MATEMATIČKI VESNIK
МАТЕМАТИЧКИ ВЕСНИК
76, 1–2 (2024), 66–83
March 2024

research paper
оригинални научни рад
DOI: 10.57016/MV-RBJS4362

AN EFFICIENT SOLUTION APPROACH TO THE p-NEXT CENTER PROBLEM

Jelena Tasić

Abstract. An extension of the p-center problem, called the p-next center problem, is considered in this paper. In practice, it has been shown that centers can close suddenly due to a problem (accident, staff shortage, technical problem, etc.). In this case, customers should proceed to the backup center - the one closest to the closed center. Both the p-center problem and the p-next center problem are NP-hard, so approximation methods are suitable for solving them. In this paper, an efficient solution approach based on Skewed Variable Neighborhood Search (SVNS) is proposed for the p-next center problem. The performance of the proposed SVNS method is evaluated on a set of pmed instances with up to 900 nodes. The obtained computational results are presented and compared with the best results from the literature, confirming the efficiency and stability of the proposed method in solving the p-next center problem.

1. Introduction

The standard p-center problem was first introduced in 1965 by Hakimi in [5]. It is one of the most studied problems in combinatorial optimization and applied mathematics in general, as it is widely used in practice. For a given set of n locations, the goal is to find exactly p locations ($p < n$) for centers and assign each user to its nearest center in such a way that the maximal distance from a user to its nearest center is minimal.

If there is no method to solve a certain problem in polynomial time, this problem is called an NP-hard problem. This basically means that these problems cannot be solved by any exact method in a reasonably short time. The main advantage of exact methods is that they provide optimal solutions, if such solutions exist. Despite the fact that combinatorial optimization problems usually have a finite number of feasible solutions, this number is often very large, especially for high-dimensional problems,

2020 Mathematics Subject Classification: 90C27, 90C59, 90B80
Keywords and phrases: Combinatorial optimization; p-next center problem; variable neighborhood search; fast interchange heuristic.

and exact methods are practically useless in this case. On the other hand, it is often not possible to apply exact methods because there is no suitable mathematical model to which they could be applied or the model is too complex. In this case, it is often desirable to obtain solutions of high quality, but in a short time. For all these reasons, approximate methods are not only frequently used to solve such difficult problems, but they are often the only possible ones.

The p-center problem is one of the most studied NP-hard combinatorial optimization problems [8] due to its important practical applications, e.g. in public services such as hospitals, police stations, fire stations, but also for deployment of taxi stations, parking lots, storage rooms, etc. Given its complexity, numerous approximation methods for its solving have been proposed in the literature [11]. In this paper, we consider an extension of the p-center problem, called the p-next center problem. Some unpredictable situations may occur forcing some centers to close. Natural disasters such as earthquakes, floods, forest fires, storms, etc. can lead to the collapse of a center. In addition, different types of problems can result with the same outcome, e.g. power failure, lack of staff, technical malfunctions, etc. It can be assumed that a user cannot know in advance whether a center is closed. If a user arrives at a closed center, they should proceed to the so-called backup center. This backup center should be a center that is on the shortest distance from the closed primary center. For example, one cannot know in advance whether an ATM is out of service. If this is the case upon his arrival, the user would go to the ATM closest to the closed one. The centers should be determined in such a way that the maximal distance of a user to the nearest backup center via the primary center is minimal. The p-next center problem under consideration is also NP-hard as an extension of the p-center problem. The described differences between p-center and p-next-center problems can be illustrated by the example shown in Figure 1. For a given set of $n = 7$ locations, $p = 3$ centers are to be determined. The optimal solution for the p-center problem is $\{A, B, D\}$ (marked with ⧫), and the objective function value is 12. The distance between user C and its nearest center A is 12 (marked with the dashed line), and this is the largest distance among all users. For the p-next center problem, the optimal solution is $\{A, B, C\}$ and the objective function value is 31. The distance between the user F and its nearest center B is 11, and the distance between the center B and its backup center A is 20, so the total distance is 31. In this case, this is the greatest distance among all users.

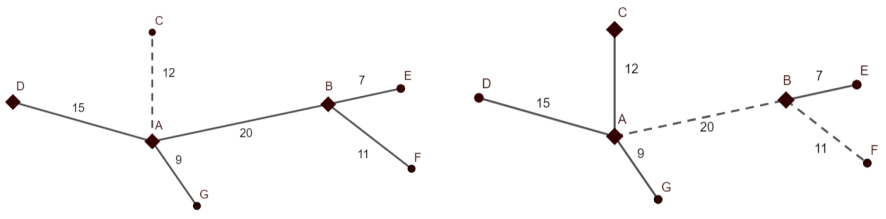

Figure 1: Optimal solution for p-center problem (left) and p-next center problem (right) for $n = 7, p = 3$

The p-next center problem was defined in 2015 by Albareda-Sambola et al. in [1]. The authors were motivated by a power distribution problem that occurred due to several earthquakes in Spain. They formulated several mathematical models for this problem and compared them. In the study by López-Sánches et al. [10] from 2018, a Greedy Randomized Adaptive Search (GRASP) and Variable Neighborhood Search (VNS), as well as their hybridization, were proposed as approximate methods for solving the p-next center problem. According to the computational results presented in [10] on a set of pmed instances with up to 200 nodes, the hybrid method obtained results of better quality compared to GRASP and VNS. In 2021, the Filtered VNS method was developed by Ristić et al. in [13], which is based on the idea of using different criteria for the acceptance of a solution approximation. The Filtered VNS method was tested on the same instances as in [10], extended by some larger examples with up to 900 nodes. In [9], the authors proposed three variants of evolutionary approach with a biased random-key Genetic Algorithm that incorporates a multi-parent strategy and a path-relinking as an intensification search procedure. They also presented optimal solutions for 401 out of extended set of 416 instances by running the exact solver CPLEX for one week and using 24 threads, but starting from the best solutions found by some other algorithms. Mousavi in [12] integrated local search with the combination of two strategies to exploit flat subspaces in the search space. The first strategy is to accept not only downhill moves, but also some flat moves. Let x and x' be two solutions with the same objective function value. The move from the solution x to the solution x' is called a flat move and should be analyzed using some heuristic function. In this way, the flat moves can be distinguished and the more promising ones can be accepted. The second strategy is to always accept flat moves unless forbidden by tabu restriction. The proposed algorithm was tested on 132 pmed instances with up to 200 nodes and outperformed the algorithm from [10].

The rest of the paper is organized as follows. In Section 2, the mathematical formulation of the p-next center problem is given. Sections 3 and 4 contain the description of the proposed approach to solve the considered problem. Experimental results are presented in Section 5 and a statistical analysis of the obtained results is given. General remarks and conclusions are given in Section 6 along with some future research directions.

2. Mathematical formulation of the problem

Let A be the set of users, each of them being a candidate for a center, let p be the total number of centers to be allocated, and let $D = (d_{ij})_{n\times n}$ be the distance matrix. In order to formulate the p-next center (PNC) problem as an integer linear programming problem, two sets of binary variables are defined:

$$y_j = \begin{cases} 1, & \text{if there is a center at the location } j, \\ 0, & \text{otherwise.} \end{cases}$$

$$x_{ij} = \begin{cases} 1, & \text{if } j \text{ is the nearest center to the location } i \text{ (different from } i\text{)}, \\ 0, & \text{otherwise} \end{cases}$$

Note that $x_{ij} = 1$ has a different meaning depending on whether there is a center at location i or not. If there is a center at location i, this means that a center j is a backup center for the center i. Otherwise, $x_{ij} = 1$ means that the center j is the primary center for the user i. The mathematical formulation of the PNC problem can be written as follows [1]:

$$\min w$$

$$\sum_{j \in A} y_j = p, \tag{1}$$

$$\sum_{j \in A,\, j \neq i} x_{ij} = 1, \quad i \in A, \tag{2}$$

$$x_{ij} \leq y_j \quad i, j \in A,\, i \neq j, \tag{3}$$

$$y_j + \sum_{k \in A,\, d_{ik} > d_{ij}} x_{ik} \leq 1 \quad i, j \in A,\, i \neq j, \tag{4}$$

$$w \geq \sum_{k \in A, k \neq j} d_{jk} x_{jk}, \quad j \in A, \tag{5}$$

$$w \geq d_{ij}(x_{ij} - y_i) + \sum_{k \in A, k \neq j} d_{jk} x_{jk}, \quad i, j \in A, i \neq j, \tag{6}$$

$$x_{ij} \in \{0, 1\}, \quad i, j \in A, i \neq j, \tag{7}$$

$$y_j \in \{0, 1\}, \quad j \in A. \tag{8}$$

The goal is to minimize the value of the continuous variable w, which represents the maximal distance from the user to its backup center (passing through the primary center). The model's constraints have the following meaning: there are exactly p centers established (constraint (1)), each user is allocated to exactly one primary center, and each primary center has exactly one backup center (constraint (2)). A user cannot be allocated to the location where the center is not established (constraints (3)). Constraints (4) are there to ensure that each user is allocated to the nearest established center. The constraints (5) set w to the correct value, i.e. the longest distance from a user to its backup center, and the constraints (6) ensure that all users with the same primary center are redirected to the same backup center, if necessary, and that the total distance in this case is equal to the distance from the user to the primary center plus the distance from the primary center to the backup center. The constraints (7) and (8) define the binary nature of the variables.

3. Skewed VNS method

Combinatorial optimization problems often have a large number of local optima that are not global optima. The number of local optima can increase exponentially with increasing dimensions of the problem. Such problems are very difficult to solve because classical approaches that guarantee finding the true global optimum take an unacceptably long time and often get stuck in one of the local optima, even for small problem dimensions. Therefore, many existing approaches for solving combinatorial optimization problems rely on (meta)heuristic methods. Metaheuristic methods provide general ideas and techniques that can be applied to a large number of problems [4,15,16]. Each step in a metacheuristic approach should be adapted to a specific problem to be solved. The main task is to find a balance between two processes: the local improvement of a solution and a diversification technique that should ensure that different parts of a search space are visited and also help escaping from local optima that are not global.

Variable Neighborhood Search (VNS) is a metaheuristic method proposed by Hansen and Mladenović in [6] in 1997. Since then, it has become one of the most popular metaheuristics as it achieves very promising results that usually outperform other approaches from the literature for both discrete and continuous optimization problems. A specific feature of the VNS method is the change of the neighborhood in which the local search is performed. This is done in a systematic way in order to avoid local optima. The method is based on the fact that a local optimum in a certain neighborhood does not necessarily have to be a local optimum in another neighborhood. On the other hand, the global optimum is the local optimum for any neighborhood.

The basic VNS algorithm (BVNS) combines these ideas with the stochastic approach, which is achieved by the so-called shaking step. Let $N = \{N_1, \ldots, N_{k_{max}}\}$ be a set of predefined neighborhood structures such that $N_k(x), 1 \leq k \leq k_{max}$ defines a k-th neighborhood of the solution x. The main idea of the shaking step is to examine these different neighborhood structures of a given solution x in order to escape the local optimum. To do this, one chooses another solution x' from the neighborhood $\mathrm{N}_k(x)$ and then performs some local search procedure with x' as the initial solution. If the obtained local optimum improves the objective function value, the algorithm moves on to this new solution and continues the search from there. Otherwise, the exploration of the larger neighborhood is considered in the shaking step. This neighborhood change guides the VNS while exploring the solution space.

The Skewed VNS (SVNS), explained below, performs especially well for problems with separated and far apart local optima. For such problems, it is sometimes impossible to reach distant optima using only the shaking procedure performed within the limited neighborhoods and the local search, as the BVNS method allows only moves that improve the current solution. On the other hand, if the significantly larger neighborhoods are considered, VNS behaves like a multistart algorithm. To overcome these limitations, SVNS accepts as new solutions not only improved solutions, but in some cases also those that are worse than the current best solution. A solution x'' obtained

from the local search is evaluated not only by its objective function value, but also by its distance to the current best solution x^*. The distance is defined by the metric $\varrho(\cdot,\cdot)$. In other words, moving from x^* to x'' is allowed if $f(x'') \leq f(x^*) + \alpha\varrho(x^*, x'')$, where $f(\cdot)$ is the objective function and α is a chosen parameter.

4. Main contribution

In this paper, an SVNS method for solving the p-next center problem is proposed. The main focus is on the development of an efficient local search method in order to obtain a high-quality solution in the shortest possible time.

The solution of the p-next center problem is represented by an integer vector of length p containing the indices of the locations that are chosen as centers. Once the locations of the centers are known, each user can be allocated to its nearest (primary) center, and a backup center can easily be found for each primary center.

For the method to be successful, the chosen neighborhood structure must be suitable for the problem. In this paper, the neighborhoods for the shaking step and the local search are defined as follows: $N_k(x) = \{x' \in S : d(x, x') = k\}$, where S is the feasible set, i.e. a set of all p-dimensional vectors with mutually different location indices, and $d(x, x')$ is defined as the total number of different locations chosen as centers for the solutions x and x'. For example, if $x = [1, 3, 6, 5, 9]$ and $x' = [6, 2, 4, 1, 9]$, x and x' have three equal locations for centers (1,6,9) and differ in two (3 and 5 in x, 2 and 4 in x'), i.e. $d(x, x') = 2$. In the shaking step, neighborhoods N_k, $k = 1, 2, \ldots, k_{max}$ are used, and the local search is performed in the neighborhood N_1 of the solution x'. The cardinality of the N_1 neighborhood is $p(n - p)$.

The local search has a major influence on the quality of the SVNS solutions obtained. On the one hand, the local search improves the current solution, on the other hand, one must not explore a part of the solution space for too long. This could lead to a local optimum that is not the global optimum. It is very important that a local search method is efficient but not very time consuming. The local search in this paper is performed using two procedures called: Move evaluation and Update. A similar technique was first mentioned in 1983 in [19], but its application started only in 1997 when it was used for solving the p-median problem in [7] and later for the standard p-center problem in [11]. To the best of the author's knowledge, this technique has not yet been used for the p-next center problem, and this is one of the main contributions of this paper. As defined above, the neighborhood $N_1(x)$ contains all solutions x' that have exactly one center different than the solution x. Before the search begins, two vectors c_1 and c_2 of length n are to be determined for the current solution x:

$c_1(i)$ – the closest center to the location i,

$c_2(i)$ – the second closest center to the location i, $\quad i = 1, 2, \ldots, n$.

To simplify the notation, it is assumed that the set of given locations is $A = \{1, 2, \ldots, n\}$. If there is a center at location i, then $c_1(i)$ is a backup center for center i. If the lo-

cation i is a user, then $c_1(i)$ is the primary center for i. If this center fails, then user i should proceed to its backup center, and that is $c_1(c_1(i))$. Note that $c_1(c_1(i))$ is generally not the same as $c_2(i)$, since the former is the second closest center from location i but passing through the primary center, and the latter is the second closest center directly from i to that center. The goal is to examine the entire neighborhood $N_1(x)$ and find the solution $x' \in N_1(x)$ that represents the local minimum in this neighborhood. This is done using the procedure Move evaluation which, for a given location $in \notin x$, finds a center $out \in x$ so that the value of $f((x \cup in) \setminus out)$ is minimal. In other words, for a given location in, this procedure finds the best choice of a center out in the current solution to be exchanged. This procedure also calculates the value $f((x \cup in) \setminus out)$, but it does not perform the interchange. The interchange is performed by the Update procedure, which also updates the values for the arrays c_1 and c_2 for the newly obtained solution $(x \cup in) \setminus out$.

4.1 Move evaluation procedure

Changing one element from the solution implies that one existing center should leave the solution and be replaced by a location that was not a center until that swap. For a given location that enters the solution (becomes a center), the Move evaluation procedure determines which center from the solution should be replaced by it, so that the objective function value is as small as possible. Two vectors r and z of length n play an important role in this procedure. In order to describe the procedure, the notation given in the Table 1 is used.

n	the number of locations
p	the number of centers to be established
$d(i,j)$	the distance between the locations i and j, $i,j = 1,2,\ldots,n$.
x	vector of length p containing the indices of the locations that are centers (current solution)
in	location that is becoming the part of solution (center)
$c_1(x)$, $c_2(x)$	arrays described above
$r(j)$	the largest distance from a user allocated to the center $j = x(l)$, $l = 1,2,\ldots,p$ to the center j, increased by the distance between j and its backup center, among all the users allocated to the center j.
$z(j)$	the largest distance between a user that was allocated to the center $j = x(l)$, $l = 1,2,\ldots,p$ and its new primary center increased by the distance between primary and backup center, under the assumption that center j is removed from the solution
out	center that is to be eliminated from the current solution (to be determined)
f	objective function value for the solution $(x \cup in) \setminus out$ (to be calculated)

Table 1: Notation for the Move evaluation procedure

Swapping two elements $in \in A \setminus x$ and $out \in x$ can affect all other users in the following way:

- *in* becomes the primary center for some users because it is closer to them than their current primary center. All of these users will need to update both their primary centers and their backup centers.

- *in* becomes the backup center of an existing center and thus a new backup center for some users. For all these users, only the backup centers need to be updated.

- *out* was the primary center for some users. For all these users, both the primary and the backup centers have to be determined.

- *out* was a backup center for some users. In this case, only their backup centers need to be determined.

By adding the location *in* to the solution, some of the users can gravitate to this new center, which means that the new center *in* is closer to them than their current primary center, i.e. $d(i, in) < d(i, c_1(i))$, $i \in A \setminus x$. For all these users, adding the center *in* to the solution would change their primary center and consequently their backup center. Let the set of all these users be denoted by X. The new objective function value could be reached for some of these users, so the value $f' = \max\limits_{i \in X}\{d(i, in) + d(in, c_1(in)\}$ should be determined. On the other hand, the remaining users could change their primary or backup center due to the removing of a center $out \in x$ from the solution. This center is still to be determined. For each center $out' \in x$, the remaining users can be divided into the following groups:

- Users whose primary and backup center would not change due to the removing of the center out'. Let X_j be a set of all such users with the same primary center j ($j, c_1(j) \neq out'$) and $d(i, j) < d(i, in)$, $i \in X_j$. Although the removal of the center out' from the solution had no effect on the users from X_j, the backup center for j could change due to the addition of the center *in* to the solution. Therefore, a new backup center new_backup_j for j should be determined among *in* and $c_1(j)$, and for each X_j the value $r(j) = \max_{i \in X_j}\{d(i, j) + d(j, new\_backup\_j)\}$ is to be calculated. The value $\max_j r(j)$ is a candidate for the new objective function value.

- Users u who would lose their primary or backup center by removing the center out'. For the users whose primary center was out', i.e. $c_1(u) = out'$, the new primary center $new\_primary\_u$ should be chosen between *in* and $c_2(u)$ in the following way:

$$new\_primary\_u = \begin{cases} in, & \text{if } d(u, in) < d(u, c_2(u)), \\ c_2(u), & \text{otherwise.} \end{cases}$$

After the primary center for all these users u has been changed, the new backup center should also be determined. If the primary center for user u was replaced by *in* (i.e. $new\_primary\_u = in$), its new backup center new_backup_u should be determined by:

$$new\_backup\_u = \begin{cases} c_1(in), & \text{if } c_1(in) \neq out', \\ c_2(in), & \text{if } c_1(in) = out'. \end{cases}$$

If, on the other hand, the primary center for the user u was replaced by $c_2(u)$, two cases must be considered in order to find the new backup center *new_backup_u*:

1. If $c_1(c_2(u)) \neq out'$, then *new_backup_u* is determined in the following way:
$$new\_backup\_u = \begin{cases} in, & \text{if } d(c_2(u), in) \leq d(c_2(u), c_1(c_2(u))), \\ c_1(c_2(u)), & \text{otherwise.} \end{cases}$$
2. If $c_1(c_2(u)) = out'$, then *new_backup_u* is found by:
$$new\_backup\_u = \begin{cases} in, & \text{if } d(c_2(u), in) \leq d(c_2(u), c_2(c_2(u))), \\ c_2(c_2(u)), & \text{otherwise.} \end{cases}$$

• For users u whose backup center was out' ($c_1(c_1(u)) = out'$), a new backup center *new_backup_u* should be determined in the following way:
$$new\_backup\_u = \begin{cases} in, & \text{if } d(c_1(u), in) < d(c_1(u), c_2(c_1(u))), \\ c_2(c_1(u)), & \text{otherwise.} \end{cases}$$

In the last two cases, the value $z(out')$ should be calculated as the maximal distance between each user u, that was using the center out' as a primary or backup center, and a new backup center (passing through the new primary center).

For a given location in and a fixed center $out' \in x$, the objective function value is
$$f((x \cup in) \setminus out') = \max\{f', \max_{j \neq out'} r(j), z(out')\}.$$
The goal is to determine the best center out to be deleted from the current solution x, and that is the one that minimizes the expression above. i.e.
$$out = \arg\min_{out' \in x} \{f((x \cup in) \setminus out')\} = \arg\min_{out' \in x} \{\max\{f', \max_{j \neq out'} r(j), z(out')\}\}.$$
In order to efficiently calculate the values $f', z(j), r(j), j = 1, \ldots, n$, every user is considered only once. Before that, the initial values must be set as follows:
$$f' = 0,$$
$$r(x(j)) = \min\{d(x(j), in), d(x(j), c_1(x(j)))\},$$
$$z(c_1(x(j))) = \max\left\{z(c_1(x(j))), \min\{d(x(j), in), d(x(j), c_2(x(j)))\}\right\}, j = 1, 2, \ldots, p.$$
At the beginning, for each user u it is established whether he gravitates towards the new center in, i.e. if $d(u, in) < d(u, c_1(u))$. If that is the case, the value f' should be updated: $f' = \max\{f', d(u, in) + d(in, c_1(in)\}$. If the value f' is changed in this step, current user u must be saved: $critical := u$, as well as his backup center $backup_{critical} := c_1(in)$. If the user u does not gravitate towards the center in, then the values $z(c_1(u))$, $z(c_1(c_1(u)))$ and $r(c_1(u))$ are updated for this user:
$$r(c_1(u)) = \max\Big\{r(c_1(u)),$$
$$\min\{d(u, c_1(u)) + d(c_1(u), c_1(c_1(u))), d(u, c_1(u)) + d(c_1(u), in)\}\Big\}, u \neq in,$$
$$r(c_1(in)) = \max\{r(c_1(in)), d(in, c_1(in))\},$$

$$z(c_1(u)) = \max\{z(c_1(u))),$$
$$d(u, new\_primary\_u) + d(new\_primary\_u, new\_backup\_u), \}u \neq in,$$
$$z(c_1(in)) = \max\{z(c_1(in))), d(in, c_2(in))\}.$$

Since the primary center $c_1(u)$ for user u is being deleted, the new primary center *new_primary_u* should be found between *in* and $c_2(u)$, depending on which one is closer to u. If $new\_primary\_u = in$, then the new backup center *new_backup_u* is $c_1(in)$ or $c_2(in)$ (in case that $c_1(u) = c_1(in)$, i.e. that $c_1(in)$ is the center that is being removed from the solution). If *new_primary_u* is $c_2(u)$, then the new backup center *new_backup_u* is to be chosen between $c_1(c_2(u))$ and *in* or $c_2(c_2(u))$ and *in*, if $c_1(c_2(u)) = c_1(u)$.

$$z(c_1(c_1(u))) = \max\{z(c_1(c_1(u)))), d(u, c_1(u)) + d(c_1(u), new\_backup\_u)\}$$

Since the backup center for the user u is deleted, the new backup center *new_backup_u* should be found between *in* and $c_2(c_1(u))$, whichever is closer to $c_1(u)$.

When all the users have been analyzed, values $z(x(j)), j = 1, 2, \ldots, p$ must be updated similar as mentioned above. This is necessary because once certain location is removed from the solution and it is not a center anymore, that location should be considered as a user. The distance from that location to its backup center should be calculated and it is a candidate for the objective function value after the swap. Next step is to find the values g_1 and g_2:

$$g_1 = \max\{r(x(l)) \,|\, l = 1, 2, \ldots, p\}, \text{ i.e. } g_1 = r(x(l^*)) \text{ for some } l^* \in \{1, 2, \ldots, p\},$$
$$g_2 = \max_{l \neq l^*}\{r(x(l)) \,|\, l = 1, 2, \ldots, p\}.$$

The final step is to construct an array $g(l), l = 1, 2, \ldots, p$ in a following way:

- If $l \neq l^*$, then $g(l) = \max\{f', z(x(l)), g_1\}$, If $x(l) = backup_{critical}$, then the value f' must be modified as $f' = f' - d(in, backup_{critical}) + d(in, c_2(in))$. The elimination of the center $x(l)$ is considered in this step, which could have been used as a backup center for *in* while updating the value f'. If this is the case, a new backup center for *in* must be found and the value of f' must be corrected before determining the value of $g(l)$.

- If $l = l^*$, then $g(l) = \max\{f', z(x(l)), g_2\}$, If $x(l) = backup_{critical}$, the similar correction as above must be made : $f' = f' - d(in, backup_{critical}) + d(in, c_2(in))$.

The best center *out* to be removed from the solution x, and objective function value f of the solution $(x \cup in) \setminus out$ are found in the following way: $f = \min\{g(l) | l = 1, 2, \ldots, p\}$ and let l^{**} be the index of the found minimum, then $out = x(l^{**})$.

4.2 Update procedure

As already mentioned, Move evaluation procedure does not swap the given center *in* and the found center *out*. This swapping is performed by the Update procedure, as is the calculation of the values of the vectors c_1 and c_2 for the new solution $(x \cup in) \setminus out$, more precisely the update of the closest and second closest centers for the users

affected by this swap. The same notation is used as for the Move evaluation procedure. The Update procedure for the PNC problem does not differ from this procedure for the PC problem [11]. For each location u that is such that $c_1(u) = out$, the new primary center must be found (i.e. $c_1(u)$ must be updated) between the locations in and $c_2(u)$, depending on which is closer to the location u. If this is $c_2(u)$, this center becomes the new $c_1(u)$, and then the second closest center for u must be found ($c_2(u)$ must be updated). For all remaining locations, such that $c_1(u) \neq out$, if $d(u, c_1(u)) > d(u, in)$, which means that in is the closest center for the user u, the new value for $c_2(u)$ should be set to $c_1(u)$, and the new value for $c_1(u)$ must be set to in. On the other hand, if $d(u, c_1(u)) < d(u, in)$ (i.e. the closest center remains the same), and if $d(u, in) < d(u, c_2(u))$, then only center $c_2(u)$ should be changed to in. On the other hand, if $d(u, in) > d(u, c_2(u))$, and if $c_2(u) = out$ new center $c_2(u)$ should be found.

4.3 Local search

The local search is implemented using two procedures already mentioned: Move evaluation and Update. The search starts from a certain initial solution x^*. For this solution, the arrays c_1 and c_2 must be found. Let f^* be the objective function value obtained by the solution x^*. In addition, the so-called critical user i^* must be found. The critical user is the one that travels the maximal distance f^* to its backup center, i.e. $f^* = d(i^*, c_1(i^*)) + d(c_1(i^*), c_1(c_1(i^*)))$. The following steps should be repeated while there is improvement in objective function value:

- For every location in that is not a part of the solution x^* and that is satisfied:
$$d(i^*, in) + d(in, c_1(in)) < d(i^*, c_1(i^*)) + d(c_1(i^*), c_1(c_1(i^*)))$$
start the Move evaluation procedure in order to find location out and objective function value f obtained by the solution $(x^* \cup in) \backslash out$. Let (in^*, out^*) be the best pair of locations to be interchanged, i.e. the solution $(x^* \cup in^*) \backslash out^*$ is the one that gives the minimum value of the objective function among all solutions from the neighborhood $N_1(x^*)$.

- If $f((x^* \cup in^*) \backslash out^*) \geq f^*$, the search is finished. If this is not the case, the following changes should be made: $f^* := f((x^* \cup in^*) \backslash out^*)$, using the procedure Update new values for x^*, c_1, c_2 must be set, as well as the new critical user i^* and previous step should be started again.

In this way, the local search of the N_1 neighborhood with the best improvement strategy is implemented in an efficient way.

4.4 Skewed VNS for PNCP

The proposed SVNS for the PNCP consists of the following steps.
Step 1: Initiallization.

1.1. Set the value of parameter k_{max} and define Neighborhood structures $N_1, \ldots, N_{k_{max}}$.

1.2. Generate the initial solution x^* at random, construct the corresponding arrays arrays c_1 and c_2. Calculate the objective function value f^* of the solution x^*.

1.3. Copy the values x^*, c_1, c_2 and f^* to the variables x', c_1', c_2', f'.

1.4. Set the values of parameters α and maximal number of iterations $iter\_max$ (stopping criteria parameter)

1.5. Define the distance $\varrho(x_1, x_2)$ as the total number od differing locations for the solutions x_1 and x_2.

Step 2: Set the parameter $iter$ to 1.

Step 3: Repeated the following steps until $iter \leq iter\_max$:

3.1. Set the parameter k to 1.

3.2. Until $k \leq k_{max}$ and $iter \leq iter\_max$ repeat the following steps:

3.2.1. $c_1' = c_1$ and $c_2' = c_2$

3.2.2. Apply the shaking step starting from the solution x' and c_1' and c_2' in order to obtain the solution x_shake. This is done by randomly choosing k different locations to become part of the solution and k centers to be removed from the solution, and then by executing the Update procedure k times.

3.2.3. Increase the parameter $iter$ by one.

3.2.4. Using x_shake as the initial solution and c_1' and c_2', apply the described local search method based on the Move evaluation and Update procedures. Let x'' be the resulting solution and f'' the objective function value obtained by x''.

3.2.5. If $f'' < f^*$ then the x^* should change: $x^* = x'', f^* = f''$.

3.2.6. If $f'' < f' + \alpha\varrho(x', x'')$ then: $x' = x''$, $f' = f''$, $c_1 = c_1'$, $c_2 = c_2'$ and $k = 1$, else the k parameter should be increased by one.

5. Experimental results

The proposed method was tested on the instances pmed1-pmed8 using the first n nodes and various values for p. The instances are available at [2]. Before the SVNS method can be implemented, distance matrices must be generated for each pmed dataset. This can be done using the Floyd-Warshall algorithm.

The SVNS method was tested on a total of 290 instances, which were divided into four groups:

- small instances (with up to 50 nodes)
- medium instances (between 60 and 200 nodes)
- large instances (containing between 250 and 400 nodes)
- additional 40 instances used in [13] (containing between 100 and 900 nodes)

The testing was done on the desktop computer with Intel Core i7-11700 3.6GHz processor and 16GB of RAM in a 64bit Windows 10 environment, using the parameter values given in Table 2.

Instance	$iter_{max}$	k_{max}	α	number of executions
small	500	5	0.1	20
medium	5000	5	0.1	20
large	1000	5	0.1	20
additional	1000	5	0.1	20

Table 2: SVNS parameter values

The CPLEX 20.1 solver was used to obtain the optimal solutions. The complete results of the proposed SVNS method for all 290 instances can be found on the [18]. The SVNS results for the set of small instances are presented in Table 3 and for the set of large instances in Tables 4 and 5 in the Appendix.

The following data is specified for each instance in addition to its name: $n-$ number of nodes, $p-$ number of centers, $best-$ the best result among all 20 executions, $\#best-$ number of executions (out of 20) where $best$ was found, $t_{best}-$ average time that elapsed, until $best$ was found for the first time (in seconds), $t_{tot}-$ average total time elapsed (in seconds), $agap-$ average gap between the values obtained and $best$ (in percent) and std_dev- standard deviation of the values obtained.

The optimal solution was found for all 50 small instances. The parameters $agap$ and std_dev were at most 0.94 and 4.09 respectively. The optimal solution was found for 122 out of 126 medium instances. For the remaining 4 instances, SVNS provided the upper bounds, which are marked with "*". The parameters $agap$ and std_dev were at most 4.00 and 2.1 respectively. An optimal solution was found for 61 out of 72 large instances, and upper bounds were found for the remaining 11 instances. The parameters $agap$ and std_dev are at most 9.86 and 5.75 respectively. Additional group of instances was used for testing in the work [13]. For 13 out of 42 instances from this group, the upper bounds were improved, while SVNS found the same value as the Filtered VNS for the remaining 29 instances, but in a shorter time.

5.1 Comparison with the previously proposed methods

In order to compare the results obtained with the SVNS method with the results of previous work, the Friedman test proposed by Demšar [3] is performed. This is a non-parametric test based on a ranking of all methods to be compared for each instance. The best method for a given instance is ranked 1, the second best is ranked 2 and so on. If the results are equal, the tied methods should receive the average rank. For this part, a subset of 63 instances that appeared in all previous works is considered. The first criterion for ranking is the objective function value. If two or more methods achieve the same value, the less time consuming method gets the better (lower) rank. To scale the times given in the previous work, the CPU ranks from CPU Benchmark [20] were used. A Table with the ranks can be found at [17].

The hypotheses to be tested are: the null hypothesis, which states that there are no significant differences between the methods compared and that the differences in solution quality are random, and the alternative hypothesis, which states that at least one method produces statistically different results from the other methods. The recommended test significance is $\alpha = 0.05$. The average ranks are calculated for each method. If the method i has the rank r_{ij} for the instance j, then the average rank R_i for the method i is calculated as $\frac{1}{N}\sum_{j=1}^{N} r_{ij}$, where N is the total number of instances to be compared (in this case 63). The average ranks for the compared methods are: 3.333 for GRASP-VNS hybrid from [10], 4.571 for Filtered VNS from [13], 4.079 for Evolutionary approach from [9], 1.4289 for Local Search from [12] and 1.619 for SVNS. For $N = 63$ instances and $k = 5$ methods for comparison, using the formula:

$$\chi_F^2 = \frac{12N}{k(k+1)}\left(\sum_{l=1}^{k} R_k^2 - \frac{k(k+1)^2}{4}\right),$$

the value $\chi_F^2 = 209.37$ is calculated. This value is greater than the critical value 9.49 for $\alpha = 0.05$ and $df = k - 1 = 4$ [14], so the null hypothesis is rejected. According to Friedman, there are significant differences between the tested methods. To determine which of the methods differ, the Nemenyi test is used. Two methods are significantly different if their average ranks differ by at least the critical difference CD. The value CD is calculated as follows: $CD = q_a\sqrt{\frac{k(k+1)}{6N}}$, and in this case $CD = 0.7685$, for $q_a = 2.728$ [3]. This means that the results of the SVNS and LS methods differ significantly from those of the three previously proposed methods.

6. Conclusion

This paper considers the p-next center problem, assuming that one or more centers may suddenly fail due to various problems. In this case, the users arriving at the closed center should be redirected to the backup center. An SVNS method using the local search that relies on the Move evaluation procedure and the Update procedure is suggested. All the elements of the metaheuristic have been adjusted to the specific features of the p-next center problem. The proposed method has been tested on the pmed set of instances, and the results were compared with the previously published results on this topic. From all the presented, it can be concluded that the proposed method is suitable for solving the p-next center problem and the statistical analysis shows that the presented results are significantly better than the results of three previously proposed methods.

References

[1] M. Albareda-Sambola, Y. Hinojosa, A. Marín, J. Puerto, *When centers can fail: A close second opportunity*, Comput. Oper. Res., **62** (2015), 145–156.

[2] J. E. Beasley, *OR-Library*, http://people.brunel.ac.uk/∼mastjjb/jeb/orlib/pmedinfo.html, (*last access: 5. January 2022.*)

[3] J. Demšar,*Statistical comparisons of classifiers over multiple data sets*, J. Mach. Learn. Res., **7** (2006), 1–30.

[4] F. W. Glover, G. A. Kochenberger, *Handbook of metaheuristics*, Springer, Science & Business Media, 2006.

[5] S.L. Hakimi, *Optimum distribution of switching centers in a communication network and some related graph theoretic problem*, Oper. Res., **13(3)** (1965), 462–475.

[6] P. Hansen, N. Mladenović, *Variable neighborhood search*, Computers & Operations Research, 1997.

[7] P. Hansen, N. Mladenović, *Variable neighborhood search for the p-median*, Locat. Sci., **5(4)** (1997), 207–226.

[8] O. Kariv, S.L. Hakimi, *An algorithmic approach to network location problems. Part 1: The p-Centers*, SIAM J. Appl. Math., **37(3)** (1979), 513–538.

[9] M. A. Londe, C. E Andrade, L. S. Pessoa, *An evolutionary approach for the p-next center problem*, Expert Syst. Appl., **175** (2021), 114728.

[10] A.D. López-Sánchez, J. Sánchez-Oro, A.G. Hernandez-Dõaz, *GRASP and VNS for solving the p-next center problem*, Comput. Oper. Res., (2018).

[11] N. Mladenović, M. Labbe, P. Hansen, *Solving the p-Center Problem with Tabu Search and Variable Neighborhood Search*, Networks, **42(1)** (2003), 48–64.

[12] S. R. Mousavi, *Exploiting flat subspaces in local search for p-Center problem and two fault-tolerant variants* , Comput. Oper. Res., (2022).

[13] D. Ristić, N. Mladenović, R. Todosijević, D. Urosević, *Filtered variable neighborhood search method for the p-next center problem*, IJTTE, (2021)

[14] D. J. Sheshkin, *Handbook of parametric and nonparametric statistical procedures*, Chapman & Hall/CRC, 2000.

[15] E. G. Talbi, *Metaheuristics: from design to implementation*, John Wiley &Sons, 2009.

[16] E. G. Talbi, *Hybrid metaheuristics*, Springer, 2013.

[17] J. Tasić, *Ranks for the comparison with the previously suggested methods*, http://www.matf.bg.ac.rs/p/files/1672706856-77-ranking.pdf, (*last access: 15. January 2023.*)

[18] J. Tasić, *Complete results of the SVNS mehod for the p−next center problem*, http://www.matf.bg.ac.rs/p/files/1674256817-77-rezultati_za_rad.pdf, (*last access: 15. January 2023.*)

[19] R. Whitaker, *A fast algorithm for the greedy interchange for large-scale clustering and median location problems*, INFOR, **21(2)** (1983).

[20] *CPU Benchmark*, https://www.cpubenchmark.net, (*last access: 10. January 2023.*)

(received 23.01.2023; in revised form 25.05.2023; available online 17.02.2024)

University of Belgrade, Faculty of Mathematics, Serbia
E-mail: jelena.tasic@matf.bg.ac.rs

Appendix

Instance	n	p	$best$	$\#best$	$t_{best}(s)$	$t_{total}(s)$	$agap(\%)$	std_dev
pmed1	10	5	84	20	0.00	0.01	0	0
pmed1	20	5	120	20	0.00	0.02	0	0
pmed1	20	10	95	20	0.00	0.01	0	0
pmed1	30	5	126	20	0.00	0.03	0	0
pmed1	30	10	95	20	0.00	0.02	0	0
pmed1	40	5	144	20	0.00	0.06	0	0
pmed1	40	10	111	20	0.01	0.04	0	0
pmed1	40	20	89	20	0.00	0.03	0	0
pmed1	50	20	89	20	0.01	0.04	0	0
pmed2	10	5	121	17	0.00	0.00	0.87	2.07
pmed2	20	5	147	20	0.00	0.02	0	0
pmed2	20	10	99	20	0.00	0.01	0	0
pmed2	30	5	169	20	0.00	0.03	0	0
pmed2	30	10	110	20	0.00	0.03	0	0
pmed2	40	5	164	20	0.00	0.06	0	0
pmed2	40	10	112	19	0.01	0.05	0.94	4.09
pmed2	40	20	96	20	0.01	0.04	0	0
pmed2	50	10	140	20	0.00	0.06	0	0
pmed2	50	20	99	20	0.00	0.05	0	0
pmed3	10	5	77	20	0.00	0.00	0	0
pmed3	20	5	145	20	0.00	0.02	0	0
pmed3	20	10	77	20	0.00	0.01	0	0
pmed3	30	5	157	20	0.00	0.04	0	0
pmed3	30	10	122	20	0.00	0.02	0	0
pmed3	40	5	157	20	0.00	0.06	0	0
pmed3	40	10	105	20	0.00	0.04	0	0
pmed3	40	20	77	20	0.00	0.04	0	0
pmed3	50	10	125	15	0.01	0.06	0.4	0.69
pmed3	50	20	87	20	0.00	0.05	0	0
pmed4	10	5	126	20	0.00	0.00	0	0
pmed4	20	5	139	20	0.00	0.02	0	0
pmed4	20	10	125	20	0.00	0.01	0	0
pmed4	30	5	173	20	0.00	0.04	0	0
pmed4	30	10	122	20	0.00	0.03	0	0
pmed4	40	20	85	20	0.00	0.031	0	0
pmed4	50	10	126	20	0.00	0.06	0	0
pmed4	50	20	91	20	0.01	0.05	0	0
pmed5	10	5	125	20	0.00	0.00	0	0
pmed5	20	5	139	20	0.00	0.02	0	0
pmed5	20	10	91	20	0.00	0.01	0	0
pmed5	30	5	155	19	0.01	0.04	0.03	0.14
pmed5	30	10	120	20	0.00	0.03	0	0
pmed5	40	5	164	20	0.00	0.05	0	0
pmed5	40	10	127	20	0.00	0.04	0	0
pmed5	40	20	91	20	0.00	0.03	0	0
pmed5	50	10	121	20	0.00	0.06	0	0
pmed5	50	20	89	12	0.01	0.04	0.45	0.55

Table 3: Results for small pmed instanaces with $10 \leq n \leq 50$, $5 \leq p \leq 20$

Instance	n	p	$best$	$\#best$	$t_{best}(s)$	$t_{total}(s)$	$agap(\%)$	std_dev
pmed11	250	30	44	12	1.56	2.26	2.5	3.80
pmed11	250	50	42	5	0.88	1.69	2.5	1.76
pmed11	250	70	42	20	0.43	1.16	0	0
pmed11	250	90	42	20	0.13	0.91	0	0
pmed11	300	60	51	20	0.26	2.05	0	0
pmed11	300	80	51	20	0.10	1.55	0	0
pmed11	300	100	51	20	0.05	1.31	0	0
pmed11	300	150	51	20	0.02	1.01	0	0
pmed12	250	30	48*	5	1.64	2.12	3.96	3.29
pmed12	250	50	45*	1	0.63	1.70	5	2.32
pmed12	250	70	43	4	0.72	1.25	1.98	1.101
pmed12	250	90	43	19	0.36	0.99	0.12	0.51
pmed12	300	60	72	20	0.03	1.16	0	0
pmed12	300	80	72	20	0.03	0.88	0	0
pmed12	300	100	72	20	0.02	0.75	0	0
pmed12	300	150	72	20	0.02	0.92	0	0
pmed13	250	30	48*	3	1.41	2.09	5.83	4.18
pmed13	250	50	44	1	1.16	1.77	5.45	2.32
pmed13	250	70	44	16	0.68	1.26	0.80	1.65
pmed13	250	90	44	20	0.21	1.02	0	0
pmed13	300	60	43*	1	1.74	2.33	7.09	3.99
pmed13	300	80	41*	2	1.09	1.85	5.12	2.66
pmed13	300	100	39	5	1.22	1.71	2.31	1.60
pmed13	300	150	39	20	0.34	1.25	0	0
pmed14	250	30	60	20	0.42	2.16	0	0
pmed14	250	50	60	20	0.14	1.57	0	0
pmed14	250	70	60	20	0.05	0.97	0	0
pmed14	250	90	60	20	0.04	0.74	0	0
pmed14	300	60	60	20	0.20	1.91	0	0
pmed14	300	80	60	20	0.13	2.09	0	0
pmed14	300	100	60	20	0.07	1.25	0	0
pmed14	300	150	60	20	0.02	1.05	0	0
pmed15	250	30	49	15	1.17	2.05	0.51	0.88
pmed15	250	50	49	20	0.36	1.54	0	0
pmed15	250	70	49	20	0.12	1.03	0	0
pmed15	250	90	49	20	0.07	0.82	0	0
pmed15	300	60	44	17	1.15	2.01	0.34	0.81
pmed15	300	80	44	20	0.61	1.99	0	0
pmed15	300	100	44	20	0.30	1.76	0	0
pmed15	300	150	44	20	0.03	0.99	0	0

Table 4: Results for large pmed instanaces with $250 \leq n \leq 400$, $30 \leq p \leq 200$

Instance	n	p	$best$	$\#best$	$t_{best}(s)$	$t_{total}(s)$	$agap(\%)$	std_dev
pmed16	350	40	37*	1	3.14	4.77	9.86	5.75
pmed16	350	70	34*	12	1.86	2.94	2.35	3.03
pmed16	350	90	33	2	1.26	2.33	2.73	0.91
pmed16	350	120	33	15	1.14	2.45	0.76	1.31
pmed16	400	80	34*	17	8.75	12.87	0.74	1.83
pmed16	400	100	33	4	5.71	9.77	2.42	1.21
pmed16	400	140	33	15	2.48	6.53	0.76	1.31
pmed16	400	200	33	20	0.73	4.37	0	0
pmed17	350	40	39*	19	2.33	4.44	0.13	0.56
pmed17	350	70	37	14	1.65	2.69	1.35	2.18
pmed17	350	90	37	19	0.68	2.10	0.27	1.18
pmed17	350	120	37	20	0.23	1.59	0	0
pmed17	400	80	37	18	6.08	11.16	0.41	1.29
pmed17	400	100	37	20	3.09	9.37	0	0
pmed17	400	140	37	20	0.70	6.02	0	0
pmed17	400	200	37	20	0.15	4.04	0	0
pmed18	350	40	50	20	0.24	4.45	0	0
pmed18	350	70	50	20	0.08	2.35	0	0
pmed18	350	90	50	20	0.05	1.74	0	0
pmed18	350	120	50	20	0.02	1.31	0	0
pmed18	400	80	50	20	0.48	9.33	0	0
pmed18	400	100	50	20	0.42	7.23	0	0
pmed18	400	140	50	20	0.19	4.86	0	0
pmed18	400	200	50	20	0.02	3.99	0	0
pmed19	350	40	38*	1	2.47	4.40	6.84	3.05
pmed19	350	70	35	1	1.99	2.955	5.57	2.11
pmed19	350	90	35	12	1.60	2.38	1.14	1.40
pmed19	350	120	35	20	0.62	1.76	0	0
pmed19	400	80	34*	2	9.47	13.04	6.62	3.07
pmed19	400	100	32	4	7.13	9.72	6.41	4.01
pmed19	400	140	32	19	3.22	6.48	0.16	0.68
pmed19	400	200	32	20	0.35	4.06	0	0

Table 5: Results for large pmed instanaces with $250 \leq n \leq 400$, $30 \leq p \leq 200$