Mathematics Interdisciplinary Research **9** (3) (2024) 289 − 314

Original Scientific Paper

Simultaneous Location of k Portable Emergency Service Centers and Reconstruction of a Damaged Network

Sedigheh Nazari and Ardeshir Dolati

Abstract

This paper addresses the problem of optimizing the reconstruction of links in a network in the aftermath of natural disasters or human errors, such as landslides, floods, storms, earthquakes, bombing, war, etc. We aim to determine the optimal sequence for reconstructing the destroyed links within a specific time horizon, while simultaneously locating (k) portable emergency service centers (where ($k > 2$)) throughout the entire network. In this paper, the problem is considered in a tree structure. A greedy algorithm and a heuristic method, namely, maximum radius, are proposed to solve the problem. We evaluate the performance of the proposed algorithms using randomly generated data. The experimental results confirm the effectiveness of the proposed methods.

Keywords: Facility location problem, k-Center problem, Portable k-center problem.

2020 Mathematics Subject Classification: 90C27, 90B18.

How to cite this article
S. Nazari and A. Dolati, Simultaneous location of k portable emergency service centers and reconstruction of a damaged network, *Math. Interdisc. Res.* **9** (3) (2024) 289-314.

*Corresponding author (E-mail: dolati@shahed.ac.ir)
Academic Editor: Gholam Hossein Fath-Tabar
Received 15 October 2023, Accepted 27 February 2024
DOI: 10.22052/MIR.2024.253656.1443

1. Introduction

Location theory is a practical and well-known subject in the combinatorial optimization literature. It is concerned with determining the optimal locations of facilities in a given network or space, based on a specified objective function. Classical location theory comprises two primary problems: the median facility location problem and the center facility location problem. The 1-median problem aims to identify the point in the network that minimizes the sum of the weighted distances from all vertices to it. On the other hand, the 1-center problem aims to determine the point in the network where the maximum weighted distance of other vertices to it is minimized. These problems are also known as min-sum and min-max, respectively. These concepts were first introduced by Hakimi [1].

In location theory, the k-median and k-center problems arise when there are multiple facilities to locate. Kariv and Hakimi [2] showed that the k-median problem on a general network is NP-hard and presented an algorithm with a time complexity of $O(n^2k^2)$ to find the k-median of a tree (for $k > 1$). Goldman [3] offered a linear time algorithm for the 1-median problem. Daskin and Maass [4] described a linear time algorithm for the 1-median problem on trees and collected results from various literature that have used metaheuristic algorithms to solve the k-median problem. Recently, Duran-Mateluna et al. [5] investigated a Benders decomposition approach for the k-median problem, which also applies for the uncapacitated facility location problem. Wang et al. [6] introduced the first reinforcement learning-based method to solve the uncapacitated k-median problem.

Megiddo [7] proposed a linear time algorithm to solve the 1-center problem on weighted trees. Kariv and Hakimi [8] proved the NP-hardness of the k-center problem on general graphs and presented polynomial time algorithms to obtain the 1-center problem. They also presented an algorithm with a time complexity of $O(n^2 \log n)$ to find the absolute and vertex k-center on trees. Megiddo and Tamir [9] proposed an algorithm with a time complexity of $O(n \log^3 n)$ for finding the absolute k-center on a tree. Jeger and Kariv [10] gave an algorithm with a time complexity of $O(kn \log n)$ to solve this problem. Banik et al. [11] have made advances in the k-center problem and have developed improved algorithms with running times of $O(n \log n + k \log^2 n \log(n/k))$ and $O(n \log n + k^2 \log^2(n/k))$. After three decades, Wang and Zhang [12] solved an open problem raised by Megiddo and Tamir [9]. They presented an algorithm with a time complexity of $O(n \log n)$ to find the absolute k-center on trees. Wang et al. [13] defined a variation of the k-center problem, which has been applied in emergency rescue stations in the high-speed railway network.

Locating temporary service centers is a subject that has been studied by researchers. This type of problem is especially relevant during natural disasters and emergency situations, which may damage the network and require disaster management to take action. Supporting the victims and properly reconstructing the network are the main goals of disaster management. Setting up portable temporary treatment sites, such as temporary outpatient clinics, field hospitals, and temporary shelter sites, is an effective policy for helping disaster victims. Accelerating the access of citizens affected by natural disasters to medical centers is a crucial issue that can be addressed by these temporary service centers. Several studies have addressed the problem of locating temporary service centers. For example, Kılcı et al. [14] considered the problem of locating temporary shelter sites after an incident in Turkey and suggested a mixed integer linear programming mathematical model for locating shelter sites. Cavdur et al. [15] studied a two-stage stochastic program for the problem of temporary disaster response facility allocation. They considered an earthquake case study in Turkey. Huiyong et al. [16] modeled hierarchical earthquake shelter planning in an urban area as a two-stage mathematical programming model and performed a case study in Shanghai city. Celik [17] and Trivedi [18] used the decision-making trial and evaluation laboratory method to locate temporary locations in disasters. Drakaki et al. [19] considered the refugee settlement site planning problem with an intelligent multi-agent system modeling method. This method examined refugee accommodation sites in Greece. Oksuz and Satoglu [20] located temporary medical centers after natural disasters. They developed a two-stage stochastic programming model for this problem and examined this model for the real case of the Kartal District of Istanbul city for an earthquake that happened. Karatas and Yakıcı [21] studied a multiobjective facility location problem for choosing a set of temporary emergency service centers for a regional natural gas distribution company in Turkey after natural disasters.

In this paper, we focus on the problem of locating temporary service centers, which is a novel-defined problem in the context of incremental network design. Incremental network design often involves constructing network elements over different periods of time. Readers interested in this topic can refer to Sharp [22] and Hartline [23] for a comprehensive understanding of incremental network design and related issues. Several authors have studied combinatorial optimization problems from a network design perspective. For example, Hartline and Sharp [24] presented incremental forms of some combinatorial maximization problems such as knapsack, bipartite matching, and maximum flow problems. Federico et al. [25], [26] investigated the incremental knapsack problem and achieved approximate results. Bienstock et al. [27] studied a variant of the incremental knapsack problem and gave a constant factor approximation algorithm and a polynomial time approximation scheme. Recently, the incremental knapsack problem has been studied by Faenza et al. [28] and Aouad and Segev [29]. Engel et al. [30] presented a greedy algorithm for solving the incremental minimum spanning tree. The incremental version of the shortest path problem was shown to be NP-hard in Baxter et al. [31], who derived a 4-approximation algorithm. Kalinowski et al. [32] offered two mixed integer programming formulations for the incremental version of the maximum flow problem and presented some heuristic methods to solve this problem. They also analyzed the performance of the proposed MIP formulations and heuristics. Plaxton [33] was the first author, who considered the metric uncapacitated k-median and facility location problems in the context of incremental network design and presented an approximation algorithm. Du et al. [34] studied the incremental k-center problem and presented a polynomial time algorithm with a competitive ratio $\frac{3\sqrt{3}}{2}$ for a version of the problem where centers are located on the boundary of a convex polygon. The incremental connected facility location problem was first proposed in the work of Arulselvan et al. [35], who presented a mixed integer programming approach.

In this paper, we investigate the problem of reconstructing links in a network that has been damaged due to natural disasters or human errors such as landslides, floods, storms, earthquakes, bombing, war, etc. After a disaster, service centers can be destroyed or inaccessible, which necessitates the establishment of temporary and portable service centers in suitable locations while broken links are repaired over time, optimally. The order in which the damaged links are reconstructed is crucial in ensuring fair and efficient service provision to the affected population. The long-term reconstruction of damaged networks after natural disasters has been studied in transportation networks. Gokalp et al. [36] examined road networks after disasters that destroyed links, which required recovery over a time horizon, and developed a heuristic.

The present study addresses a problem in which the locations of service centers are temporary, and the number of centers is predetermined. The objective is to select the appropriate order for the reconstruction of broken links in each time period while providing optimal service centers to individuals. The aim of the study is to minimize the costs incurred by individuals, which can include the time spent accessing the service centers. To tackle this problem, we utilize the topology tree as the underlying network structure, which becomes an unconnected network, or forest, after a disaster. A greedy algorithm and a heuristic method are proposed to optimally repair the tree structure, and their efficiency is compared by using numerical experiments. The results demonstrate the effectiveness of the proposed methods in solving the problem.

The rest of this paper is organized as follows. In Section 2, we provide preliminary information related to the problem. In Section 3, we formulate our new problem. In Section 4, we present a greedy algorithm and a heuristic method. We consider a case in which the costs of reallocating facilities is taken into accoun, and this is discussed in Section 5. In Section 6, we present a computational experiment. Finally, Section 7 reports our conclusions.

2. Perilimanaries

Consider a connected undirected graph $G = (V, E)$, where V and E are the sets of vertices and edges, respectively. Each vertex $v \in V$ has a positive weight $w(v)$, and a positive length $l(e)$ is associated with every edge $e \in E$. The shortest path distance between vertices x and y is denoted by $d(x, y)$. Let $U \subseteq V$ be a set of vertices. The distance between this set and $v \in V$ is defined as follows:

$$d(v, U) = \min_{u \in U}\{d(v, u)\}. \tag{1}$$

Suppose that

$$r(U) = \max_{v \in V}\{w(v)\ d(v, U)\}. \tag{2}$$

U^* is a vertex k-center of G if

$$r(U^*) = \min\{r(U), \quad U \subseteq V, |U| = k\}. \tag{3}$$

$r(U^*)$ or simply r_k is called the k-radius of G. If U and consequently U^* contain some points on the edges of G, then U^* is called an absolute k-center, and the corresponding radius is called an absolute k-radius [8]. An example of a 2-center of a weighted tree is illustrated in Figure 1, where the 2-radius is 40.

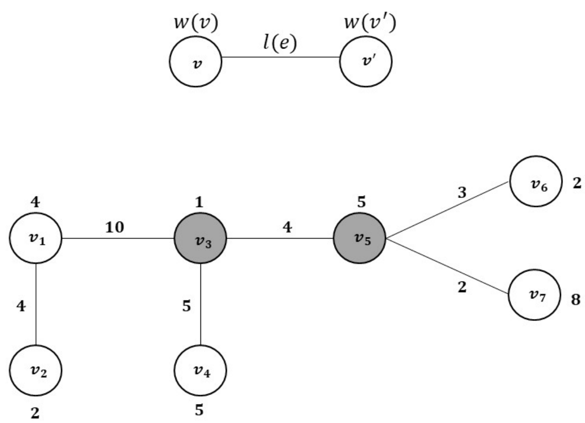


Figure 1: 2-center of a tree T.

Kariv and Hakimi [8] proved that the k-center problem on general graphs is an NP-hard problem. However, they presented some polynomial time algorithms for trees. Specifically, they introduced an $O(n^2 \log n)$ time algorithm for the k-center problem on weighted trees, using a linear time-dominated set subroutine.

3. Problem formulation

Consider a tree $T_0 = (V, E)$ of order n with vertex set V and edge set E. We define an urban region associated with every vertex and the edges in E denote the links between these regions. Each vertex $v \in V$ has a positive weight $w(v)$, and a positive length $l(e)$ is associated with each edge $e = v_s v_r \in E$. Suppose that some links in this network are destroyed, either due to human errors or a natural disaster, resulting in a damaged network that has now turned into a forest. Our goal is to determine an appropriate order for repairing broken links over a period of time while portable temporary service centers are located to serve people in different components of this forest. Therefore, E is partitioned into two sets: the set of existing edges E' and the set of potential edges E_t^d, which are the destroyed links at time t that need to be repaired. The existing edges E' and the vertices V in this damaged network form a forest $G = (V, E')$ that includes components $T_i = (V_i, E_i)$ for $i = 1, 2, ..., k$, where V_i and E_i are the vertex set and edge set, respectively. E_t^d at time t is represented as follows:

$$E_t^d = \{e = v_i v_j \mid e \in E,\ v_i \in V_i,\ v_j \in V_j,\ 1 \leq i \neq j \leq k\}, \qquad t \in \{1, 2, \cdots, T\}. \tag{4}$$

Clearly, $|E_1^d| = k - 1$. Assume that one damaged link is reconstructed in each time period. Therefore, $|E_t^d| = |E_{t-1}^d| - 1$. Henceforth, we shall refer to two components as potentially adjacent if there exists a potential edge between them. Suppose that k ($k > 2$) portable temporary service centers will remain in the network until the end of the network reconstruction, the locations of these centers can be changed during the reconstruction process. The disaster management team aims to locate k temporary emergency service centers in G such that every component has one center before the damaged links are reconstructed. We introduce a time horizon T such that $T = k - 1$. The goal is to transform G into a tree by adding potential edges in it over the time horizon T while ensuring that people are served fairly with emergency and mobile temporary service centers over T. One potential edge should be added to the network in each time period, and the length of the time horizon ensures that all potential edges will be added to the network. The goal of the problem is to minimize an aggregate value referred to as the *total k-radius* over the time horizon by selecting the appropriate arrangement to add potential edges. We refer to this problem as Incremental Reconstruction on Trees (IRT).

Let G^t denotes the unconnected network at time t, where $t < T$. Clearly, G^T is a connected tree. We make the assumption that before reconstructing the network, each component of G^t contains 1-center, and upon the reconstruction of a link, the number of centers in the new connected component is equal to the sum of the centers of its constituent components. For $i = 1, 2, \cdots, k - t + 1$, we denote the ith component and its number of portable centers at time t by T_i^t and q_{it}, respectively. $V(T_i^t)$ corresponds to the vertices of T_i^t at time t. It is important to note that G^t has $k - t + 1$ components. Without affecting the overall argument, we index the components from 1 to $k-t+1$ in each time period, as the indexing of components does not affect the solution of the problem.

The *total k-center* is calculated as follows:

$$R_k = \bar{r}_k^1 + \cdots + \bar{r}_k^T, \tag{5}$$

where $\bar{r}_k^t$ is the *major covering radius* during time period t and we define it as follows:

$$\bar{r}_k^t = \sum_{i=1}^{k-t+1} r_i^t, \tag{6}$$

in which r_i^t for $i = 1, 2, ..., k-t+1$ is the q_{it}-radius of the ith component at time t. The *major covering radius* in the last period of time equals the k-radius of T_0. When a potential edge is selected in each period of time, a new connected component is created and indexed as follows. Suppose that $e \in E_t^d$ is selected at time t and it connects T_i^t and T_j^t such that $i < j$. We index the new connected component as T_i^{t+1}, and we rename the components indexed T_s^t for $s = j+1, j+2, \cdots, k-t+1$ as T_{s-1}^{t+1}.

As previously mentioned, the order in which potential edges are added to the network is of great importance. In particular, the number of possible orders can be very large. Specifically, if the given forest has k components, then $(k-1)!$ orders must be considered in order to find the optimal one. To illustrate this point, consider the damaged network depicted in Figure 2 which resulted from a disaster, with vertex and edge weights assigned. In this network, there are four potential edges, namely e_1, e_2, e_3, and e_4, and five connected components. There are (5-1)!=24 possible orders, in which the potential edges can be added to the network with different total 5-radius values. The order (e_1, e_4, e_3, e_2) with a *total 5-radius* value of 26960 is optimal. All feasible solutions to this problem are presented in Table 1, while Table 2 illustrates the sequence of centers in all periods of time for the optimal solution. Furthermore, Figure 3 shows how to index components of the forest in Figure 2, during periods of time for the optimal solution.

In the following section, we will show that the *IRT* problem is solvable using a greedy algorithm and present a heuristic to solve it.

4. Algorithms

The first methodology utilized, employs a greedy algorithm that is presented in the following.

4.1 A greedy algorithm

In our proposed greedy algorithm, at time period t, a potential edge is selected that the sum of q_{it+1}-radii of the components T_i^{t+1} is minimized. By employing

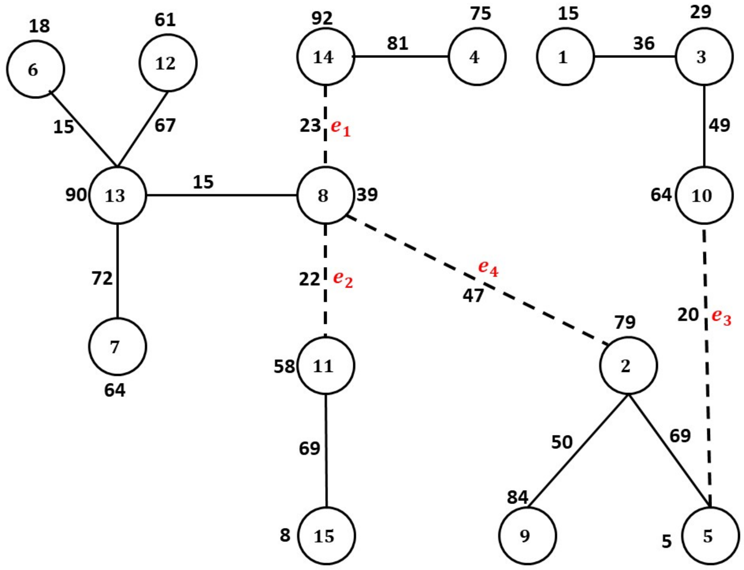


Figure 2: A damaged network G.

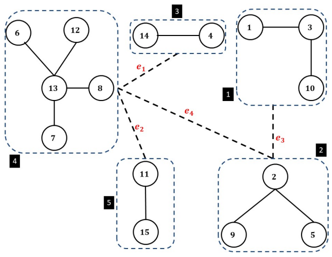
Table 1: All possible permutations of the problem for the damaged network in Figure 2. The optimal solution is highlighted in bold.

State	Order	R_5	State	Order	R_5
1	(e_1, e_2, e_3, e_4)	32540	13	(e_3, e_1, e_2, e_4)	36793
2	(e_1, e_2, e_4, e_3)	30011	14	(e_3, e_1, e_4, e_2)	33875
3	(e_1, e_3, e_2, e_4)	32407	15	(e_3, e_2, e_1, e_4)	42048
4	(e_1, e_3, e_4, e_2)	29489	16	(e_3, e_2, e_4, e_1)	44233
5	(e_1, e_4, e_2, e_3)	27349	17	(e_3, e_4, e_1, e_2)	35732
6	$(\mathbf{e_1, e_4, e_3, e_2})$	**26960**	18	(e_3, e_4, e_2, e_1)	40835
7	(e_2, e_1, e_3, e_4)	37795	19	(e_4, e_1, e_2, e_3)	29206
8	(e_2, e_1, e_4, e_3)	35266	20	(e_4, e_1, e_3, e_2)	28817
9	(e_2, e_3, e_1, e_4)	42917	21	(e_4, e_2, e_1, e_3)	34053
10	(e_2, e_3, e_4, e_1)	45102	22	(e_4, e_2, e_3, e_1)	38767
11	(e_2, e_4, e_1, e_3)	37451	23	(e_4, e_3, e_1, e_2)	33203
12	(e_2, e_4, e_3, e_1)	42165	24	(e_4, e_3, e_2, e_1)	38306

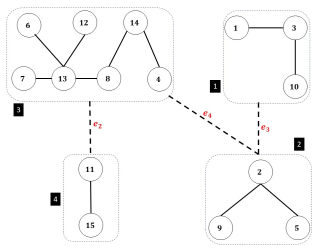
this methodology, we aim to minimize the *major covering radius* within each time period. In the steps of this algorithm, q_{it}-radii of components T_i^t are stored to avoid repetitive computations in next periods of time. The details of this method

Table 2: The places of facilities in all periods of time for the optimal solution of the damaged network in Figure 2.

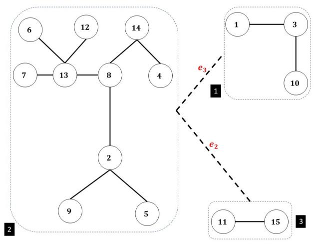
Period	The locations of the facilities
1	{9,10,11,13,14}
2	{4,9,10,11,13}
3	{2,4,10,11,13}
4	{2,4,5,11,13}
5	{4,5,7,9,13}



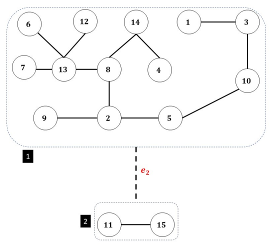
(a) Components in $t = 1$



(b) Components in $t = 2$



(c) Components in $t = 3$



(d) Components in $t = 4$

Figure 3: Representation indexing of components of damaged network shown in Figure 2 in periods $t = 1$ to $t = 4$.

have been presented in Algorithm 1.

Algorithm 1 The greedy algorithm

Input: A forest G with k components $T_i^1 = (V(T_i^1), E_i)$ is given. r_i^1 is the q_{i1}-radius of each T_i^1 ($q_{i1} = 1$ and $i = 1, 2, ..., k$). E_1^d ($|E_1^d| = k-1$) and E' are the set of potential edges and exisiting edges of G, respectively. $Inf_1 = \{T_i\left(V(T_i^1), q_{i1}\right),\quad i = 1, 2, ..., k\}$ and $Inf_2 = \{T_i\left(V(T_i^1), q_{i1}, r_i^1\right),\quad i = 1, 2, ..., k\}$ are the information about components and their radii. $R_k = 0$, $Ord = \{\}$.

for $t \in \{1, 2, ..., k-1\}$
    for $e = v_i v_j \in E_t^d$, $(i < j)$
        If $T_i\left((V(T_i^t) \cup V(T_j^t)), q_{it} + q_{jt}\right) \in Inf_1$,
            Choose $T_i\left((V(T_i^t) \cup V(T_j^t)), q_{it} + q_{jt}, \beta\right)$ from Inf_2,
            $r_m^t = \beta$.
        else
            $\alpha = q_{it} + q_{jt}$.
            Compute α - rdius of the connected tree T_i^t with vertices $V(T_i^t) \cup V(T_j^t)$.
            $Inf_1 = Inf_1 \cup \{T_i\left((V(T_i^t) \cup V(T_j^t)), \alpha\right)\}$.
            $Inf_2 = Inf_2 \cup \{T_i\left((V(T_i^t) \cup V(T_j^t)), \alpha, \alpha - radius\right)\}$.
            $r_m^t =$ The computed α-radius.
        $\overline{r}_k^t(e) = \left(\sum_{\substack{s=1 \\ s \neq i,j}}^{k-t+1} r_s^t\right) + r_m^t$.
        $T_i^t(e) =$ The connected tree with vertices $V(T_i^t) \cup V(T_j^t)$.
        $q_{it}(e) = q_{it} + q_{jt}$.
        $r(e) = r_m^t$.
    $\overline{r}_k^t(e') = \min_{e \in E_t^d}\{\overline{r}_k^t(e)\},\quad e' = v_i v_j,\quad i < j$.
    $E' = E' \cup \{e'\}$.
    $E_{t+1}^d = E_t^d - \{e'\}$.
    $r_i^{t+1} = r(e')$.
    $r_s^{t+1} = r_s^t,\quad s \neq i, j$.
    for $s \in \{1, 2, ..., i-1, i+1, ..., j-1\}$
        $T_s^{t+1} = T_s^t$.
        $V(T_s^{t+1}) = V(T_s^t)$.
    for $s \in \{j, j+1, ..., k-t+1\}$
        $T_s^{t+1} = T_{s-1}^t$.
        $V(T_s^{t+1}) = V(T_{s-1}^t)$.
    $T_i^{t+1} = T_i^t(e')$.
    $V(T_i^{t+1}) = V(T_i^t(e'))$.
    for $s \in \{1, 2, ..., k-t+1\}$ & $s \neq i, j$
        $q_{st+1} = q_{st}$.
    $q_{it+1} = q_{it}(e')$.
    $R_k = R_k + \overline{r}_k^t(e')$.
    $Ord = Ord \cup \{e'\}$.

Output: Ord and R_k.

In the following, we solve the problem by using this greedy method for the damaged network illustrated in Figure 2. The procedure involves entering potential edges into the network using this method over a period of time from $t = 1$ to $t = 4$. Tables 3 to 6 present the process of edge selection during this time period, with bolded rows indicating the selected edges. The *total* 5-*radius* value is 26960, as illustrated in Table 1.

Table 3: Greedy method process for the damaged network shown in Figure 2 at time $t = 1$.

Edge	q_{i2}-radii	Major radius	The locations of the facilities
$\mathbf{e_1}$	$\mathbf{r_1^1 = 1421}$ $\mathbf{r_2^1 = 3950}$ $\mathbf{r_3^1 = 4876}$ $\mathbf{r_5^1 = 552}$	$\mathbf{\bar{r}_5^1 = 10799}$	$\{\mathbf{4, 9, 10, 11, 13}\}$
e_2	$r_1^1 = 1421$ $r_2^1 = 3950$ $r_3^1 = 6075$ $r_4^1 = 4608$	$\bar{r}_5^1 = 16054$	$\{8, 9, 10, 13, 14\}$
e_3	$r_1^1 = 3950$ $r_3^1 = 6075$ $r_4^1 = 4608$ $r_5^1 = 552$	$\bar{r}_5^1 = 15185$	$\{5, 9, 11, 13, 14\}$
e_4	$r_1^1 = 1421$ $r_2^1 = 4608$ $r_3^1 = 6075$ $r_5^1 = 552$	$\bar{r}_5^1 = 12656$	$\{2, 10, 11, 13, 14\}$

Table 4: Greedy method process for the damaged network shown in Figure 2 at time $t = 2$.

Edge	q_{i3}-radii	Major radius	The locations of the facilities
e_2	$r_1^2 = 1421$ $r_2^2 = 3950$ $r_3^2 = 4140$	$\bar{r}_5^2 = 9511$	$\{4, 7, 9, 10, 13\}$
e_3	$r_1^2 = 3950$ $r_3^2 = 4876$ $r_4^2 = 552$	$\bar{r}_5^2 = 9378$	$\{4, 5, 9, 11, 13\}$
$\mathbf{e_4}$	$\mathbf{r_1^2 = 1421}$ $\mathbf{r_2^2 = 4876}$ $\mathbf{r_4^2 = 552}$	$\mathbf{\bar{r}_5^2 = 6849}$	$\{\mathbf{2, 4, 10, 11, 13}\}$

The presented problem involves a given forest with k components, which yields

Table 5: Greedy method process for the damaged network shown in Figure 2 at time $t = 3$.

Edge	q_{i4}-radii	Major radius	The locations of the facilities
e_2	$r_1^3 = 1421$ $r_2^3 = 4140$	$\bar{r}_5^3 = 5561$	$\{4, 7, 9, 10, 13\}$
$\mathbf{e_3}$	$\mathbf{r_1^3 = 4620}$ $\mathbf{r_3^3 = 552}$	$\mathbf{\bar{r}_5^3 = 5172}$	$\{\mathbf{2, 4, 5, 11, 13}\}$

Table 6: Greedy method process for the damaged network shown in Figure 2 at time $t = 4$.

Edge	q_{i5}-radius	Major radius	The locations of the facilities
$\mathbf{e_2}$	$\mathbf{r_1^4 = 4140}$	$\mathbf{\bar{r}_5^4 = 4140}$	$\{\mathbf{4, 5, 7, 9, 13}\}$

$(k-1)!$ feasible solutions. Solving large size problems by this method is time consuming. Therefore, we present a heuristic method, namely maximum radius method, to solve this problem for larger sizes. Before explicating this approach, we will state a proposition in the following.

Recall that in the IRT problem, the total radius is calculated by summing up the major covering radii over a specified time horizon. Each *major covering radius* in a given time period is obtained by updating the *major covering radius* from the previous time period using the formula provided in the following proposition.

Proposition 4.1. *If we choose $e = v_iv_j \in E_t^d$ at time t to establish a connection between T_i^t and T_j^t (where $i < j$), the major covering radius at time $t+1$ is*

$$\bar{r}_k^{t+1} = \bar{r}_k^t - (r_i^t + r_j^t) + r_i^{t+1}. \tag{7}$$

4.2 The maximum radius method

Now, we present a proposed heuristic, referred to as the maximum radius method. By considering Equation (7), the key idea for selecting a damaged link to repair at time t, is to choose the one connects two components T_i^t and T_j^t with the maximum value of $r_i^t + r_j^t$. This approach aims to minimize the *major covering radius* in each time period to the greatest extent possible. For each t, we define:

$$M = \mathrm{argmax}_{i,j}\{r_i^t + r_j^t \mid \text{The potential edge between } T_i^t \text{ and } T_j^t \text{ is selected}, \\ 1 \leq i < j \leq k - t + 1\}. \tag{8}$$

If M in (8) is unique, for example $M = r_a^t + r_b^t$, then we choose the potential edge that connects T_a^t and T_b^t at time t.

Now consider a period in which M in (8) is not unique. We define:

$$W = \max \left\{ \sum_{v_s \in V(T_a^t) \cup V(T_b^t)} w(v_s), \qquad M = r_a^t + r_b^t \right\}. \tag{9}$$

If there exists only one pair of potentially adjacent components in which the sum of their vertex weights is equal to W, they will be connected at this period. Otherwise, among such potentially adjacent components whose sum of vertex weights equals W, two potentially adjacent components are randomly selected to be connected. By doing so, the components of the supposed forest are connected, and a tree is eventually formed. The order in which the components are connected determines the order of reconstruction of the damaged links in the network. This procedure is represented in Algorithm 2. To provide further clarification of the procedure described above, we refer to the damaged network depicted in Figure 2. The steps for using the maximum radius method is summarized in Table 7. Notably, in this particular instance, the heuristic approach yields the exact solution.

Table 7: The numerical result of selecting potential edges of the damaged network shown in Figure 2 by the maximum radius method.

t	q_{it}-radii	The major covering radius	M	The selected edge
1	r_1^1=1421 r_2^1=3950 r_3^1=6075 r_4^1=4608 r_5^1=552	$\overline{r}_5^1$=16606	$r_3^1 + r_4^1 = 10683$	e_1
2	r_1^2=1421 r_2^2=3950 r_3^2=4876 r_4^2=552	$\overline{r}_5^2$=10799	$r_2^2 + r_3^2 = 8826$	e_4
3	r_1^3=1421 r_2^3=4876 r_3^3=552	$\overline{r}_5^3$=6849	$r_1^3 + r_2^3 = 192$	e_3
4	r_1^4=4620 r_2^4=552	$\overline{r}_5^4$=5172	$r_1^4 + r_2^4 = 5172$	e_2

Algorithm 2 The maximum radius algorithm

Input: A forest G with k components $T_i^1 = (V(T_i^1), E_i)$ is given. r_i^1 is the q_{i1}-radius of each T_i^1 ($q_{i1} = 1$ and $i = 1, 2, ..., k$). E_1^d ($|E_1^d| = k - 1$) and E' are the set of potential edges and exisiting edges of G, respectively. $R_k = 0$, $Ord = \{\}$.

for $t \in \{1, 2, ..., k - 1\}$
  $Set = \{\}$
  for $e = v_i v_j \in E_t^d$, $(i < j)$
    $Set = Set \cup \{r_i^t + r_j^t\}$.
  $Max_1 = \max(Set)$.
  $M_1 = \{(i, j) \mid r_i^t + r_j^t = Max_1, \quad e = v_i v_j \in E_t^d\}$.
  if $|M_1| \neq 1$
    $W = \{\}$.
    for $(a, b) \in M_1$
      $W(a, b) = \displaystyle\sum_{v_s \in V(T_a^t) \cup V(T_b^t)} w(v_s)$.
      $W = W \cup \{W(a, b)\}$.
    $Max_2 = \max(W)$.
    $M_2 = \{(i, j) \in W \mid W(i, j) = Max_2\}$.
    if $|M_2| = 1$
      Choose the unique element (a, b) of M_2.
      $e' = v_a v_b$.
    else
      Choose an element (a, b) of M_2 randomly.
      $e' = v_a v_b$.
  else
    Choose the unique element (a, b) of M_1.
    $e' = v_a v_b$.
  $E' = E' \cup \{e'\}$.
  $E_{t+1}^d = E_t^d - \{e'\}$.
  $r_a^{t+1} = (q_{at} + q_{bt})$-radius the new connected component which is named by $T_a^t(e')$.
  $\overline{r}_k^t(e') = \left(\sum_{\substack{s=1 \\ s \neq a,b}}^{k-t+1} r_s^t\right) + r_a^{t+1}$.
  $r_i^{t+1} = r_i^t, \quad i \neq a, b$.
  for $s \in \{1, 2, ...a - 1, a + 1, ..., b - 1\}$
    $T_s^{t+1} = T_s^t$.
    $V(T_s^{t+1}) = V(T_s^t)$.
  for $s \in \{b + 1, b + 2, ..., k - t + 1\}$
    $T_s^{t+1} = T_{s-1}^t$.
    $V(T_s^{t+1}) = V(T_{s-1}^t)$.
  $T_a^{t+1} = T_a^t(e')$.
  $V(T_a^{t+1}) = V(T_a^t(e'))$.
  $q_{at+1} = q_{at} + q_{bt}$.
  $q_{st+1} = q_{st}, \quad s \neq a, b$.
  $R_k = R_k + \overline{r}_k^t(e')$.
  $Ord = Ord \cup \{e'\}$.

Output: Ord and R_k.

5. Determining the changed location of temporary emergency centers

As the network is beeing expanded, some facilities should be relocated for making the system more efficient. However, the relocation of temporary service centers incurs costs. For instance, changing the location of ambulances involves costs such as fuel and time consumption. Similarly, relocating temporary fueling centers incurs costs of re-establishing and installing these places. Thus we address a new type of problem, where the costs of re-establishing service centers are taken into consideration in addition to service costs.

Suppose that $A_{t-1} = \{v_1^{t-1}, v_2^{t-1}, ..., v_s^{t-1}\}$ represents a set of locations in two components at time $t-1$, where the potential edge between them has been selected at time t, according to our methods. Let $A_t = \{v_1^t, v_2^t, ..., v_s^t\}$ be the set of changed locations in the new connected component at time t. Each relocation of a facility for moving from a location v_i^{t-1} to a location v_j^t incurs a cost, say c_{ij}. Obviously, if $v_i^{t-1} = v_j^t$, then $c_{ij} = 0$. The relocation cost between vertices is given in a matrix referred to as C. We define binary variable x_{ij}^t at time t as follows:

$$x_{ij}^t = \begin{cases} 1, & \text{If the facility } v_i^{t-1} \text{ is relocated at } v_j^t \text{ at time } t, \quad v_i^{t-1} \neq v_j^t, \\ 0, & \text{o.w.} \end{cases}$$

In order to determine the changed locations of facilities with minimum cost, for each $t \in \{2, 3, ..., T\}$,

$$\min \quad \sum_{i:v_i^{t-1} \in A_{t-1}} \sum_{j:v_j^t \in A_t} c_{ij} x_{ij}^t, \tag{10}$$

$$s.t \quad \sum_{i:v_i^{t-1} \in A_{t-1}} x_{ij}^t = 1, \qquad \forall\, j : v_j^t \in A_t, \tag{11}$$

$$\sum_{j:v_j^t \in A_t} x_{ij}^t = 1, \qquad \forall\, i : v_i^{t-1} \in A_{t-1}, \tag{12}$$

$$x_{ij}^t \in \{0, 1\}, \quad \forall i \text{ and } \forall j. \tag{13}$$

The objective function (10) is to minimize the total cost incurred in the assignment of vertices from set A_{t-1} at time $t-1$ to vertices in set A_t at time t. The constraint (11) indicates that each vertex $v_j^t \in A_t$ at time t is assined to just one vertex v_i^{t-1}. Similarly, the constraint (12) shows that every vertex $v_i^{t-1} \in A_{t-1}$ is assined to just one vertex at time t. The decision variables x_{ij}^t are binary, taking values in $\{0, 1\}$, indicating whether vertex v_i^{t-1} is assigned to vertex v_j^t or not.

Clearly, after adding each potential edge, we may relocate some facilities. Therefore, for this purpose, we consider the above assignment model, immediately after adding each potential edge.

6. Computational experiments

To evaluate the performance of the proposed methods, numerical results are presented in two sub-sections: for graphs with small, and large sizes. In our study, the size of the graph is dependent on the number of potential edges it has. As the number of potential edges in a graph increases, the computational time also increases. For small-sized graphs, two proposed algorithms have been compared to an exact method based on two criteria: the value of the objective function, and the algorithm's execution time. In this section, the exact method we have used involves bruteforce enumerating all feasible solutions, and presenting the best one as the optimal solution. Subsequently, for large-sized, the heuristic method is compared to the greedy method, and its performance is examined based on the value of the objective function and the execution time of the algorithm. As the IRT problem is a new addition to the literature, we faced limitations in testing the model with real-world data or utilizing previous references as benchmarks. To overcome this, we generate distinct random forests with vertices and potential edges using Python. Our model was implemented using Python 3.10 on a system consisting of Intel(R), Core(TM) i7-3770 CPU running at 3.40 GHz with 8.00 GB RAM under the windows 10 operating system (64-bit). To verify this experiment, we test our model on some randomly generated samples in 50 iterations.

6.1 Instances with small size

In this sub-section, to evaluate the performance of the methods based on the objective function value, we calculate the deviation percentage between the objective function values of the mentioned methods and the exact method on instance I using the following equation. Here, O^I_{Meth} and O^I_{Ex} refer to the objective function values for the instance I generated by the mentioned methods and the exact method, respectively.

$$\Delta^I_{Meth} = \frac{(O^I_{Meth} - O^I_{Ex})}{O^I_{Ex}} \times 100\%. \tag{14}$$

We used three features to evaluate the performance of the mentioned methods, which are explained below.

- $\%OPT_{Meth}$: The percentage of instances for which a method attains the exact solution, i.e., $\Delta^I_{Meth} = 0$.
- $\overline{\Delta}_{Meth}$: The average of deviation percentage values i.e., $\frac{1}{50}\sum_{I=1}^{50} \Delta^I_{Meth}$.
- max Δ_{Meth}: The maximum of deviation percentage values i.e., $\max_I \Delta^I_{Meth}$.

To check the performance of methods in terms of the execution time, we employ Equation (15), which denotes the acceleration rate of the mentioned methods relative to the exact method. Here, T_{Ex} and T_{Meth} denote the CPU time required by the exact method and the mentioned methods, respectively.

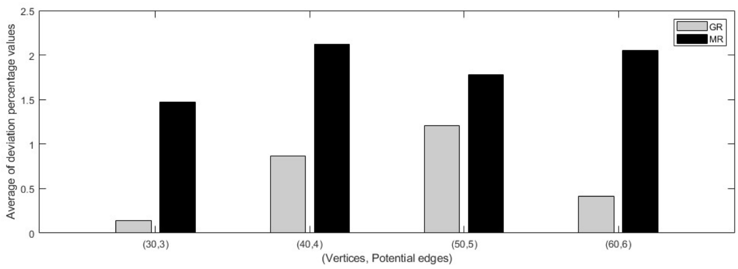
$$\lambda_{Meth} = \frac{T_{Ex}}{T_{Meth}}. \tag{15}$$

We shall employ the notations GR and MR instead of the symbol $Meth$, for representing the greedy and maximum radius methods, respectively.

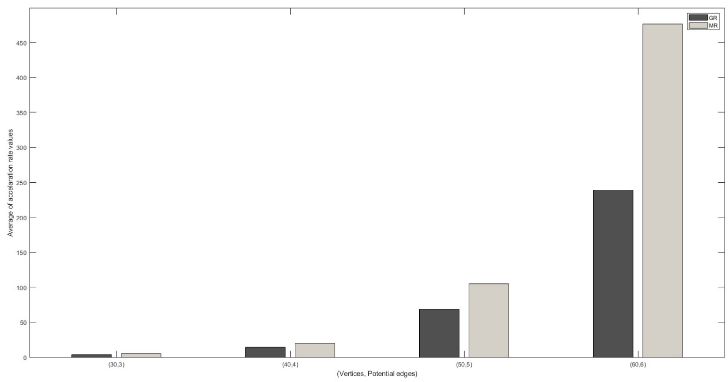
We consider graphs with 3 to 6 potential edges as small-sized instances. Solving precisely the samples with more than 6 potential edges requires an extraordinarily long time, and these samples are considered as large-sized samples. Our experiments have been conducted on three categories: unweighted type I, unweighted type II and weighted graphs. Unweighted type I graphs are samples in which the weight of all vertices are considered one, and the weight of edges is a positive number. However, Unweighted type II graphs are samples in which the weight of all vertices and edges are equal to one. Here, the random samples are graphs whose weights have been randomly, and uniformly chosen between 1 and 100. The results related to small-sized samples have been presented in Tables 8 to 10. The pairs within column I in all tables indicate the number of vertices, and potential edges for each sample, respectively. δ_{GR} and δ_{MR} represent the standard deviations in these instances.

As shown in Tables 8 to 10, the optimality percentage of the greedy method is significantly higher compared to the maximum radius method, and in most cases, it has achieved the exact solution in all categories. By considering $\overline{\Delta}_{Meth}$ columns in these two methods, it has been observed that this value is lower in the greedy method compared to the heuristic method in all instances. The greedy method has achieved a more satisfactory performance in terms of the maximum percentage deviation values in most instances compared to the maximum radius method. However, in three instances, the maximum radius method yields better outcomes. By comparing the acceleration rates column of these two methods in Tables 8 to 10, it is evident that the maximum radius method performs the fastest in small-sized samples. Analyzing these data reveals that these two methods have significantly higher speeds compared to the exact method.

The average of percentage deviation values and accelerations rate for unweighted and weighted instances with small size over 50 iterations are depicted in Figures 4 to 6. In all figures, GR and MR denote results of the greedy and maximum radius methods, respectively. These figures clearly demonstrate the superiority of the average of deviation percentage values of the greedy method compared to the heuristic method, and the high speed of two proposed methods is evident compared to the exact method. It is evident that as the graph size increases, the speed of the methods significantly increases.



(a) Average of deviation percentage values



(b) Average of accelaration rates

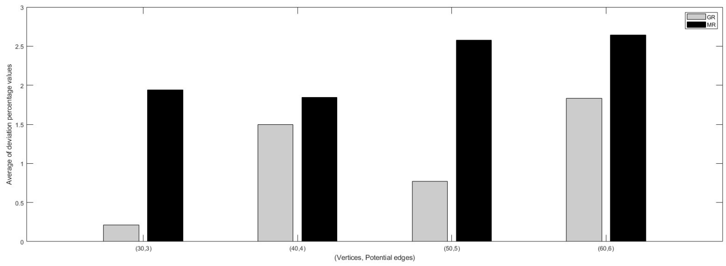
Figure 4: Results of small-sized unweighted (type I) instances.

6.2 Instances with large size

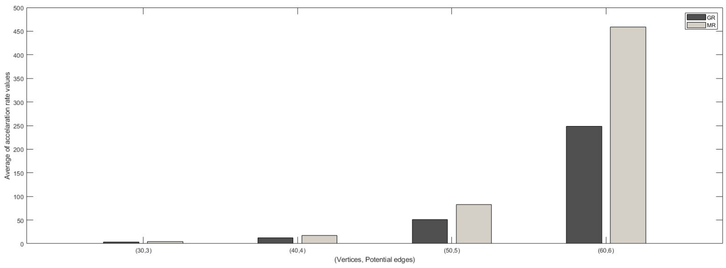
The high percentage of optimality and the low average deviation percentage values in the greedy method for small-sized graphs, as well as the impracticality of exactly solving the problem for large-sized graphs, compel us to compare the heuristic method in terms of its performance of objective value and algorithm execution time with the greedy method, in this sub-section. In evaluating the efficiency of the objective function, we calculate a relative distance on instance I according to the following formula. Here O^I_{MR} and O^I_{GR} refer to the objective function value of the maximum radius method and the greedy method, on instance I, respectively.

$$Rel^I_{MR} = \frac{(O^I_{MR} - O^I_{GR})}{O^I_{GR}} \times 100\%. \tag{16}$$

Similar to the previous sub-section, we utilize three following features in evaluating the efficiency of the objective function.



(a) Average of deviation percentage values



(b) Average of accelaration rates

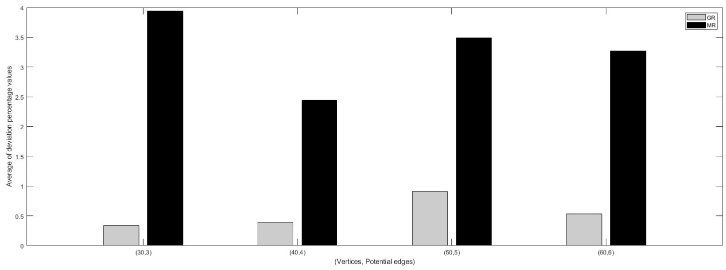
Figure 5: Results of small-sized unweighted (type II) instances.

- $\%GR/MR$: The percentage of instances for which the heuristic method attains the solution of greedy method, i.e., $Rel^I_{MR} = 0$.
- $\overline{Rel}_{MR}$: The average of relative distance values i.e., $\frac{1}{50}\sum_{I=1}^{50} Rel^I_{Meth}$.
- $\max\ Rel_{MR}$: The maximum of relative distance values i.e., $\max_I Rel^I_{MR}$.

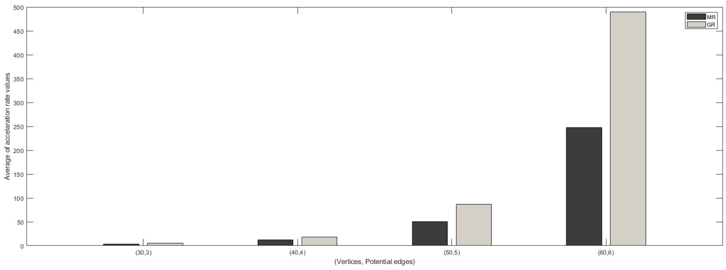
In evaluating the efficiency of methods in terms of the execution time, we use the same equation (15), where we compare the heuristic method with the greedy method i.e., we compute $\lambda'_{MR} = \frac{T_{GR}}{T_{MR}}$, which T_{GR} and T_{MR} denote the CPU time required by the greedy method and the mentioned heuristic method, respectively.

Here, we have considered graphs with potential edges of 7 to 20 as large-scale samples. Tables 11 to 13 represent large-scale instances results.

As observed in Tables 11 to 13, the maximum radius method yields the solution of greedy method more, in unweighted type II instances related to two other categories. The results denote that the average relative distance values in the maximum radius method is atmost 2.2209 within these three categories. Additionally, the accelaration rates of the heuristic method increase with the size of instances. The low relative distance and high accelaration rate of the heuristic



(a) Average of deviation percentage values



(b) Average of accelaration rates

Figure 6: Results of small-sized weighted instances.

method relative to the greedy method underscore the efficiency of the maximum radius method.

7. Conclusions

This study addresses the problem of determining the optimal order of reconstruction for links that have been destroyed due to human errors or natural disasters in a tree network. To solve the problem, we propose a greedy method and a heuristic method: the maximum radius method. To evaluate the efficiency of the proposed methods, numerical computations have been performed on randomly generated graphs. Numerical results are presented in two sub-sections for small-sized, and large-sized instances. The results obtained from the first section allow us to compare the heuristic with the greedy method. The numerical results indicate that the greedy method is a suitable approach for solving the IRT problem. However, as the dimension of the problem increases, the accelaration rate of the heuristic algorithm improves compared to the greedy method, while the relative distance between them does not show significant growth. Therefore, in large dimensions, the proposed heuristic method is recommended for problem solving.

Table 8: Comparison of the proposed methods with the exact method for small-sized unweighted (type I) instances.

I	$\%opt^I_{GR}$	$\overline{\Delta}^I_{GR}$	$\max \Delta^I_{GR}$	$\overline{\lambda}^I_{GR}$	δ_{GR}	$\%opt^I_{MR}$	$\overline{\Delta}^I_{MR}$	$\max \Delta^I_{MR}$	$\overline{\lambda}^I_{MR}$	δ_{MR}
(30,3)	96	0.1421	6.4945	3.6916	0.9115	50	1.4703	14.0930	4.9452	2.5714
(40,4)	78	0.8666	9.1481	14.3691	2.2816	42	2.1243	8.1198	19.6701	2.5388
(50,5)	68	1.2076	13.8068	69.0512	2.6106	26	1.7786	13.8441	105.1187	2.5113
(60,6)	84	0.4150	10.4423	239.2687	1.6218	24	2.0516	22.6331	476.1025	3.6424

Table 9: Comparison of the proposed methods with the exact method for small-sized unweighted (type II) instances.

I	$\%opt^I_{GR}$	$\overline{\Delta}^I_{GR}$	$\max \Delta^I_{GR}$	$\overline{\lambda}^I_{GR}$	δ_{GR}	$\%opt^I_{MR}$	$\overline{\Delta}^I_{MR}$	$\max \Delta^I_{MR}$	$\overline{\lambda}^I_{MR}$	δ_{MR}
(30,3)	96	0.2105	5.2632	3.8180	1.0314	68	1.9433	11.7647	4.8813	2.9618
(40,4)	76	1.4987	16.0000	12.4865	3.2890	62	1.8425	9.3750	17.3602	2.6613
(50,5)	82	0.7694	13.5135	51.1064	2.2125	48	2.5772	16.2162	82.4668	3.5228
(60,6)	60	1.8344	12.7273	248.2547	2.8800	34	2.6452	13.4615	459.3150	2.9010

Table 10: Comparison of the proposed methods with the exact method for small-sized weighted instances.

I	$\%opt^I_{GR}$	$\overline{\Delta}^I_{GR}$	$\max \Delta^I_{GR}$	$\overline{\lambda}^I_{GR}$	δ_{GR}	$\%opt^I_{MR}$	$\overline{\Delta}^I_{MR}$	$\max \Delta^I_{MR}$	$\overline{\lambda}^I_{MR}$	δ_{MR}
(30,3)	92	0.3356	7.5509	3.6054	1.2988	57	3.9404	37.4691	4.9995	7.0954
(40,4)	92	0.3914	11.0283	12.0818	1.7665	48	2.4444	17.0126	18.1598	3.9469
(50,5)	72	0.9103	8.8698	50.2514	2.0470	20	3.4884	20.1794	86.8427	3.9197
(60,6)	82	0.5345	8.3763	247.1256	1.5482	26	3.2698	14.8221	489.6613	4.1944

Conflicts of Interest. The authors declare that they have no conflicts of interest regarding the publication of this article.

Acknowledgments. We would like to express our sincere gratitude to the anonymous reviewers for their valuable comments and constructive feedback, which greatly contributed to the improvement of this paper.

Table 11: Comparison of the maximum radius method with the greedy method for large-sized unweighted (type I) instances.

I	$\%GR/MR^I$	$\overline{Rel}^I_{MR}$	$\max Rel^I_{MR}$	$\overline{\lambda}'^I_{MR}$	δ_{MR}
(70,7)	22	1.3209	7.5586	2.2408	1.9541
(80,8)	16	1.6661	5.7352	2.3991	1.6985
(90,9)	8	1.1110	11.6113	2.5580	2.7555
(100,10)	8	0.5988	3.6060	2.8786	1.9874
(110,11)	4	0.8865	5.5463	2.6762	2.1295
(120,12)	8	1.4374	8.9489	3.4181	2.4594
(130,13)	2	1.2795	8.5296	3.2689	2.4539
(140,14)	4	0.3174	5.9053	3.3609	2.1592
(150,15)	2	0.6598	6.4892	3.7447	1.6030
(160,16)	0	0.8127	6.0663	3.7448	1.7602
(170,17)	6	1.2688	17.6140	4.3048	3.0523
(180,18)	2	1.0460	9.4050	4.2288	2.2480
(190,19)	4	0.9894	5.6503	4.0540	1.9194
(200,20)	0	0.8317	7.7139	4.6576	2.2181

Table 12: Comparison of the maximum radius method with the greedy method for large-sized unweighted (type II) instances.

I	$\%GR/MR^I$	$\overline{Rel}^I_{MR}$	$\max Rel^I_{MR}$	$\overline{\lambda}'^I_{MR}$	δ_{MR}
(70,7)	22	1.5407	21.0526	2.0920	4.5317
(80,8)	22	0.4500	5.1020	2.3374	2.5332
(90,9)	26	0.5784	12.6126	2.4625	3.2577
(100,10)	16	1.4219	15.0000	2.6460	4.4931
(110,11)	18	0.9772	9.6970	2.8997	3.0103
(120,12)	14	0.7547	10.1694	3.0026	3.0369
(130,13)	14	0.9576	9.6939	3.2859	4.0111
(140,14)	12	0.7767	13.2401	3.4119	3.7896
(150,15)	10	0.4163	8.3650	3.8129	2.8675
(160,16)	10	1.7548	11.3879	4.0393	3.0802
(170,17)	14	1.0217	17.0659	4.3260	3.3828
(180,18)	4	1.3024	9.2857	4.2315	2.8043
(190,19)	6	0.6693	6.9061	4.4531	2.4329
(200,20)	4	0.2749	12.9545	4.0118	3.1028

Table 13: Comparison of the maximum radius method with the greedy method for large-sized weighted instances.

I	$\%GR/MR^I$	$\overline{Rel}^I_{MR}$	$\max Rel^I_{MR}$	$\overline{\lambda}'^I_{MR}$	δ_{MR}
(70,7)	4	2.2131	10.3244	2.1539	3.0187
(80,8)	20	1.4623	6.5767	2.3444	2.0855
(90,9)	6	1.8755	9.3386	2.4907	3.0683
(100,10)	2	1.7232	8.6549	2.6772	3.0577
(110,11)	2	1.5841	11.4609	2.8771	3.0528
(120,12)	4	2.2209	15.4808	3.1936	3.4153
(130,13)	2	2.1650	16.0001	3.3268	3.7928
(140,14)	2	1.5451	10.1637	3.5771	2.8568
(150,15)	2	1.2839	13.4785	3.6001	3.0460
(160,16)	0	1.8810	7.4444	3.9192	3.0361
(170,17)	0	1.6026	6.8355	4.0377	2.3810
(180,18)	0	1.9111	12.0054	4.4080	3.0075
(190,19)	0	1.7972	8.5497	4.6101	2.2824
(200,20)	0	1.1290	8.9509	4.3943	2.4456

References

[1] S. L. Hakimi, Optimum locations of switching centers and the absolute centers and medians of a graph, *Oper. Res.* **12** (1964) 450 – 459, https://doi.org/10.1287/opre.12.3.450.

[2] O. Kariv and S. L. Hakimi, An algorithmic approach to network location problems. II: The p-medians, *SIAM J. Appl. Math.* **37** (1979) 539 – 560, https://doi.org/10.1137/0137041.

[3] A. J. Goldman, Optimal center location in simple networks, *Transportation Sci.* **5** (1971) 212 – 221, https://doi.org/10.1287/trsc.5.2.212.

[4] M. S. Daskin and K. L. Maass, *The P-Median Problem*, Location science. Springer, Cham, 2015.

[5] C. Duran-Mateluna, Z. Ales and S. Elloumi, An efficient Benders decomposition for the p-median problem, *Eur. J. Oper. Res.* **308** (2023) 84 – 96, https://doi.org/10.1016/j.ejor.2022.11.033.

[6] C. Wang, C. Han, T. Guo and M. Ding, Solving uncapacitated P-Median problem with reinforcement learning assisted by graph attention networks, *Appl. Intell.* **53** (2023) 2010 – 2025, https://doi.org/10.1007/s10489-022-03453-z.

[7] N. Megiddo, Linear-time algorithms for linear programming in R^3 and related problems, *SIAM J. Comput.* **12** (1983) 759 – 776, https://doi.org/10.1137/0212052.

[8] O. Kariv and S. L. Hakimi, An algorithmic approach to network location problems. I: The p-centers, *SIAM J. Appl. Math.* **37** (1979) 513 – 538, https://doi.org/10.1137/0137040.

[9] N. Megiddo and A.Tamir, New results on the complexity of p-centre problems, *SIAM J. Comput.* **12** (1983) 751 – 758, https://doi.org/10.1137/0212051.

[10] M. Jeger and O. Kariv, Algorithms for finding P-centers on a weighted tree (for relatively small P), *Networks* **15** (1985) 381 – 389, https://doi.org/10.1002/net.3230150308.

[11] A. Banik, B. Bhattacharya, S. Das, T. Kameda and Z. Song, The p-center problem in tree networks revisited, *arXiv preprint arXiv 1604.07535* (2016).

[12] H. Wang and J. Zhang, An $O(n \log n)$-time algorithm for the k-center problem in trees, *SIAM J. Comput.* **50** (2021) 602 – 635, https://doi.org/10.1137/18M1196522.

[13] W. Wang, K. Yang, L. Yang and Z. Gao, Tractable approximations for the distributionally robust conditional vertex p-center problem: application to the location of high-speed railway emergency rescue stations, *J. Oper. Res. Soc.* **73** (2022) 525 – 539, https://doi.org/10.1080/01605682.2020.1843983.

[14] F. Kılcı, B. Y. Kara and B. Bozkaya, Locating temporary shelter areas after an earthquake: a case for Turkey, *Eur. J. Oper. Res.* **243** (2015) 323 – 332, https://doi.org/10.1016/j.ejor.2014.11.035.

[15] F. Cavdur, M. Kose-Kucuk and A. Sebatli, Allocation of temporary disaster response facilities under demand uncertainty: an earthquake case study, *Int. J. Disaster Risk Reduct.* **19** (2016) 159 – 166, https://doi.org/10.1016/j.ijdrr.2016.08.009.

[16] H. Li, L. Zhao, R. Huang and Q. Hu, Hierarchical earthquake shelter planning in urban areas: a case for Shanghai in China, *Int. J. Disaster Risk Reduct.* **22** (2017) 431 – 446, https://doi.org/10.1016/j.ijdrr.2017.01.007.

[17] E. Celik, A cause and effect relationship model for location of temporary shelters in disaster operations management, *Int. J. Disaster Risk Reduct.* **22** (2017) 257 – 268, https://doi.org/10.1016/j.ijdrr.2017.02.020.

[18] A. Trivedi, A multi-criteria decision approach based on DEMATEL to assess determinants of shelter site selection in disaster response, *Int. J. Disaster Risk Reduct.* **31** (2018) 722 – 728, https://doi.org/10.1016/j.ijdrr.2018.07.019.

[19] M. Drakaki, H. G. Gören and P. Tzionas, An intelligent multi-agent based decision support system for refugee settlement siting, *Int. J. Disaster Risk Reduct.* **31** (2018) 576 – 588, https://doi.org/10.1016/j.ijdrr.2018.06.013.

[20] M. K. Oksuz and S. I. Satoglu, A two-stage stochastic model for location planning of temporary medical centers for disaster response, *Int. J. Disaster Risk Reduct.* **44** (2020) p. 101426, https://doi.org/10.1016/j.ijdrr.2019.101426.

[21] M. Karatas and E. Yakıcı, A multi-objective location analytics model for temporary emergency service center location decisions in disasters, *Decis. Anal. J.* **1** (2021) p. 100004, https://doi.org/10.1016/j.dajour.2021.100004.

[22] A. M. Sharp, *Incremental Algorithms: Solving Problems in a Changing World*, Cornell University, 2007.

[23] J. Hartline, *Incremental Optimization*, Cornell University, 2007.

[24] J. Hartline and A. Sharp, *An Incremental Model for Combinatorial Maximization Problems*, International Workshop on Experimental and Efficient Algorithms, Springer, Berlin, Heidelberg, 2006.

[25] F. Della Croce, U. Pferschy and R. Scatamacchia, *Approximation Results for the Incremental Knapsack Problem*, International Workshop on Combinatorial Algorithms, Springer, Cham. 2018.

[26] F. Della Croce, U. Pferschy and R. Scatamacchia, On approximating the incremental knapsack problem, *Discrete Appl. Math.* **264** (2019) 26 – 42, https://doi.org/10.1016/j.dam.2019.02.016.

[27] D. Bienstock, J. Sethuraman and C. Ye, Approximation algorithms for the incremental knapsack problem via disjunctive programming, *arXiv preprint arXiv 1311.4563* (2013).

[28] Y. Faenza, D. Segev and L. Zhang, Approximation algorithms for the generalized incremental knapsack problem, *Math. Program.* **198** (2023) 27 – 83, https://doi.org/10.1007/s10107-021-01755-7.

[29] A. Aouad and D. Segev, An approximate dynamic programming approach to the incremental knapsack problem, *Oper. Res.* **71** (2023) 1414 – 1433, https://doi.org/10.1287/opre.2022.2268.

[30] K. Engel, T. Kalinowski and M. W. P. Savelsbergh, Incremental network design with minimum spanning trees, *J. Graph Algorithms Appl.* **21** (2017) 417 – 432, http://doi.org/10.7155/jgaa.00423.

[31] M. Baxter, T. Elgindy, A. T. Ernst, T. Kalinowski and M. W. P. Savelsbergh, Incremental network design with shortest paths, *European J. Oper. Res.* **238** (2014) 675 – 684, https://doi.org/10.1016/j.ejor.2014.04.018.

[32] T. Kalinowski, D. Matsypura and M. W. P. Savelsbergh, Incremental network design with maximum flows, *European J. Oper. Res.* **242** (2015) 51 – 62, https://doi.org/10.1016/j.ejor.2014.10.003.

[33] C. G. Plaxton, Approximation algorithms for hierarchical location problems, *J. Comput. System Sci.* **72** (2006) 425 – 443, https://doi.org/10.1016/j.jcss.2005.09.004.

[34] H. Du, Y. Xu and B. Zhu, An incremental version of the k-center problem on boundary of a convex polygon, *J. Comb. Optim* **30** (2015) 1219 – 1227, https://doi.org/10.1007/s10878-015-9933-3.

[35] A. Arulselvan, A. Bley and I. Ljubić, The incremental connected facility location problem, *Comput. Oper. Res.* **112** (2019) p. 104763, https://doi.org/10.1016/j.cor.2019.104763.

[36] C. Gokalp, P. N. Patil and S. D. Boyles, Post-disaster recovery sequencing strategy for road networks, *Transp. Res. B: Methodol.* **153** (2021) 228 – 245, https://doi.org/10.1016/j.trb.2021.09.007.

Sedigheh Nazari
Department of Mathematics,
Shahed University,
Tehran, I. R. Iran
e-mail: sedigheh_nazari@yahoo.com

Ardeshir Dolati
Department of Computer Sciences,
Shahed University,
Tehran, I. R. Iran
e-mail: dolati@shahed.ac.ir