$ at $\bar{x}$ to $\frac{1}{2}$ at $x^{\text{INT}}$. Details of Con-ARR and GA are presented in Appendix A and Appendix B respectively.

In both experiments, a P&R FLP was solved for a randomly generated metropolitan area that consisted of a number of central business districts (CBDs) and neighborhoods (residential areas). The number of CBDs was a random number between 3 and 5, and the CBDs were randomly located on a radius between 1 and 2 distance units from the center of the metropolitan area. The number of neighborhoods was a random number between 5 and 10, and the center of each neighborhood was randomly located on a radius between 6 and 10 distance units from the center of the metropolitan area. Note that the distance unit is arbitrary; it could be taken as a kilometer or a mile, for example. We generated 1,000 random instances of the P&R FLP and executed the proposed algorithms for a Monte Carlo statistical analysis.

The origin of each trip was located in a neighborhood and was uniformly distributed in an area $[-1, 1] \times [-1, 1]$ around the neighborhood center; thus, a neighborhood is a cluster of origins. The destination of each trip was assumed to be a CBD. Thus, multiple origins were paired with the same destination. P&R facilities
had to be located between the neighborhoods and the CBDs, and the candidate P&R locations for a random instance were located on a radius between 5 and 7 distance units from the center of the metropolitan area.

In Experiment 1, the P&R FLP was solved on 1,000 medium-scale instances. The number of O-D pairs was 40, and eight P&R locations were selected from among 30 candidate locations. In Experiment 2, the P&R FLP was solved on 100 large-scale instances, in which the number of O-D pairs was 1,000, and 35 P&R locations were selected from among 100 candidate locations. The two experiments differed in problem size (i.e., the number of O-D pairs and candidate P&R locations), but their geometric structures were the same. Figure 5 shows an illustrative example of six neighborhoods and five CBDs for Experiment 1 on the left and Experiment 2 on the right.

In the experiments, the proposed algorithms were terminated when a time limit was reached. The time limits for Experiment 1 and Experiment 2 were set to 1 min and 30 min, respectively, based on preliminary trial runs. Table 1 summarizes the conditions for each experiment. Because of the large search space for Experiment 2, a brute-force search was not applicable.

|                               | Experiment 1 | Experiment 2 |
|-------------------------------|--------------|--------------|
| Number of O-D pairs           | 40           | 1,000        |
| Number of P&R candidates      | 30           | 100          |
| Number of selected P&R locations | 8           | 35           |
| Brute-Force search            | Yes          | No           |
| Time limit of heuristic algorithms | 1 min      | 30 min       |
| Number of instances           | 1,000        | 100          |

The experiments were conducted using MATLAB R2020b on a computer with an Intel Core i9-9900T.
CPU and 16GB of RAM. For the experiments, the negative of the travel distance (Euclidean distance) was used as the observed utility for simplicity of calculation. Thus, the longer the distance traveled, the less attractive the alternative. Thus, the utility values for the car option and the P&R option are, respectively,

\[ V_c^j = -TD^j_c = -\sqrt{(X^d_j - X_o^j)^2 + (Y^d_j - Y_o^j)^2}, \quad \forall j \in T \]  
\[ V_p^i_j = -TD^p_{ij} = -\sqrt{(X^p_i - X_o^j)^2 + (Y^p_i - Y_o^j)^2} - \sqrt{(X^d_j - X^p_i)^2 + (Y^d_j - Y^p_i)^2}, \quad \forall i \in P, \quad \forall j \in T, \]  

where \((X_o^j, Y_o^j)\); and \((X^d_j, Y^d_j)\) are the X- and Y-coordinates of the origin and destination of O-D trip \(j\), and \((X^p_i, Y^p_i)\) are those for P&R candidate location \(i\). Note that O-D trip \(j\) comprises \{(\(X_o^j, Y_o^j\)), (\(X^d_j, Y^d_j\))\}, and the utility of a private car is always greater than that of any P&R option. The number of travelers on each O-D trip was set to one for simplicity; i.e.,

\[ R_j = 1, \quad \forall j \in T. \]

As given in (1) and (2), the utility (and therefore the modal share) of the P&R facility may be affected by other factors, such as the wait time at a P&R facility and the frequency of the transit service. For example, Fig. 6 shows the impact of wait time on the modal share of the P&R option in Experiment 2. The wait time may be added to the utility of the P&R option as follows:

\[ V_p^i_j = -TD^p_{ij} * (1 + \Delta W), \]

where \(\Delta W\) is the relative wait time at the P&R facility and can range from 0 to 0.3. As expected, the modal share of the P&R option decreases as the wait time increases because the P&R utility decreases. Likewise, other factors included in the P&R utility may influence the decision making of potential P&R users. However, we did not observe any unexpected effects of other factors in preliminary experiments, and therefore we considered only the travel distance in the experiments. Thus, the negative of the travel distance was used as the utility for both the private car mode and the P&R mode. For a MATLAB code and the data for the numerical experiments, the readers may refer to the data repository provided in Kim and Shim [22].

5.2. Comparison of Algorithm Performance

To evaluate and compare the performance of the proposed algorithms, we performed the two experiments described in the previous section. For Experiment 1, the number of feasible solutions was \(\binom{30}{8}\) = 5,852,925, and a brute-force search was computationally tractable for this medium-scale problem. We used the exact optimal solution as the benchmark to evaluate the quality of the solution produced by the proposed algorithms. In contrast, for Experiment 2, the number of feasible solutions was \(\binom{100}{35}\), which is approximately 1.87 \times e + 20 times greater than that for Experiment 1. As the number of candidate P&R locations increases, the search space increases exponentially, and a brute-force search cannot solve the large-scale instances to exact optimality within a reasonable time. Therefore, instead of using the exact optimal solution, for Experiment 2 we used the best solution found as the benchmark.
Table 2 shows the computational performance of the proposed algorithms and the comparison algorithms in Experiment 1. All algorithms successfully found the exact optimal solutions to all of the medium-scale instances (“Optimality”). NS initially converged to sub-optima on seven of the 1,000 instances. Within the computational time limit, the subsequent NS was performed with a random initial solution $x^\text{INT}(\bar{x})$. All seven sub-optima were improved to the exact optimal solutions after two or more repetitions of the NS with the random initial solutions. The brute-force (BF) search provided the exact optimal solutions as the benchmark for these medium-scale instances, and the termination criterion for all of the algorithms was the exact optimality of the solution. The “Mean Time,” “Std Time,” “Max Time,” and “Mean Iterations” rows show the average, standard deviation, and maximum computation time and the average number of iterations on the 1,000 instances until exact optimality was reached. For NS, the number of iterations is the number of edges traversed; for ARR, Acc-ARR, and Con-ARR, it is the number of trials; and for GA, it is the number of generations. NS and ARR were much faster than the brute-force search. They also outperformed Acc-ARR, Con-ARR, and GA in computational time.

Table 3 shows the computational performance of the proposed algorithms and the comparison algorithms in Experiment 2 for the time limit of 30 min. As explained above, the brute-force search cannot solve the large-scale instances, and thus there is no benchmark solution for exact optimality. The benchmark solution for each instance was the best solution found (BSF) by all five algorithms. “Number of instances with the BSF” shows that the proposed algorithms reached the BSF on all 100 instances along with Acc-ARR and Con-ARR, whereas GA did not reach any BSF. In fact, all but GA together reached the same BSF on every instance. “Mean ratio of demand to the BSF” shows the ratio of the best demand found by the algorithm.
Table 2: Comparison of the Proposed Algorithms in Experiment 1 on One Thousand Instances

|               | BF | NS | ARR | Acc-ARR | Con-ARR | GA |
|---------------|----|----|-----|---------|---------|----|
| Optimality    | 100%| 100%| 100%| 100%    | 100%    | 100%|
| Mean Time     | 158 s| 0.04 s| 0.05 s| 0.19 s  | 0.12 s  | 0.31 s|
| Std Time      | 1.31 s| 0.01 s| 0.17 s| 0.55 s  | 0.21 s  | 0.46 s|
| Max Time      | 165 s| 0.10 s| 3.6 s | 15 s    | 5.9 s   | 7.5 s|
| Mean Iterations| N/A| 954| 1,012| 3,742   | 2,190   | 522 |

The computational time for each algorithm to reach its best solution is denoted in the table by $t_f$. The two proposed algorithms reached the BSF much earlier than the computational time limit of 30 min, and each was faster than Acc-ARR, Con-ARR, and GA. In particular, they reached the BSF much faster than GA reached its own best solutions, which were inferior to the BSF. “Mean iterations until time limit” shows the average number of iterations of the algorithm within the computational time limit of 30 min. NS found the BSF on each of the large-scale instances without repetition (i.e., beginning with random solution $x^{\text{INT}}(\bar{x})$ only once). As the number of iterations within the time limit is nearly equal across ARR, Acc-ARR, and Con-ARR, the average number of iterations to reach the BSF is proportional to the mean $t_f$ for each of the three randomized rounding procedures.

Table 3: Comparison of the Proposed Algorithms in Experiment 2 on 100 Instances (Time Limit = 30 minutes)

| Number of instances with the BSF | NS | ARR | Acc-ARR | Con-ARR | GA |
|----------------------------------|----|-----|---------|---------|----|
| Mean ratio of demand to the BSF  | 1  | 1   | 1       | 1       | 0.965 |
| Mean $t_f$                       | 29 s| 70 s| 271 s   | 149 s   | 1,037 s |
| Std $t_f$                        | 10 s| 64 s| 219 s   | 169 s   | 525 s |
| Max $t_f$                        | 60 s| 316 s| 1,284 s | 845 s   | 1,793 s |
| Mean iterations until time limit | 16,088| 552,695| 552,736| 552,206| 31,581 |

We performed a sensitivity analysis to understand the effect of various numbers of selected P&R locations ($N$). We chose 100 random instances of Experiment 1 with 40 O-D and 30 candidate P&R locations and applied the algorithms to these instances to select the optimal $N$ locations that maximize demand, for $N$ values of 6, 12, 18, and 24. All of the algorithms except GA found the exact optimal solutions for all $N$ values. Figure 7 shows the average computational time taken by the algorithms (corresponding to “Mean Time” in Table 2) for the four values of $N$. The Y axis shows the computational time on a logarithmic scale. As expected, BF took significantly more computational time than did the other algorithms. The computational times of BF for $N$ values of 6 and 24 are nearly equal because $\binom{30}{6} = \binom{30}{24}$; similarly, the computational
times for $N$ values of 12 and 18 are nearly equal. The computational time of BF for an $N$ value of 12 or 18 is much greater than that for an $N$ of 6 or 24 because $\binom{30}{12} > \binom{30}{6}$. The figure also shows that NS and ARR were faster than the other algorithms for the various value of $N$, ARR begin slightly faster than NS for the $N$ values of 12, 18, and 24.

![Figure 7: Impact of $N$ on the Average Convergence Speed](image)

Now, we compare the performance of the proposed algorithms with that of the MILP formulation introduced by Aros-Vera et al. [3] on 100 medium-scale P&R FLPs (from Experiment 1) under the MNL demand function. The P&R FLPs select 15 P&R locations from among 30 candidate locations, on which BF is intractable because of the large search space. The MILP (16)-(25) was solved by Gurobi Optimizer version 9.0.3, and NS and ARR solved (26) with $\lambda = 1$. Table 4 shows the comparison of the performance of the algorithms. The proposed algorithms solved the medium-scale MNL instances to exact optimality 10,000 times faster than the MILP (MILP : NS = 295s : 0.02s) where the computational time for MILP was the system time excluding the time used for construction and reading of the model. Moreover, the proposed algorithms solved the large-scale MNL models (from Experiment 2) within two minutes, similar to the performance reported in Table 3 whereas the MILP is intractable on the large-scale MNL models generating the error message of Out-Of-Memory.

5.3. Comparison of Logit Models

The second purpose for conducting the numerical experiments was to analyze the dependency of the optimal P&R locations on the demand modeling. As stated previously, the MNL model and the NL model are based on different assumptions regarding the correlation between P&R locations. Specifically, the MNL
Table 4: Comparison of Commercial Solver (MILP) and the Proposed Algorithms in Experiment 1 with MNL model

|                     | MILP | NS  | ARR  |
|---------------------|------|-----|------|
| Optimality          | 100% | 100%| 100% |
| Mean computational time | 295 s | 0.02 s | 0.06 s |
| Std computational time | 484 s | 0.01 s | 0.05 s |
| Max computational time | 2,639 s | 0.05 s | 0.2 s |

model assumes the logsum parameter $\lambda$ to be 1, whereas the NL model estimates $\lambda$ by using maximum likelihood estimation [25]. As no a priori knowledge of $\lambda$ for the P&R FLP was available, the experiments reported in the previous section arbitrarily assumed $\lambda = 0.5$. Because $\lambda$ is an important parameter indicating the underlying correlation between P&R locations [24], we conducted a sensitivity analysis of $\lambda$ on the selected P&R locations and the P&R demand. ARR was used to solve the same 100 instances sampled from the 1,000 instances of Experiment 1 with $\lambda$ values ranging from 0.1 to 1 in steps of 0.1.

Figure 2 shows a comparison of the probabilities associated with choosing each option (the private car option and one of the P&R options) and the optimal combination of P&R locations for an instance of Experiment 1. The $N$ P&R locations selected as optimal are marked with an asterisk. The figure shows that for the same P&R FLP, the two models gave not only different values for the P&R demand but also different combinations of the $N$ P&R locations. As can be seen, P&R locations 5, 7, 15, 18, 22, and 28 were selected by both models, but the models differed in their selections for two P&R locations. Figure 8 shows the differences in the eight P&R locations selected from among 30 candidates for various value of $\lambda$ with respect to the MNL model (i.e., for $\lambda = 1$). It was found that the difference increases as the correlation between P&R locations increases (i.e., as $\lambda$ decreases). Note that the red bars in the figure are median values and the red crosses are outliers.

Because of the IIA assumption of the MNL model, the probability that a private car is chosen is lower in the MNL model than when the NL model is used. That is, the P&R demand is overestimated by the MNL model, and a solution of the P&R FLP with the MNL model will be sub-optimal, considering the correlation between P&R locations. Fig. 9 shows the optimality of the MNL solution for the NL model with various values of $\lambda$. Let $x^{NL(\lambda)}$ be the optimal solution to the NL model with $\lambda$. Then, the optimality of the MNL solution $x^{NL(1)}$ is

$$\frac{\text{DEMAND}_\lambda (x^{NL(1)})}{\text{DEMAND}_\lambda (x^{NL(\lambda)})}.$$

For example, suppose that there is a correlation between P&R locations and that the corresponding value of $\lambda$ is 0.2. For $\lambda = 0.2$, the optimality gap is approximately 1.4%. For $\lambda < 0.33$, the gap is greater than 1%. The greater the correlation between P&R locations, the greater the optimality gap of the MNL solution.
6. Conclusion

This study proposed two heuristic algorithms that can efficiently solve a park-and-ride facility location problem under the nested logit demand function. Previously the park-and-ride facility location problem was solved under the multinomial logit demand function in mixed-integer linear programming (MILP) formulations. The multinomial logit demand function does not capture the nested structure of the private car and park-and-ride options, and the MILP formulations can solve only the medium-scale park-and-ride facility location problem. Our heuristics of neighborhood search and adaptive randomized rounding were able to solve the large-scale instances extremely rapidly under the nested logit demand function, which captures the nested structure of the alternatives. In particular, numerical experiments verified that the proposed heuristics solve the medium-scale problem under the multinomial logit demand function to exact optimality much faster than the MILP formulations. Moreover, the proposed algorithms can identify the best solutions in minutes for large-scale problems that are intractable under the MILP approach.

The two heuristic algorithms proposed by this paper can solve a large-scale park-and-ride facility location problem to select a large number of new transit facilities for a new mode of transportation such as urban air mobility. In particular, our adaptive randomized rounding procedure is not only for the polyhedral structure of a specific formulation, and can address more complicated facility location problems as well. For example, a vertiport location problem may select the optimal pairs of vertiports for the commuters who use urban air mobility from one vertiport to another. The adaptive randomized rounding procedure is also expected to efficiently solve many other large-scale vertiport location problems.
Acknowledgments

The authors gratefully thank anonymous referees for helpful comments on the earlier version of the paper. We would like to thank Editage (www.editage.co.kr) for English language editing. This work is supported by the Korea Agency for Infrastructure Technology Advancement grant funded by the Ministry of Land, Infrastructure and Transport (Grant RS-2022-00143965).

Appendix A. Conventional adaptive randomized rounding

Setting $\tilde{x}(0) = \hat{x} \in \text{REL}(N)$, the following $\tilde{x}(r)$, $r > 0$, will belong to \text{REL}(N)$ because the polyhedron contains the line segment $[x^{(r-1)}, x^{\text{BKI}}]$. Moving $\tilde{x}(r)$ in \text{REL}(N), our randomized rounding method performs the following 2 step procedure (see Fig. 10):

Step 0. Initialization. With an initial distribution $\tilde{x}(0)$ and an observed $x^{\text{BKI}} = x^{\text{INT}(0)} = x^{\text{INT}}(\tilde{x}(0))$, Step 0 sets the initial smoothing constant $\alpha(1) = 1/2$ and moves on to Step 1.

Step 1. Adjusting smoothing constant. The first step moves the last distribution $\tilde{x}(r-1)$ to $\tilde{x}(r)$ on the line segment $[\tilde{x}(0), x^{\text{BKI}}]$ by exponential smoothing [31], adjusting the smoothing constant $\alpha$, and observes a random combination $x^{\text{INT}(r)}$ of $N$ facilities using distribution $\tilde{x}(r)$.

- Stopping criterion of the whole procedure: If $\alpha(r)$ is less than a certain tolerance (e.g., tolerance = $10^{-5}$ equal to the integrality tolerance of Gurobi Optimization [14]), the whole procedure stops.
Figure 10: Flow Chart of 2 Step Procedure (where \( \text{obj}^{(r)} = \text{obj}(x^{\text{INT}(r)}) \) and \( \text{obj}^* = \text{obj}(x^{\text{BKI}}) \) where \( \text{obj} = \text{DEMAND} \)

- Criterion to go to Step 2: If \( \text{obj}^{(r)} > \text{obj}^* \), the procedure updates the best known combination \( x^{\text{BKI}} = x^{\text{INT}(r)} \), where \( \text{obj} = \text{DEMAND} \). Assuming that the speed \( \alpha(r) \) is appropriate, fix the present smoothing constant \( \alpha = \alpha(r) \), set \( r \leftarrow r + 1 \) and go to Step 2.

- Major decrease of \( \alpha \): If \( \text{obj}^{(r)} = \text{obj}^* \), assume that \( \tilde{x}(r) \) is arriving too close to \( x^{\text{BKI}} \) or too fast, and slow down the speed of the move by major decrease of the smoothing constant \( \alpha(r+1) = \alpha(r)/2 \). Reset the present distribution back to \( \tilde{x}(0) \), set \( r \leftarrow r + 1 \) and repeat Step 1.

- Minor increase of \( \alpha \): If \( \text{obj}^{(r)} < \text{obj}^* \), assume that \( \tilde{x}(r) \) is moving slow and speed up the move a little bit by minor increase of the smoothing constant \( \alpha(r+1) = 2^\left\lceil \log_2 \alpha(r) \right\rceil + \alpha(r)/2 \) such as \( \alpha(r) = 1/2 + 1/4 \) to \( \alpha(r+1) = 1/2 + 1/4 + 1/8 \). Set \( r \leftarrow r + 1 \) and repeat Step 1 (moving ahead without coming back to \( \tilde{x}^{(r+1)} = \tilde{x}(0) \)).

**Step 2. Fixed smoothing constant.** The second step moves \( \tilde{x}^{(r)} \) from \( \tilde{x}^{(r-1)} \) toward \( x^{\text{BKI}} \) by exponential smoothing (31) on the line segment \( [\tilde{x}^{(r-1)}, x^{\text{BKI}}] \) without changing the smoothing constant \( \alpha \) that is adjusted in Step 1. It observes \( x^{\text{INT}(r)} \) based on \( \tilde{x}^{(r)} \).

- Criterion to go back to Step 1: If \( \text{obj}^{(r)} = \text{obj}^* \) for a run of say 10 iterations of trials, assume that \( \tilde{x}^{(r)} \) is too close to \( x^{\text{BKI}} \) and reset \( \tilde{x}^{(r)} = \tilde{x}(0) \) initializing \( \alpha(r+1) = 1/2 \). Set \( t \leftarrow t + 1 \) and go back to Step 1.

- If \( \text{obj}^{(r)} > \text{obj}^* \), the procedure updates \( x^{\text{BKI}} = x^{\text{INT}(r)} \) with the new best value. Set \( r \leftarrow r + 1 \) and repeat Step 2 from the present distribution \( \tilde{x}^{(r-1)} \) toward the new best known integer solution \( x^{\text{BKI}} \).
If \( \text{obj}(x^{\text{INT}(r)}) \leq \text{obj}(x^{\text{BKI}}) \), set \( r \leftarrow r + 1 \) and repeat Step 2 going on from the present \( \tilde{x}^{(r-1)} \) toward the best known facility location \( x^{\text{BKI}} \).

Appendix B. Genetic Algorithm

Genetic Algorithm (GA) is a population-based heuristic algorithm \(^{18}\). It is well-known and widely adopted to solve the FLP \(^{27}\). A canonical GA is given in Algorithm 3 \(^{39}\).

**Algorithm 3** Canonical GA Algorithm

(Initialization) Randomly generate an initial population

while all termination criteria are not met do

(Evaluation) Evaluate fitness values of the population

(Selection) Randomly select chromosomes (i.e., parents) based on fitness

for every two parents do

(Crossover) Mix the parent chromosomes to form a child chromosome

end for

(Mutation) With probability of \( p^m \), randomly select genes for each child and change their values

(Next generation) chromosomes \( \leftarrow \) children chromosomes

end while

Return the best objective value

Phase 1: Initialization. Each chromosome is a binary string of length \(|P|\), and \( N \) genes (i.e., elements of a chromosome) are randomly chosen and assigned to 1. Other genes are assigned to 0. Thus, a chromosome is a feasible set of \((x_1, x_2, \ldots, x_{|P|})\). The population size (pop) is a design parameter.

Phase 2: Evaluation. The fitness value of a chromosome is equal to the objective value; i.e.,

\[
\text{Fitness} = \sum_{i \in P} \sum_{j \in T} R_j p_{ij}^p,
\]

but it can also be determined with the rank of objective value \(^{11}\).

Phase 3: Selection. The selection probability of a chromosome is proportional to its fitness, and the selection process is stochastic sampling with replacement, as in the roulette wheel method \(^{39, 20, 23}\). In a canonical GA, parents produce two children; however, in this paper, they produce one child. As a result, \( 2 \times \text{pop} \) parents are selected, and thus most chromosomes are selected multiple times as parents.

Phase 4: Crossover. A canonical GA utilizes 1-point crossover, which randomly selects a crossover point and swaps the fragmented genes between the parents. Some implementations of GA utilize 2-point crossover in which parents' middle genes between the crossover points are swapped to generate children \(^{11}\). However,
such a fragment-based crossover does not guarantee the feasibility of children chromosomes because it constrains the number of chosen P&R to be equal to $N$.

Therefore, this paper proposes a weight-based crossover operator presented in Fig. B.11. Each gene of parent chromosomes is randomly weighted between 0 and 1, and genes are added together to form a child chromosome. Then, the $N$ largest genes of the child chromosome are assigned to 1 and the other genes are assigned to 0. This crossover operator guarantees that the child chromosome is inherited only from its parents' chromosomes and the number of genes with value of 1 remains at $N$. Thus, the child chromosome is also a feasible solution of the P&R FLP.

Phase 5: Mutation. A canonical GA mutates each gene (i.e., flips a bit from 0 to 1 or vice versa) with a probability of $p^m$; however, such a mutation operator can make children chromosomes infeasible. Therefore, the mutation operator in this paper exchanges a gene with value of 1 with another gene with value of 0. Each child chromosome is subject to mutation with a probability of $p^m$, which is a design parameter.
Design parameters of GA influence the performance, specifically in terms of optimality and convergence time \cite{2}. A canonical GA has design parameters such as population size, crossover probability, and mutation probability. This paper sets the crossover probability at 1, meaning all children chromosomes are inherited from both parents. However, the other parameters such as population size and mutation probability should be determined by design parameter tuning.

Prior to the numerical experiments, Monte-Carlo experiments were conducted to tune the population size (\text{pop}) and mutation probability (\text{p}^m). Based on previous experience, \text{pop} ranges from 30 to 90 in steps of 10, and \text{p}^m ranges from 1\% to 5\% in steps of 1\%. 100 instances of Experiment 1 are solved by GA for all pairs of (\text{pop}, \text{p}^m). The exact optimal solution to instances is found using a brute-force search, and the optimality of a GA solution is the ratio of the GA solution to the brute-force solution. Because the global optimal solution \textit{(i.e., the brute-force solution)} is known, the computational times at which the GA finds the global optimal solution (\text{t}_f) are recorded. The true mean \textit{(i.e., population mean)} of \text{t}_f is denoted as \mu, and its point estimate is denoted as \hat{\mu}. The time limit is set at 20 min based on sample runs.

![Box Plot of \text{t}_f in GA Design Parameter Tuning](image)

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{image}
\caption{Box Plot of \text{t}_f in GA Design Parameter Tuning}
\end{figure}

Fig. B.12 shows \hat{\mu} and 95\% confidence intervals for \mu, only for the pairs of (\text{pop}, \text{p}^m) with which the GA found the optimal solution to all instances within the time limit. Fig. B.13 shows a box plot of \text{t}_f, also corresponding to (\text{pop}, \text{p}^m) pairs of 100\% optimality. Figs. B.12, B.13 show that the (70, 0.04) pair has the fastest convergence time on average \textit{(i.e., the smallest \hat{\mu})}, as well as, in the worst case \textit{(i.e., the smallest upper bound of \text{t}_f excluding outliers)}. Hence, the parameter pair (70, 0.04) is used for the numerical experiments, but its 95\% confidence interval for \mu overlaps with those for other (\text{pop}, \text{p}^m) pairs. For instance, (70, 0.05) can be another good parameter pair for GA.
References

[1] Andre, J., Siarry, P., Dognon, T., 2001. An improvement of the standard genetic algorithm fighting premature convergence in continuous optimization. Advances in engineering software 32, 49–60.

[2] Angelova, M., Pencheva, T., 2011. Tuning genetic algorithm parameters to improve convergence time. International Journal of Chemical Engineering 2011.

[3] Aros-Vera, F., Marianov, V., Mitchell, J.E., 2013. p-hub approach for the optimal park-and-ride facility location problem. European Journal of Operational Research 226, 277–285.

[4] Baker, J.E., 1985. Adaptive selection methods for genetic algorithms, in: Proceedings of an International Conference on Genetic Algorithms and their applications, Hillsdale, New Jersey. p. 111.

[5] Basciftci, B., Ahmed, S., Shen, S., 2021. Distributionally robust facility location problem under decision-dependent stochastic demand. European Journal of Operational Research 292, 548–561.

[6] Ben-Akiva, M.E., 1973. Structure of passenger travel demand models. Ph.D. thesis. Massachusetts Institute of Technology.

[7] Ben-Akiva, M.E., Lerman, S.R., Lerman, S.R., 1985. Discrete Choice Analysis: Theory and Application to Travel Demand. MIT Press.

[8] Benati, S., Hansen, P., 2002. The maximum capture problem with random utilities: Problem formulation and algorithms. European Journal of Operational Research 143, 518–530.

[9] Bertsimas, D., Tsitsiklis, J.N., 1997. Introduction to Linear Optimization. Athena Scientific, Belmont, MA, USA.

[10] Campbell, J., 1996. Hub location and the p-hub median problem. Operations Research 44, 1–13.

[11] Chopra, S., Kim, E., Shim, S., 2019. Adaptive randomized rounding in the big parsimony problem. URL: DOI:10.13140/RG.2.2.12140.41607.

[12] General Statutes of Connecticut, 2019. CT Gen Stat § 7-339cc. 2019.

[13] Goldman, A., 1969. Optimal location for centers in a network. Transportation Science 3, 352–360.

[14] Gurobi Optimization, 2020. Gurobi optimizer (9.0.2). URL: https://www.gurobi.com/.

[15] Haase, K., Müller, S., 2013. Management of school locations allowing for free school choice. Omega 41, 847–855.

[16] Haase, K., Müller, S., 2014. A comparison of linear reformulations for multinomial logit choice probabilities in facility location models. European Journal of Operational Research 232, 689–691.
[17] Holguín, J., Yushimito, W.F., Aros-Vera, F., Reilly, J.J., 2012. User rationality and optimal park-and-ride location under potential demand maximization. Transportation Research Part B: Methodological 46, 949–970.

[18] Holland, J.H., 1992. Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT press.

[19] Jokar, M.R.A., 2017. Facility and hub location model based on gravity rule. Computers & Industrial Engineering 109, 28–38.

[20] Karakatič, S., Podgorelec, V., 2015. A survey of genetic algorithms for solving multi depot vehicle routing problem. Applied Soft Computing 27, 519–532.

[21] Kim, I.K., Kim, K., Kim, H.C., 2005. Model specification and estimation method for traveler’s mode choice behavior in pusun metropolitan area. Journal of Korean Society of Transportation 23, 7–19.

[22] Kim, S., Shim, S., 2022. Park-and-ride. URL: https://github.com/yoyoyes777/parkandride.

[23] Kim, S.H., 2019. Receding horizon scheduling of on-demand urban air mobility with heterogeneous fleet. IEEE Transactions on Aerospace and Electronic Systems 56, 2751–2761.

[24] Koppelman, F.S., Bhat, C., 2006. A self instructing course in mode choice modeling: multinomial and nested logit models. Technical Report. US Department of Transportation Federal Transit Administration.

[25] Koppelman, F.S., Wen, C.H., 1998. Alternative nested logit models: structure, properties and estimation. Transportation Research Part B: Methodological 32, 289–298.

[26] Korea Transport Database, 2019. Passenger travel demand analysis. URL: https://www.ktdb.go.kr.

[27] Kratica, J., Tošić, D., Filipović, V., Ijubić, I., 2001. Solving the simple plant location problem by genetic algorithm. RAIRO-Operations Research-Recherche Opérationnelle 35, 127–142.

[28] Krohn, R., Müller, S., Haase, K., 2021. Preventive healthcare facility location planning with quality-conscious clients. OR Spectrum 43, 59–87.

[29] López-Ospina, H., Agudelo-Bernal, Á., Reyes-Muñoz, L., Zambrano-Rey, G., Pérez, J., 2021. Design of a location and transportation optimization model including quality of service using constrained multinomial logit. Applied Mathematical Modelling 89, 428–453.

[30] McFadden, D., 1974. Conditional logit analysis of qualitative choice behavior, in: Zarembka, P. (Ed.), Frontiers in Econometrics. Academic Press, New York. chapter 4, pp. 105–142.

[31] McFadden, D., 1978. Modelling the choice of residential location. Transportation Research Record , 72–77.
[32] McFadden, D., Train, K., Tye, W.B., 1977. An application of diagnostic tests for the independence from irrelevant alternatives property of the multinomial logit model. Transportation Research Record, 39–46.

[33] McNally, M.G., 2007. The four-step model. Emerald Group Publishing Limited.

[34] Motwani, R., Raghavan, P., 1995. Randomized Algorithms. Cambridge University Press, New York, NY, USA.

[35] Nemhauser, G.L., Wolsey, L.A., 1999. Integer and Combinatorial Optimization. John Wiley & Sons, New York, NY, USA.

[36] O’Kelly, M., 1987. A quadratic integer program for the location of interacting hub facilities. European Journal of Operational Research 32, 393–404.

[37] Park, S.J., Kim, S.S., 2007. A nested logit model of auto ownership and vehicle type choices. Journal of Korean Society of Transportation 25, 133–141.

[38] Vazirani, V.V., 2001. Approximation Algorithms. Springer Berlin Heidelberg.

[39] Whitley, D., 1994. A genetic algorithm tutorial. Statistics and computing 4, 65–85.

[40] Yun, D.S., 1997. Analysis of urban workers’ travel pattern choice behavior. Journal of Korean Society of Transportation 15, 35–51.