Computers & Operations Research 175 (2025) 106916

Contents lists available at ScienceDirect

Computers and Operations Research

journal homepage: www.elsevier.com/locate/cor

A VNS method for the conditional p-next center problem

Jelena Tasić *, Zorica Dražić, Zorica Stanimirović

Faculty of Mathematics, University of Belgrade, Studentski Trg 16, 11 000 Belgrade, Serbia

ARTICLE INFO

Keywords:
Conditional p-next center problem
Variable neighborhood search
Fast interchange heuristic

ABSTRACT

This paper considers the conditional p-next center problem (CPNCP) and proposes a metaheuristic method as a solution approach. The p-next center problem (PNCP) is an extension of the classical p-center problem that captures real-life situations when centers suddenly fail due to an accident or some other problem. When the center failure happens, the customers allocated to the closed center are redirected to the center closest to the closed one, called the backup center. On the other hand, when a service network expands, some of the existing centers are usually retained and a number of new centers are opened. The conditional p-next center problem involves both of these two aspects that arise in practice and, to the best of our knowledge, has not been considered in the literature so far. Since the CPNCP is NP-hard, a metaheuristic algorithm based on the Variable Neighborhood Search is developed. The proposed VNS includes an efficient implementation of the Fast Interchange heuristic which enables the VNS to tackle with real-life problem dimensions. The exhaustive computational experiments were performed on the modified PNCP test instances from the literature with up to 900 nodes. The obtained results are compared with the results of the exact solver CPLEX. It is shown that the proposed VNS reaches optimal solutions or improves the feasible ones provided by CPLEX in a significantly shorter CPU time. The VNS also quickly returns its best solutions when CPLEX failed to provide a feasible one. In order to investigate the effects of two different approaches in service network planning, the VNS solutions of the CPNCP are compared with the optimal or best-known solutions of the p-next center problem. In addition, the conducted computational study includes direct comparisons of the results obtained when the proposed SVNS is applied to PNCP (by setting the number of existing centers to 0) with the results of recent solution methods proposed for the PNCP.

1. Introduction

The p-center problem (PCP) is one of the most studied location problems in the literature. It was defined in 1965 by Hakimi in Hakimi (1965) as follows. For a set of n locations and given distances between them, the objective is to choose p locations ($p < n$) for the centers to be established and to assign each of the remaining $n - p$ locations (customers, users) to its nearest center. All established centers are identical and there is no limit for the number of customers that can be assigned to a center. The centers should be chosen so that the maximal distance from each customer to its nearest center is minimal. In the classical p-center, the focus is on the customer in the worst position. This is a realistic perspective considering that the centers represent schools, bus stations, hospitals, fire stations, etc., so each user has to be as close as possible to its nearest center. Considering the practical importance of the p-center problem, numerous solution approaches have been proposed in the literature so far (see Celik Turkoglu and Erol Genevois, 2020; Drezner, 1984; ReVelle and Eiselt, 2005; Tansel et al., 1983). Since the p-center belongs to the class of NP-hard problems (Kariv and Hakimi, 1979), many solution methods for PCP are heuristics, for example Mladenović et al. (2003), Pelegrin (1991), Pullan (2008), etc.

1.1. The p-next center problem

In order to capture different aspects of real-life situations, the p-center problem has been modified and extended in various ways. Since the p-center problem is often used to describe emergency situations (fire, injuries, earthquakes, etc.), it is natural to consider the possible failure of some centers. This assumption leads to an extension of the PCP, called the p-next center problem (PNCP), which was defined by Albareda-Sambola et al. (2015). The main idea behind the PNCP is to determine the backup center for each primary center so that if the primary center is closed, all users assigned to that center can proceed to the backup center. Since the failure of a center is often unpredictable, it is natural to assume that users will find out about the failure when they arrive at their primary center. In such a situation, it is also reasonable to assume that users will proceed directly to some other center instead

* Corresponding author.
E-mail address: jelena.tasic@matf.bg.ac.rs (J. Tasić).

https://doi.org/10.1016/j.cor.2024.106916
Received 16 November 2023; Received in revised form 15 November 2024; Accepted 15 November 2024
Available online 26 November 2024

of returning to their home location and choosing the second closest center. Therefore, it is assumed that a customer will pass through the primary center to reach its backup center. This implies that the backup center for a closed center is always the center closest to the closed one. The goal of the PNCP is to determine the locations of p centers such that the maximal distance from a user to its backup center via the primary center is minimal.

Up to now, several methods for solving PNCP have been proposed in the literature. López-Sánchez et al. (2019) presented a Greedy Randomized Adaptive Procedure (GRASP), as well as a Variable Neighborhood Search (VNS) and the hybridization of these two methods. Computational experiments on the set of instances with up to 200 nodes showed that the hybrid method outperforms both GRASP and VNS. In the paper (Ristić et al., 2021), the authors proposed the Filtered VNS method (FVNS) and introduced an Additional set of pmed instances with up to 900 nodes, which was used in their computational study. An efficient VNS-based heuristic was also proposed as a solution method for the PNCP in Tasić (2024). The heuristic from Tasić (2024) reached or improved the best FVNS results for the standard PNCP instances with up to 200 nodes considered in Ristić et al. (2021). For the Additional pmed instances, the heuristic presented in Tasić (2024) showed to be superior to FVNS in terms of the quality of the solution. Londe et al. (2021) developed an evolutionary approach (EA) for the PNCP. Additionally, starting from the best solutions found with another algorithm, they ran the CPLEX solver with 24 threads for up to a week for some instances and reported optimal solutions for the extended set of instances. Mousavi (2023) proposed a variant of the local search method and applied two strategies to exploit the flat subspaces. In the first one, flat moves are evaluated using a certain heuristic function in order to determine whether this move leads to a promising solution. The second approach includes a tabu restriction for some flat moves, which are marked as forbidden, while all other flat moves are allowed. The local search method (Mousavi, 2023) was tested on a set of 132 benchmark instances with up to 200 nodes and provided better quality solutions than the ones reported in López-Sánchez et al. (2019). Zhang et al. (2022) proposed a weighting-based tabu search algorithm (WTS) as a solution approach for the PNCP. The WTS decomposes the PNCP into a series of decision subproblems and solves each of them with a fast tabu search procedure. A similar idea was used in the recent paper (Zhang et al., 2023) for the classical p-center problem. Recently, Ristić et al. in Ristić et al. (2023a) designed a modified Basic VNS algorithm (BVNS) that uses a refined local search and shake step, and also exploits auxiliary data structures used in VNS for the classical p-center problem. The computational results on the subset of PNCP instances used in Ristić et al. (2021) showed that the BVNS from Ristić et al. (2023a) outperformed the previous state-of-the-art PNCP methods (in terms of solution quality) on the set of PNCP instances from Ristić et al. (2023a).

1.2. Conditional p-center problem

The application of the p-center problem in practice usually turns into the conditional case. For example, when a particular service network expands, there is a need for a larger number of facilities, but this does not mean that the existing facilities must be canceled. The conditional p-center problem (CPCP) aims to find locations for p new centers while keeping the existing q centers open. The objective is the same as in the PCP: the maximal distance from a user to its nearest center, among all $p+q$ centers, should be minimized. The conditional p-median problem is similarly defined as an extension of the classical p-median problem. The CPCP is often denoted as the (p,q) center problem, as in Drezner (1995).

Lin (1975) was the first to mention the conditional p-center problem in his paper from 1975 in which the problem of adding one new facility to the existing system was discussed ($p=1, q\geq 1$). The same version of the conditional p-center problem was also considered in Handler and Mirchandani (1979). Chen (1990) studied both the conditional p-center and the conditional p-median problem in the Euclidean space. Chen and Handler (1993) considered a general variant of the CPCP ($p\geq 1$) in the plane. The relaxation method used for the PCP was adapted for the conditional problem, and its performance was evaluated on a set of instances with up to 200 randomly distributed locations.

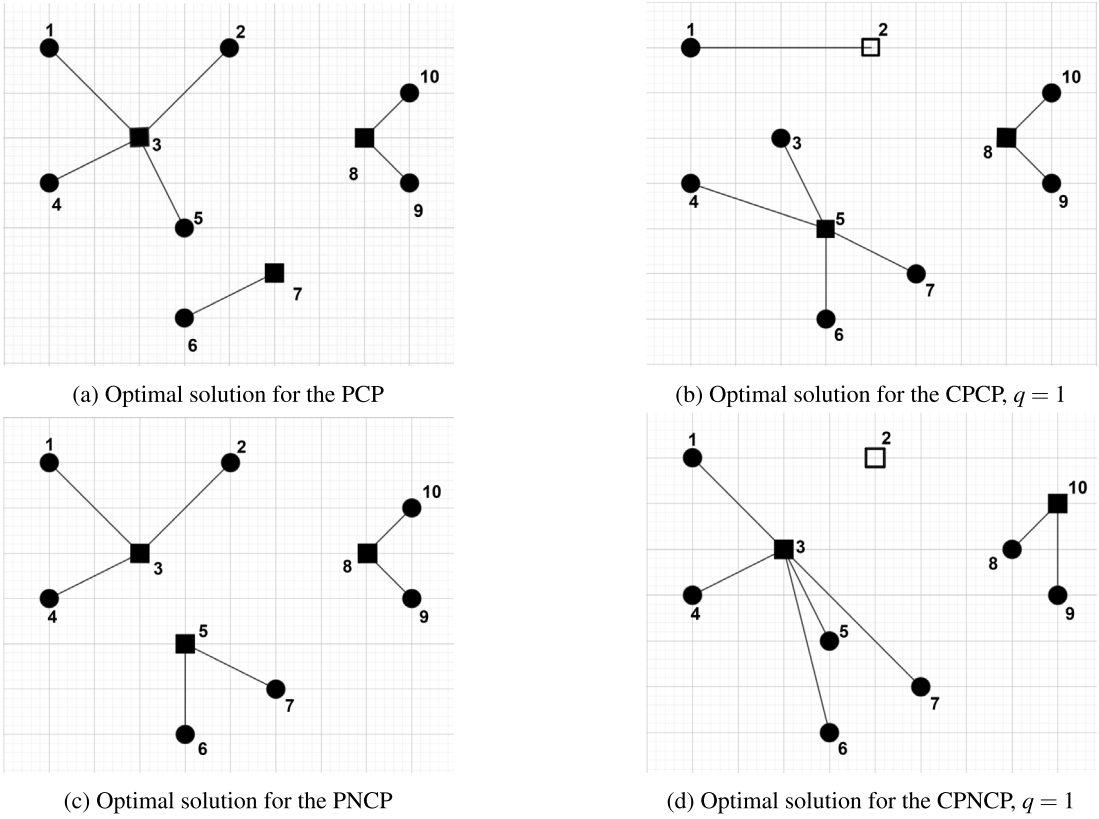
Minieka (1980) introduced the conditional versions of the p-center and p-median problem on the graph, but only the case of adding a single new facility was considered in detail. Drezner (1989) proposed a method for solving the CPCP both in a network and in the plane. The first step of this method is to sort all customers in descending order according to their distance to the nearest center. In the optimal solution of the conditional problem, the first r customers are allocated to some of the new centers. The task is to determine the value of r, and this is done using the binary search algorithm and solving one p-center problem in each iteration. Furthermore, Drezner (1989) proves that the overall complexity of the algorithm is $P(n)O(\log n)$, where n is the number of the customers and $P(n)$ is the complexity of the algorithm used for solving p-center problems. Berman and Simchi-Levi (1990) presented a solution approach to the conditional p-median problem and the conditional p-center problem based on solving one unconditional $(p+1)$-median problem and one unconditional $(p+1)$-center problem, respectively. The improvement of the method from Berman and Simchi-Levi (1990) was proposed by Berman and Drezner (2008), who performed a computational study using the CPLEX solver on the pmed set of instances with up to 700 nodes.

The Drezner's algorithm (Drezner, 1989) was further modified by Chen and Chen (2010), resulting in an iterative algorithm for the CPCP. The first step is the same as in Drezner (1989), while a different method is used to determine the value of r. In each step of the iterative algorithm, the bounds of the optimal solution are improved by solving one p-center subproblem. Initially, the subproblem with only one demand point is solved, and this point is the first demand point from the descending order. In each iteration, the next demand point from the descending order is added. In this way, at most n p-center problems are solved, but with a small number of demand points (between 1 and $r+1$), while in Drezner's method subproblems can have more than $r+1$ demand points.

Iravan et al. (2016) proposed two metaheuristic approaches based on VNS, guided multi-start principle, aggregation techniques, and some exact methods. The computational study was performed on a Traveling Salesman Problem (TSP) dataset with up to 71 009 nodes. Continuous variants of PCP and CPCP (the centers can be established anywhere in the plane) were studied in Callaghan et al. (2018). The paper provides an improved variant of the relaxation-based algorithm previously proposed by Chen and Chen (2010), and it was tested on a TSP dataset.

1.3. Motivation and main contribution of the study

The PCP, PNCP and CPCP problems mentioned above deal with many real life problems. It is well known that the PCP problem is used for locating emergency and health services such as ambulances, police, fire stations, etc. The practical significance of the CPCP is that in most situations it is more convenient to integrate new facilities into the existing network than to design a completely new system from scratch. Since establishing new service centers is usually expensive, it makes sense to keep the existing centers open and add the new ones. On the other hand, emergency networks assume that each facility can respond to all customer demands at all times. Many disasters such as earthquakes, fires, floods, tsunamis, hurricanes, etc. have caused the destruction of one or more emergency service facilities. Therefore, customers cannot rely on their primary center and have to continue to the nearest backup center. These facts have motivated us to combine the two aspects mentioned above and present a variant of the p-next center problem with the additional assumption that some centers are already established. The goal is to find the locations of a fixed number of new



(a) Optimal solution for the PCP

(b) Optimal solution for the CPCP, $q = 1$

(c) Optimal solution for the PNCP

(d) Optimal solution for the CPNCP, $q = 1$

Fig. 1. Comparison of the optimal solutions for (a) PCP, (b) CPCP, (c) PNCP, (d) CPNCP on a small-size network.

centers such that the maximal distance of a customer to its backup center is minimal. We refer to this location problem as the conditional p-next center problem (CPNCP) or (p, q)-next center problem. To the best of our knowledge, this is the first reference concerning this variant of PNCP. Since the PNCP is NP-hard (as an extension of the PCP), the considered CPNCP also belongs to the class of NP-hard optimization problems.

In order to graphically illustrate the effects of using different network design strategies given by PCP, CPCP, PNCP, and CPNCP on the optimal solution, let us consider a small example with $n = 10$ users and 3 centers. Let $\{1, 2, \ldots, 10\}$ be the set of given locations, as shown in Fig. 1, and assume that three centers are required for this network. The Fig. 1(a) shows the optimal solution for the PCP for $p = 3$. The centers are located at nodes 3, 7, and 8 (marked with ■), while the allocations of the customers to the centers are denoted with black lines. The maximal distance to its closest center is traveled by users 1 and 2 and the distance is 2.83. The optimal solution for the CPCP, assuming that one center is already located at node 2 while two more centers need to be established ($p = 2$, $q = 1$), is given in Fig. 1(b). In this case, the two new centers are located at nodes 5 and 8, and user 1 travels the maximal distance of 4. Fig. 1(c) shows the optimal solution for the PNCP for $p = 3$. The centers are opened at nodes 3, 5, and 8, while the maximal distance of 5.89 is traveled by user 10. The value of 5.89 is obtained by summing up the distance from 10 to its primary center 8 and the distance from center 8 to its backup center 5. In Fig. 1(d) the optimal solution of the CPNCP is given, assuming that one center is already opened at node 2 and two additional centers need to be opened ($p = 2$, $q = 1$). In the optimal solution of the CPNCP, new centers are established at nodes 3 and 10, with user 7 traveling the longest total distance of 7.07 (the distance from 7 to 3 increased by the distance from 3 to 2). Although these problems have some similarities, adding new realistic factors to the model results in significantly different optimal solutions in terms of the locations of the established centers and the corresponding objective function values. For example, the optimal solutions of CPCP and CPNCP, which are shown in Figs. 1(b) and 1(d), respectively, differ in all located centers, with the exception of the center at location 2, which was already opened. This means that additional insight regarding the possible collapse of centers leads to different sets of new centers to be opened in the CPCP and CPNCP. Figs. 1(c) and 1(d) show that in the case of the CPNCP, the already opened center at location 2 is not part of the optimal solution of PNCP. This implies that the existing centers affect the decision on the locations for opening the remaining centers.

The previous example illustrates the need to introduce the CPNCP that captures two important aspects of modeling an emergency service network: maintaining the existing service centers opened and ensuring that in case of a primary service center failure, its users are redirected to the backup center closest to the primary one. Apart from emergency service networks, the proposed CPNCP also finds its applications in other areas. CPNCP may be used to make strategic decisions in any service facility network facing problems such as power outages, staff or capacity shortages, technical failures, etc. One example is the localization of ATMs in urban areas. A user cannot know in advance whether an ATM is out of order. If that turns out to be the case on his arrival, the user would go from there to the nearest ATM. When deciding where to locate new ATMs, this scenario should also be considered. Another example is the gas station network of the same company. Many drivers are accustomed to the quality of fuel offered by a particular company and tank the fuel only at the gas stations of this company. The companies further motivate drivers to use only their services by offering loyalty cards and discounts depending on the amount of fuel filled at their gas stations. So, if the closest gas station to a driver's home or workplace is out of order, due to maintenance, fuel distribution or working hours, the driver will most likely go to the gas station that is the closest to the primary one. A similar scenario occurs when planning the network of drugstores of the same company, the network of supermarkets of the same brand, the network of hospitals of a private healthcare company, etc. If a customer is satisfied with the services

offered by one company, he will most likely go to the branch that is nearest to his living or working place, and if the service is not available, he will go to the nearest branch from there. Companies encourage this customer behavior in different ways by offering them coupons and discounts if they spend a certain amount of money in their branches.

The main contributions of this paper are the following.

- We consider the conditional p-next center problem as a variant of the p-next center problem. To the best of our knowledge, the CPNCP has not been studied in the literature so far. The example presented in Fig. 1 illustrates the differences between the optimal solutions of PCP, PNCP, CPCP, and the considered CPNCP for the same problem instance. The mathematical formulation of the CPNCP is presented, which is obtained by adapting the integer linear programming (ILP) formulation of the PNCP from Albareda-Sambola et al. (2015).
- As the considered CPNCP is NP-hard, we design and implement an efficient variant of VNS, called Skewed VNS (SVNS), to solve the CPNCP. The use of the SVNS metaheuristic is motivated by the importance of having an efficient solution algorithm to tackle instances with large problem dimensions that occur in practice. In the case of real-life problem dimensions, exact methods often fail to provide a solution due to memory or time limits. In addition, SVNS can be used in situations where a high-quality solution is required in a short time. This may be the case when designing an emergency system consisting of mobile emergency units that are temporarily set up at certain positions and can be moved to other positions, if necessary. During a natural disaster or war crisis, the set of potential locations for the centers or the set of established centers may suddenly change completely or in great extent (for example, locations must be moved to another area or region), making reassignment of customers to backup centers either inefficient or impossible. This situation requires prompt reactions and a redesign of the emergency system, which can be efficiently done by the SVNS metaheuristic.
- The proposed SVNS includes an efficient implementation of the Fast Interchange (FI) heuristic in the Local Search phase. In addition, recent studies on PNCP (Ristić et al., 2021, 2023b; Tasić, 2024) show that a VNS-based metaheuristic is a successful method for PNCP, and therefore the SVNS metaheuristic is investigated as a promising solution approach for CPNCP. The elements of the proposed SVNS have been carefully designed and implemented in accordance with the characteristics of the problem. Moreover, in our SVNS, we use Move Evaluation and Update procedures instead of the classical swap and/or update from scratch, which leads to a speedup of the algorithm and a reduction of its time complexity.
- In order to investigate the effects of the possibility of accepting worse solutions, we perform experiments for different values of the parameter α, which controls the solution acceptance in the SVNS. Note that in the case of $\alpha = 0$ (also included in the analysis), the algorithm is reduced to the classical BVNS. In a separate subsection, the results of the parameter tuning test and their statistical analysis are presented. The obtained results and the statistical tests indicate that there is a significant difference between using BVNS and SVNS for solving CPNCP, i.e., that the proposed SVNS is superior to BVNS for the considered problem.
- For the purpose of computational analysis, we modified the well-known ORLIB pmed instances from the literature to fit the considered CPNCP. In our experimental study, we used 716 instances divided into 5 groups with up to 900 nodes. The commercial CPLEX solver with the proposed ILP formulation for the CPNCP formulation was used to obtain optimal solutions or upper bounds, when possible. The results obtained using the proposed SVNS with FI-based Local Search are compared with the results of the exact solver CPLEX that uses proposed ILP formulation of the CPNCP. The obtained results show that our SVNS reaches optimal solutions or improves the best feasible solutions provided by CPLEX in significantly shorter CPU times. The SVNS also quickly returns its best solutions for instances that were out of reach for CPLEX due to memory limits.
- In order to analyze the effects of using two different approaches in service network planning, we compare the objective function values of CPNCP for given p and q and the corresponding values of PNCP when the number of centers to be established is $p + q$. Our motivation was to investigate two scenarios that arise as a result of using PNCP and CPNCP models in real-life situations.
- Finally, we investigated the potential of the proposed SVNS to solve the PNCP. For this purpose, we set the number of existing centers to 0 and run the SVNS on the set of PNCP instances from the literature. The obtained results are compared with the results of recent solution approaches for the PNCP, showing that SVNS represents a promising solution method for the PNCP as well.

The remaining part of the paper is organized as follows. Problem definition and ILP mathematical model for the CPNCP are presented in Section 2. A detailed description of the proposed Skewed Variable Neighborhood Search with Fast Interchange is given in Section 3. The computational results are presented and analyzed in Section 4. A summary of the results and some possible research directions are outlined in Section 5.

2. Problem definition and mathematical model

In this section, we provide the description of the conditional p-next center problem and present its mathematical formulation. Let A be the set of all locations: customers, established centers, and candidates for new centers. Let $Q \subset A$ be the set of already established centers, where $|A| = n$ and $|Q| = q$. The distance matrix is denoted as $D = (d(i,j))_{n\times n}$, where $d(i,j)$ represents the distance (time, cost, etc.) between locations i and j. For a given network of q centers, the problem is to find the set P of additional $p = |P|$ centers and allocate each customer to its nearest (primary) center among all $p+q$ established centers. It is assumed that customers travel to their primary center and if they find out about its eventual failure on the spot, they move on to the closest (backup) center from there. Therefore, the objective is to minimize the distance from each customer to its backup center, passing through a primary center.

In other words, the value

$$w(P) = \max_{i\in A}\left\{ \min\{\min_{j\in Q} d(i,j), \min_{j\in P} d(i,j)\} + \min\{\min_{k\in Q, k\neq j'} d(j',k), \min_{k\in P, k\neq j'} d(j',k)\}\right\},$$

$$j' = \operatorname{argmin}\{\min_{j\in Q} d(i,j), \min_{j\in P} d(i,j)\}$$

is to be minimized. The conditional p-next center problem can be formulated as an integer linear program (ILP). For that purpose, the following binary variables are used:

$$y_j = \begin{cases} 1, & \text{if there is a center at location } j, \text{ i.e. if } j \in Q \cup P, \\ 0, & \text{otherwise.} \end{cases}$$

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

It is important to notice that if $x_{ij} = 1$, and the location i is a customer, that means that its primary center is the location j. On the other hand, if the location i is a center, then j is its backup center.

In Albareda-Sambola et al. (2015) authors presented an ILP formulation for the PNCP. In this study, we adapt the PNCP formulation from Albareda-Sambola et al. (2015) into the corresponding model for the CPNCP. Using the notation introduced above, the CPNCP can be formulated as the following integer linear program:

$$\min\ w \tag{1}$$

$$\sum_{j\in A} y_j = p + q, \tag{2}$$

$$\sum_{j\in A,\, j\neq i} x_{ij} = 1, \quad i \in A, \tag{3}$$

$$x_{ij} \leq y_j, \quad i, j \in A,\, i \neq j, \tag{4}$$

$$y_j + \sum_{k\in A,\, d(i,k)>d(i,j)} x_{ik} \leq 1, \quad i, j \in A,\, i \neq j, \tag{5}$$

$$w \geq \sum_{k\in A, k\neq j} d(j,k)x_{jk}, \quad j \in A, \tag{6}$$

$$w \geq d(i,j)(x_{ij} - y_i) + \sum_{k\in A, k\neq j} d(j,k)x_{jk}, \quad i, j \in A, i \neq j, \tag{7}$$

$$x_{ij} \in \{0,1\}, \quad i, j \in A, i \neq j, \tag{8}$$

$$y_j \in \{0,1\}, \quad j \in A \setminus Q, \tag{9}$$

$$y_j = 1, \quad j \in Q. \tag{10}$$

The objective (1) is to minimize the value of the continuous variable w, which represents the maximal travel distance from a customer to his backup center.

In Constraint (2), the total number of established centers is set to $p + q$. Constraint (3) ensures that each user is allocated to exactly one primary center and that there is exactly one backup center for each primary center. Customers cannot be allocated to a location without an established center, which is defined in Constraint (4). Constraint (5) indicates that each customer is allocated to its nearest established primary center.

Constraints (6) and (7) are necessary to set w to the correct value in both possible cases: when the location is either a user or a center. If the location is a center, it serves itself as the primary center (the distance is equal to 0), and therefore, a two-leg trip is avoided. Condition (6) ensures that the value of w is greater than or equal to the largest distance from any location j to its nearest center different from j. Since the variable x_{jk} has different meanings depending on the role of location j (user or center), the sum on the right-hand side of inequality (6) is interpreted differently. If j is a user, this sum represents the distance from j to its primary center k. On the other hand, if j is a center, the distance to itself (its primary center) is 0, and therefore this sum represents the distance to its backup center.

In case of a failure, all users with the same primary center proceed to the same backup center. In this case, the total travel distance is equal to the distance from a user to its primary center increased by the distance from the primary to the backup center, which is specified in Constraint (7). Indeed, if i is a user with its primary center j ($x_{ij} = 1$, $y_i = 0$), the value of the variable w will be greater than or equal to the sum of $d(i,j)(x_{ij} - y_i) = d(i,j)$ (distance from i to its primary center) and the distance from the primary center j to the backup center k. In all other cases, where i is a center ($y_i = 1$), the value of $x_{ij} - y_i$ will be less than or equal to 0, which implies that Constraint (7) is weaker than Constraint (6), and consequently, only the distance from center i to its backup center is considered.

Constraints (8)–(9) reflect the binary nature of the variables. The indicators for the q already established centers are set to 1 in Constraint (10).

3. Variable neighborhood search algorithm for the CPNCP

The algorithm proposed in this paper for solving the CPNCP is based on the Variable Neighborhood Search method introduced by Hansen and Mladenović in 1997 (Hansen and Mladenović, 1997). This metaheuristic provides a general solution strategy that can be applied to various continuous and discrete optimization problems. When designing a VNS algorithm, a set of different neighborhood structures must be defined. In order to keep the search process away from the local optimum traps, VNS involves a systematic change of the neighborhoods in which the local search is performed. This is motivated by three facts: a local optimum in a certain neighborhood is not necessarily a local optimum in some other neighborhood, the global optimum is the local optimum for each neighborhood, and the points of the local optimum with respect to different neighborhoods are close to each other (Hansen and Mladenović, 1997).

Different variants of the VNS method have been proposed in the literature. Variable Neighborhood Descent (VND) is a deterministic VNS variant that changes the set of neighborhoods in the predefined order within the local search step. In addition to the local search, the basic VNS variant (BVNS) includes a shaking step as a stochastic procedure that enables visiting solutions in various parts of the search space. In the shaking step, a new solution is randomly chosen from some predefined neighborhood, and this solution is used as the initial one for some local search procedure. If the solution obtained by the local search is better than the best obtained solution so far, the algorithm moves to this new solution and the search is continued from there (Move or Not step). Otherwise, the larger neighborhood is explored in the shaking step. In this way, diversification is achieved through the shaking step, and the local search intensifies the search. The combination of these two steps reduces the risk of the algorithm getting stuck in a local optimum that is not the global one. In the Variable the Neighborhood Decomposition search (VNDS), the local search works on a subproblem that is obtained by fixing certain number of attributes, while the others have been changed. The size of a subproblem changes systematically in each iteration of VNDS according to the size of the neighborhood used in the shaking step. If a VNS-based method is used in the local search step, a two-level VNS approach is obtained. A detailed description of the different VNS variants is beyond the scope of this paper and can be found in Brimberg et al. (2023) and Hansen et al. (2010).

In situations where a VNS algorithm is trapped in a valley of the current best solution, shaking and local search may not be sufficient to ensure the escape from the local optimum trap. In such situations, it is necessary to increase a diversification of the search process to some promising regions that are far away from the local optimum valley. To achieve this, it makes sense not only to allow the moves that improve the current best solution, but also to allow the algorithm to move to a slightly worse solution, if these two solutions are enough away from each other. The inclusion of this modified criterion for the solution acceptance leads to a variant of the VNS method called Skewed VNS (SVNS). In SVNS, the acceptance criterion is defined as follows. If a new solution y is better than the current solution x, it is always accepted. A new solution y that is worse than x (i.e., has the greater objective function value $f(y) > f(x)$) is be accepted if $f(y) \leq f(x) + \alpha\rho(x,y)$, where $\rho(\cdot,\cdot)$ is a metric used to define the distance and α is the parameter of the method that controls the acceptance criterion. The SVNS showed to be successful for solving various optimization problems in the literature. The SVNS outperformed other VNS variants in cases where it was important to enhance the exploration of far away valleys, see Brimberg et al. (2019, 2015), Macedo et al. (2015), Mrkela and Stanimirović (2022), Mladenović et al. (2022), etc.

Various problems based on the p-center and its variants are solved by VNS-based methods: p-center in Mladenović et al. (2020, 2003), the PNCP in Ristić et al. (2023a, 2021), Sánchez-Oro et al. (2022), and Tasić (2024), the unconditional and conditional vertex p-center in Iravan et al. (2016), the capacitated vertex p-center in Quevedo-Orozco and Ríos-Mercado (2015), etc. The above observations motivated us to apply a VNS-based approach for solving CPNCP. We have conducted a series of preliminary computational experiments with different variants of VNS, and the SVNS showed to be the most successful in terms of solution quality and running times when solving CPNCP. We believe that the increased impact of diversification in SVNS, which was achieved by allowing the algorithm to move to a slightly worse solution that is not too close to the incumbent one, is the reason why SVNS showed to be superior than other VNS variants in the case of CPNCP.

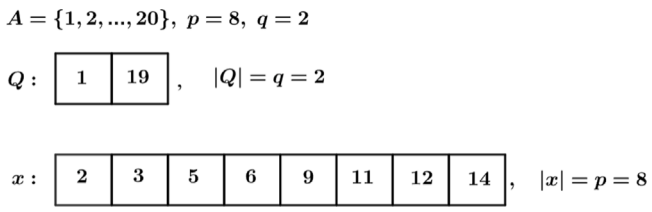


Fig. 2. Solution encoding.

3.1. The proposed SVNS implementation for solving the CPNCP

When designing a metaheuristic for solving a given problem, the encoding of the solution plays an important role. In the proposed SVNS for the CPNCP, an integer encoding of the solutions is used. More precisely, to each location from the set A, a unique integer index from the set $\{1, 2, \dots, n\}$ is assigned. The solution of the CPNCP is represented by a vector of p integers, where each integer represents the index of a location with an established center. Let Q be the vector containing the indices of q locations with already established centers. The set of feasible solutions S is defined as the set of all vectors x of length p containing the indices of the locations for the new centers that are different from the indices from Q, i.e., $1 \le x(i) \le n$ and $x(i) \ne Q(j), i = 1, 2, \dots, p, j = 1, 2, \dots, q$.

Let us consider an example of a problem with $n = 20$ locations. Let the indices of two ($q = 2$) already opened centers be 1 and 19. The goal is to determine $p = 8$ additional centers among the remaining $n-q = 18$ locations in such a way that the value of the objective function value is minimized. In this case, the set of feasible SVNS solutions in this case is $S = \{x : |x| = 8, 1 \le x(i) \le 20, x(i) \ne 1, x(i) \ne 19, i = 1, \dots, p\}$. The cardinality of the set S is $\binom{18}{8} = 43.758$. One of the solutions $x \in S$ is shown in Fig. 2.

For $a, b \in A$, let $d(a, b)$ represent the distance (delivery costs, travel time) between the locations a and b obtained from the distance matrix D. The objective function value $f(x)$ for a feasible solution $x \in S$ is calculated as follows.

- For $a \in A$, we choose the closest center c from the set $x \cup Q$ such that $d(a, c)$ is minimal. If a is a center, then $a = c$ and $d(c, c) = 0$. Let us denote this minimum distance as d_1.
- For the chosen primary center c, we choose its backup center b from the set $(x \cup Q) \setminus \{c\}$ such that $d(c, b)$ is minimal. Let us denote this minimum distance as d_2.
- Finally, the value of $f(x)$ is calculated as $\max_{a \in A}\{d_1 + d_2\}$.

The proposed SVNS uses a set of neighborhoods $N_k(x)$, $k = 1, 2, \dots, k_{max}$ of a solution x that contains all solutions x' that differ from x in exactly k centers. In other words: $N_k(x) = \{x' \in S : \rho(x, x') = k\}$, where $\rho(x, x')$ is the Hamming distance between the integer arrays x and x' of length p. More precisely, $\rho(x, x')$ is equal to p minus the number of common elements of the vectors x and x'.

The main steps of the proposed SVNS metaheuristic for the CPNCP are given by the Algorithm 1. An initial feasible solution is chosen as an arbitrary point from S.

The shaking step $Shaking()$ generates the new solution $x_{shake} \in N_k(x)$ as follows. For a given k, we randomly choose k centers from the current solution x to be closed and k random locations from $A \setminus (x \cup Q)$ for opening centers. The resulting solution is denoted as x_{shake}.

Starting from the solution x_{shake}, the local search procedure $LS()$ explores the neighborhood $N_1(x_{shake})$ completely to find a local minimum in this neighborhood. The cardinality of the $N_1(x_{shake})$ neighborhood is $p \cdot (n-q-p)$. If the obtained local minimum represents an improvement of the objective function value, the process is restarted from that solution. The local search stops when no further improvement of the objective function value is possible. The detailed description of the proposed local search procedure is given in Section 3.2.

Finally, the solution x'' obtained by the local search is compared with the best solution found x^*, in order to decide whether it should be accepted or not. If $f(x'') < f(x^*)$, the current best solution is updated, the algorithm moves to x'' and the search is continued in the first neighborhood N_1 of x''. If the solution x'' found by $LS()$ has a lower quality than the best known solution x^*, but fulfills the condition outlined in step 12 of the Algorithm 1, we take x'' as the current solution and continue the search in $N_1(x'')$ without updating the best solution. The condition $f(x'') < F + \alpha\rho(x, x'')$ is a relaxed rule that allows the algorithm to visit a solution x'' that is worse than the incumbent solution x, but only if x'' is sufficiently different from x. A metric for difference is measured by previously defined distance function ρ. The parameter $\alpha > 0$ controls the level of diversification: the search should be recentered to solution x'' if x'' is slightly worse than x and the distance $\rho(x'', x)$ is large enough. If neither of these two conditions is satisfied, the search is continued in the next neighborhood N_{k+1} of the current solution.

The described steps are repeated until a stopping criterion is satisfied. In this SVNS implementation, the maximal number of VNS iterations is used as a stopping criterion. We count a VNS iteration by the execution of the shaking step followed by the local search step.

Algorithm 1 SVNS algorithm for the CPNCP

1: Initialization: select the set of neighborhoods N_k, $k = 1, 2, \dots, k_{max}$
2: Randomly choose an arbitrary initial point $x^* \in S$ and set $f^* \leftarrow f(x^*)$
3: Set $x \leftarrow x^*$, $F \leftarrow f^*$
4: **while** the stopping criterion is not met **do**
5: $k \leftarrow 1$
6: **while** $k \le k_{max}$ **do**
7: $x_{shake} \leftarrow Shaking(x, k)$ ▷ Shaking step
8: $x'' \leftarrow LS(x_{shake})$ ▷ Local Search step
9: **if** $f(x'') < f^*$ **then**
10: $x^* \leftarrow x'', f^* \leftarrow f(x'')$
11: **end if**
12: **if** $f(x'') < F + \alpha\rho(x, x'')$ **then** ▷ Move or Not step
13: $x \leftarrow x''$, $F \leftarrow f(x'')$, $k \leftarrow 1$
14: **else**
15: $k \leftarrow k + 1$
16: **end if**
17: **end while**
18: **end while**

3.2. Local search based on the fast interchange heuristic

Exploring the entire $N_1(x)$ neighborhood of the solution x means that all solutions x' that satisfy $\rho(x, x') = 1$ must be evaluated. In order to speed up the algorithm and avoid swapping users and centers that does not lead to an improvement of the objective function value, we use the Fast Interchange (FI) heuristic to implement the $LS()$ procedure. Similar procedures were mentioned in Whitaker (1983) and Mladenović et al. (2003). The pseudocode for the $LS()$ procedure is given in Algorithm 2.

The implementation of local search for the CPNCP is based on two main steps: determining the best choice of a location *in* to enter the solution, and finding the best choice of a center *out* to be removed from the solution. Let the critical user be the one who travels the longest distance to its backup center. To reduce the value of the objective function, the distance of the critical user i^* to its backup center must be reduced. This can be achieved if: (i) *in* becomes the new primary center for i^*, (ii) *in* becomes the new backup center for i^*, (iii) *in* becomes the new backup center for some other center that is as far away from i^* as its primary center.

Algorithm 2 Local search for the CPNCP

```
1: procedure LS(solution x)
2:     Construct the corresponding arrays c1 and c2 for x
3:     f* ← f(x)
4:     Find the critical user i*
5:     while true do
6:         w* ← ∞
7:         for every location in ∉ (x ∪ Q) do
8:             if in reduces total distance traveled by i* then
9:                 out ← MoveEvaluation(in, x, c1, c2)
10:                F ← f((x ∪ {in}) \ {out})
11:                if F < w* then
12:                    w* ← F, in* ← in, out* ← out
13:                end if
14:            end if
15:        end for
16:        if f* ≤ w* then
17:            break
18:        else
19:            f* ← w*
20:            Update(x, in*, out*, c1, c2)
21:            Find the new critical user i*
22:        end if
23:    end while
24: end procedure
```

Each time we want to add a new center to the solution, the $LS()$ procedure does not check all possible locations from $A \setminus (x \cup Q)$, but only the locations that can reduce the distance between the critical user to its backup center. In order to make an adequate choice of the new center in, the procedure $MoveEvaluation()$ determines the best center out to be removed from the solution. The detailed steps of this procedure are described in Section 3.2.1.

Replacing some centers in the solution generally requires reassigning all users to their (possibly new) primary and backup centers. To avoid doing this from scratch every time, we use 2 auxiliary vectors c_1 and c_2 corresponding to the current solution x. The vectors c_1 and c_2 are defined as follows:

$c_1(i)$ – the center closest to the location i (among all $p + q$ established centers),
$c_2(i)$ – the second closest center to the location i, $i = 1, 2, \ldots, n$.

The value $c_1(i)$ represents the primary center for the customer i, and its backup center is given with $c_1(c_1(i))$. If the center $c_1(i)$ fails, the customer i proceeds to the center $c_1(c_1(i))$. In the general case, $c_1(c_1(i))$ is not identical to $c_2(i)$. If there is an established center at location i, then $c_1(i)$ is a backup center for the center i. For example, let us consider a network with $n = 10$ nodes with a previously opened center at location 3 and newly opened centers at nodes 5 and 8, as shown in Fig. 3. Note that for location 2, the primary center is 3, i.e., $c_1(2) = 3$. The backup center for location 2 is the center 5 that is closest to its primary center 3, i.e. $c_1(c_1(2)) = 5$. On the other hand, the second closest center to location 2 is center 8, i.e. $c_2(2) = 8$.

Finally, the exchange of a chosen pair (in, out) is performed by the procedure *Update()*, which also makes the necessary changes to the vectors c_1 and c_2 so that they correspond to the updated solution $(x \cup \{in\}) \setminus \{out\}$. The pseudocode for the *Update()* procedure is given in Algorithm 3.

3.2.1. Move evaluation procedure

Previous successful applications of the Move Evaluation algorithm (such as the one from Mladenović et al. (2003) for PCP) gave us some general guidelines on how to design and implement our Move Evaluation procedure (MEP) for the CPNCP. Comparing with the MEP procedure for PCP, one can notice that each segment of the implemented MEP for CPNCP has been modified and upgraded with the new elements to make it work properly for CPNCP. For example, since we are considering a problem involving backup centers, not even the primary centers are determined in the same way as for PCP. Adding or excluding a center from a solution can result in many more cases to consider. This is reflected in a more complex calculation of the elements of the arrays r and z compared to the PCP case. The elements of the

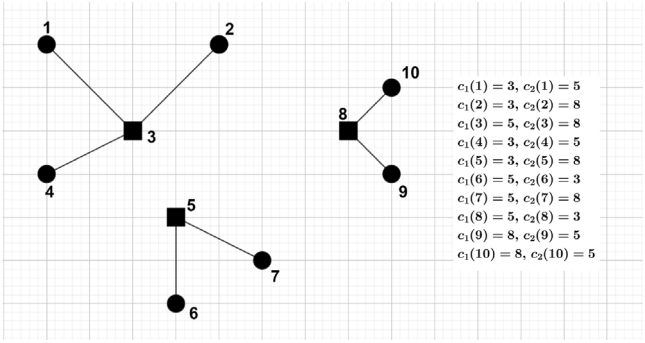


Fig. 3. Vectors c_1 and c_2 for a solution of the CPNCP for $n = 10$, $q = 1, p = 2, Q = \{3\}$, new centers are located at nodes 5 and 8.

Algorithm 3 Update procedure for the CPNCP

```
1: procedure UPDATE(x, in, out, c1, c2)
2:     x ← (x ∪ {in}) \ {out}
3:     for every center j ∈ x do
4:         Recalculate c1(j) and c2(j)
5:     end for
6:     for every user i ∈ A \ (x ∪ Q) do
7:         if c1(i) = out then
8:             if d(i, in) < d(i, c2(i)) or (d(i, in) = d(i, c2(i)) and d(in, c1(in)) < d(c2(i), c1(c2(i)))) then
9:                 c1(i) ← in
10:            else
11:                c1(i) ← c2(i)
12:                Find the second closest center center_2 to the location i
13:                c2(i) ← center_2
14:            end if
15:        else
16:            if d(i, in) < d(i, c1(i)) or (d(i, in) = d(i, c1(i)) and d(in, c1(in)) < d(c1(i), c1(c1(i)))) then
17:                c2(i) ← c1(i)
18:                c1(i) ← in
19:            else
20:                if d(i, in) < d(i, c2(i)) or (d(i, in) = d(i, c2(i)) and d(in, c1(in)) < d(c2(i), c1(c2(i)))) then
21:                    c2(i) ← in
22:                else
23:                    if c2(i) = out then
24:                        Find the second closest center center_2 to the location i
25:                        c2(i) ← center_2
26:                    end if
27:                end if
28:            end if
29:        end if
30:    end for
31: end procedure
```

array g must also be determined with some additional checks. In the rest of the subsection, the MEP proposed for the CPNCP is explained in detail.

For the given solution x and a candidate location in, the MEP determines the center $out \in x$ that is the best choice to be removed from x when a center is opened at location in. In other words, $f((x \cup \{in\}) \setminus \{out\})$ should be as minimal as possible. The MEP procedure also calculates the objective function value $f((x \cup \{in\}) \setminus \{out\})$, but it does not perform the exchange of the centers in and out.

The input for MEP contains:

- location $in \in A \setminus (x \cup Q)$, which should become part of the solution,
- arrays c_1 and c_2 that correspond to the solution x.

The output of the procedure includes:

- the center out, which should be removed from the solution x,
- the evaluation of the swap, i.e. the value $f((x \cup \{in\}) \setminus \{out\})$.

Adding the location in to the solution can have the following two effects on the current solution: (i) There exists a customer i (possibly more than one) for which the center in is closer than its current primary center, i.e., $d(i, in) < d(i, c_1(i))$ so that customer i should be reallocated to the center in. Consequently, the backup center $c_1(c_1(i))$ should also be updated for such a customer. (ii) There is an established center j, for which the location in is closer to j than the current backup center for j. In this case, in becomes the new backup center for j, as well as for all customers allocated to j.

On the other hand, removing a center out from the solution affects all users whose primary or backup center was out. Therefore, for all users whose primary center was out, a new primary and a new backup center must be determined, and for all users whose backup center was out, the backup center must be updated.

Motivated by the possible effects that the swapping of the elements in and out could cause, and to implement MEP efficiently, we use two auxiliary vectors: the vector r of a length $p+q$ and the vector z of length p, which are defined as follows:

- $r(j) = \max_{i \in S_j} \{d(i, j) + d(j, k)\}$, $j = 1, 2, \ldots, p+q$, where S_j denotes the set of all users assigned to the center $j \in x \cup Q$ and k is the backup center for the center j, i.e., $k = c_1(j)$.
- Assuming that the center $j = x(l)$, $l = 1, 2, \ldots, p$ is removed from the current solution, for all the users $i \in S_j$ allocated to the center j, the total travel distance, i.e. the distance to the new backup center through the new primary center, should be calculated. The element $z(j)$ is set to the longest of these distances: $z(j) = \max_{i \in S_j} \{d(i, new\_primary) + d(new\_primary, new\_backup)\}$, $j = 1, 2, \ldots, p$.

Each step of MEP is described in detail by the pseudocode in Algorithm 4. Considering the previously described effects caused by removing the center out and/or adding the center in to the solution, we can divide all users into three main groups: (i) users for whom the new center in is closer than their primary center, (ii) users who are not affected by the removal of center out but whose backup center is changed by opening in, and (iii) users whose primary or backup center is out. In each of these cases, the vectors r, z, c_1, and c_2 are suitably modified by updating only the information related to the given changes in the solution. It is possible to calculate all values $z(j), r(j), j = 1, \ldots, n$ by considering each user only once. In addition, a value $f((x \cup \{in\}) \setminus \{out\})$ is determined, representing a candidate for the objective function value after the swap.

The MEP procedure considers each candidate to be removed $out' \in x$ in an effective way, and determines the best choice out to be removed from the current solution as the one that minimizes the objective function value. The complexity of the MEP procedure is $O(n - p - q)$ and it is called $n - p - q$ times. Updating the information about the changes made in a solution requires $O(n(p + q))$ operations, which is the complexity of the Update procedure. Therefore, the worst-time complexity of the FI procedure used in the Local search is $O((n-p-q)^2)$.

Algorithm 4 Move Evaluation procedure for the CPNCP

procedure Move Evaluation(in, x, c_1, c_2)
  Initialization:
  $f' \leftarrow 0$,
  $r(x(j)) \leftarrow \min\{d(x(j), in),\ d(x(j), c_1(x(j)))\}$,
  $z(c_1(x(j))) \leftarrow \max\left\{ z(c_1(x(j))),\ \min\{d(x(j), in),\ d(x(j), c_2(x(j)))\} \right\}$, $j = 1, 2, \ldots, p+q$
  for every user $i \in A \setminus (x \cup Q)$ **do**
    if $i \neq in$ and $(d(i, in) < d(i, c_1(i))$ or $(d(i, in) = d(i, c_1(i))$ and $d(in, c_1(in)) < d(c_1(i), c_1(c_1(i)))))$ **then**
      $f' \leftarrow \max\{f',\ d(i, in) + d(in, c_1(in))\}$
      if $f' = d(i, in) + d(in, c_1(in))$ **then**
        $critical\_user \leftarrow i$, $backup\_critical \leftarrow c_1(in)$
      end if
    else
      ▷ Update the values $r(c_1(i))$
      $r(c_1(i)) \leftarrow \max\left\{ r(c_1(i)),\ \min\{d(i, c_1(i)) + d(c_1(i), c_1(c_1(i))), d(i, c_1(i)) + d(c_1(i), in)\} \right\}$, $i \neq in$
      $r(c_1(in)) \leftarrow \max\{r(c_1(in)),\ d(in, c_1(in))\}$
      ▷ Update the values $z(c_1(i))$
      Determine $new\_primary\_i$ between in and $c_2(i)$ and new_backup_i for the user i
      $z(c_1(i)) \leftarrow \max\{z(c_1(i)),\ d(i, new\_primary\_i) + d(new\_primary\_i, new\_backup\_i),\}$ $i \neq in$
      $z(c_1(in)) \leftarrow \max\{z(c_1(in)),\ d(in, c_2(in))\}$
      ▷ Update the values $z(c_1(c_1(i)))$
      Determine new_backup_i for the user i between in and $c_2(c_1(i))$
      $z(c_1(c_1(i))) \leftarrow \max\{z(c_1(c_1(i))),\ d(i, c_1(i)) + d(c_1(i),\ new\_backup\_i)\}$
    end if
  end for
  for every center $x(j), j = 1, 2, \ldots p$ **do** ▷ Update the values $z(x(j))$
    Determine $new\_primary\_j$ between in and $c_1(j)$ and new_backup_j
    $z(x(j)) \leftarrow \max\{z(x(j)),\ d(x(j), new\_primary\_j) + d(new\_primary_j, new\_backup\_j)\}$
  end for
  Find $g_1 \leftarrow \max\{r(x(l)) \mid l = 1, 2, \ldots, p+q\}$, let the l^* be the corresponding index
  Find $g_2 \leftarrow \max_{l \neq l^*}\{r(x(l)) \mid l = 1, 2, \ldots, p+q\}$
  for $l = 1, 2, \ldots, p$ **do**
    if $x(l) = backup\_critical$ **then**
      $f' \leftarrow f' - d(in, backup\_critical) + d(in, c_2(in))$ ▷ Update f'
    end if
    if $l \neq l^*$ **then**
      $g(l) \leftarrow \max\{f', z(x(l)), g_1\}$
    else
      $g(l) \leftarrow \max\{f', z(x(l)), g_2\}$
    end if
  end for
  Find: $f \leftarrow \min\{g(l) \mid l = 1, 2, \ldots, p\}$
  Find the center to be eliminated: $out \leftarrow x(l^{**})$, where l^{**} is the index of the found minimum
end procedure

4. Computational experiments

The computational experiments were performed on a desktop computer with Intel Core i7-11700 3.6 GHz processor and 16 GB of RAM in a 64 bit Windows 10 environment. The ILP formulation presented in Section 2 was solved with the IBM ILOG CPLEX 22.1.0 solver. CPLEX uses 16 logical processors and stops either when it finds an optimal solution or it reaches the maximum time of 7200 s per thread. The SVNS proposed in this paper was implemented in the C++ language and the execution was carried out on a single core.

4.1. Dataset

We have used 716 instances divided into 5 groups, all derived from the instances for the p-next center problem used in the papers (Londe et al., 2021; Mousavi, 2023; Ristić et al., 2021; Tasić, 2024). The small size instances consider $20 \leq n \leq 50$, the medium size instances $50 < n \leq 200$, the large size instances $200 < n \leq 400$, and the extra large size instances $800 \leq n \leq 900$. The fifth group contains instances with $100 \leq n \leq 900$, which are derived from Additional group of pmed instances from the paper (Ristić et al., 2021). In order to generate test instances for solving CPNCP we used two different approaches.

The first approach is used to generate instances with $20 \leq n \leq 100$. It is based on the approach used in Iravan et al. (2016) where the authors generated test instances for CPCP by adapting the instances of p-center problem. For each PNCP instance, we solved the q-next problem to optimality by using the ILP formulation from Albareda-Sambola et al. (2015). We chose the q established centers from the obtained optimal solution to be the q existing centers in the CPNCP. For example, for the (15,5)-next center problem, the existing 5 centers are the optimal solution of the 5-next center problem. As in Iravan et al. (2016), the values q are varied with a step of 5 in each considered instance.

The second approach, which is also proposed in Iravan et al. (2016), is used for PNCP instances with $n > 100$. Note that no optimal solution is known for these instances. Therefore, the q exiting centers for CPNCP are randomly chosen from the PNCP solutions obtained by the heuristic in Tasić (2024). It should be noted that these solutions are not necessarily the optimal ones. Assuming that the complexity of the instance decreases with the increase of q, we limited the values of q to vary from 2 to at most 10% of the value of $p+q$.

In the paper (Ristić et al., 2021), the authors presented an Additional set of pmed test instances for the PNCP with up to 900 nodes. These instances were also adapted according to the second approach. In this way, the fifth set of CPNCP instances, denoted as Additional pmed dataset, is obtained.

The following notation is used for all instances considered in this study. The instance name pmedX_n_(p+q)_q contains information about the total number of nodes (n), the total number of centers ($p+q$), and the number of already established centers (q). For example, the instance pmed1_90_20_5 is generated from the instance pmed1_90_20 for the PNCP. It has $n = 90$ nodes, $q = 5$ existing centers, and the total number of centers should be $p+q = 20$. Therefore, the aim is to open additional $p = 20 - 5 = 15$ centers.

4.2. Parameter tuning

One of the main problems in the implementation of a metaheuristic methods is the estimation of the parameter values that should be used when solving instances or groups of data sets with different properties. On the other hand, the efficiency and effectiveness of a metaheuristic method are reflected in its robustness with respect to the parameter values that must be specified in advance. The most important parameter of SVNS is the parameter α, which affects the quality and diversification of the solutions. Our aim was to evaluate the efficiency of the proposed SVNS without fine adjustments of the parameters to the specific instances or dataset groups. Therefore, a series of preliminary computational experiments were performed to determine the best value of the parameter α to be used for all further tests.

Table 1
Comparison on different values of parameter α.

Instance group	$\#Inst$	α	$best_{avg}$	$\#best_{avg}$	t_best_{avg}
$n \leq 50$	15	0	**102.73**	19.73	0.00
		0.1	**102.73**	**20**	0.00
		0.2	**102.73**	**20**	**0.00**
		0.3	**102.73**	**20**	0.00
		0.5	**102.73**	**20**	0.00
$50 < n \leq 100$	40	0	**96.78**	17.78	0.18
		0.1	**96.78**	**20**	**0.10**
		0.2	**96.78**	19.98	0.10
		0.3	**96.78**	19.65	0.12
		0.5	**96.78**	19.28	0.16
$100 < n \leq 200$	34	0	61.85	16.18	0.93
		0.1	**61.76**	**19.5**	**0.53**
		0.2	**61.76**	**19.56**	0.64
		0.3	**61.76**	18.35	0.70
		0.5	**61.76**	17.47	0.75
$200 < n \leq 400$	26	0	44.15	12.04	3.95
		0.1	**43.77**	**18.23**	**2.27**
		0.2	43.88	17.12	2.40
		0.3	44.12	15.77	2.61
		0.5	44.27	15.61	3.14

The parameter tuning tests were performed with a subset of 115 randomly selected problem instances from the small, medium and large instance sets, i.e. about 20% of the total number of instances are used for this purpose. The set of values considered for the parameter α was $\{0, 0.1, 0.2, 0.3, 0.5\}$. Note that in the case of $\alpha = 0$, the SVNS algorithm is reduced to the classical BVNS. For each value of α, the algorithm was executed 20 times for each instance. The stopping criteria was the maximum number of SVSN iterations. In this way, we ensure that the SVNS requires similar total execution times for the same instance when different values of α are used. For each instance, we store the best solution value obtained ($best$), the number of times the algorithm obtained the best value ($\#best$), and the average total execution time required to reach the best solution for the first time (t_best).

Table 1 shows the summarized average results obtained for the total number of $\#Inst$ instances in each group. The detailed results of the parameter tuning test can be found in Appendix C (Tasić et al., 2024c). For each instance group, the value of α that led to the best average objective function value is shown in bold. Table 1 shows that $\alpha = 0$ led to the worst SVNS performance, while the algorithm performed slightly better for $\alpha = 0.5$. For instances with $n \leq 50$, only $\alpha = 0$ did not lead SVNS algorithm to the best solutions in all 20 runs in all cases. For instances with $50 < n \leq 100$, only $\alpha = 0.1$ provided the best solutions in all 20 runs in all cases. Furthermore, for $\alpha = 0.1$, the SVNS required the shortest average time to reach these solutions for the first time. Considering $best_{avg}$ and t_best_{avg} the worst SVNS performance was for $\alpha = 0$, followed by $\alpha = 0.5$. For the set of instances with $100 < n \leq 200$, only $\alpha = 0$ did not lead SVNS to the best solutions in all cases. Looking at the $\#best_{avg}$ and t_best_{avg} columns, $\alpha = 0$ performed the worst, $\alpha = 0.5$ was slightly better, while $\alpha = 0.1$ and $\alpha = 0.2$ gave the best results. Finally, for $200 < n \leq 400$ the best SVNS results considering all three measures were obtained with $\alpha = 0.1$, and the worst with $\alpha = 0$, followed by $\alpha = 0.5$. The worst performance of the algorithm at $\alpha = 0$ clearly shows that the SVNS is more suitable for solving the CPNCP than the BVNS.

We also performed a statistical analysis to gain a better insight into the results obtained. Following the recommendations of Demšar (2006), the Friedman test was performed. This is a non-parametric test based on a ranking of the performance of the SVNS method for all five different values of α for each instance. The value that results in the best SVNS performance for a given instance is ranked as 1, the second best value is ranked as 2, and so on. Since we have three measures to compare

Table 2
The average ranks for Friedman test.

α	0	0.1	0.2	0.3	0.5
$rank_{avg}$	4.32	2.27	2.3	2.64	3.46

(*best*, *#best*, and *t_best*), the first ranking criterion is the value of the objective function (*best*). If two or more method variants obtained the same value of *best*, the method with the higher value of *#best* is given a better ranking. If the values of *best* and *#best* are the same, the method that reached the result faster for the first time (according to *t_best*) gets the better ranking.

The tested null hypothesis is: There is no significant difference between the compared methods with different α values, while the alternative hypothesis is: At least one α value provides a significantly different result than the others. The recommended test significance is $p = 0.05$. For each α value, the average value of the assigned ranks is calculated and shown in the Table 2.

For $N = 115$ instances and $k = 5$ different α values, we calculated $\chi^2 = 138$ using the formula

$$\chi^2 = \frac{12N}{k(k+1)}\left(\sum_{i=1}^{k} rank_{avg_i}^2 - \frac{k(k+1)^2}{4}\right). \quad (11)$$

To reject the null hypothesis, this value must be equal to or greater than the critical value for the χ^2 distribution with $df = k - 1 = 4$ degrees of freedom at the significance level 0.05, which is 9.49 (Sheshkin, 2000). Since $\chi^2 = 138 > 9.49$, this means that the null hypothesis should be rejected and that there are significant differences between the compared values of α.

To determine which difference(s) is/are significant, we used the Nemenyi test. According to Nemenyi, two methods lead to significantly different performances if their average ranks differ by at least $CD = q\sqrt{\frac{k(k+1)}{6N}}$, where q stands for the critical value of the Nemenyi test. In our case, $q = 2.728$ (Demšar, 2006). We calculated the value $CD = 0.5688$ and came to the conclusion that SVNS with $\alpha = 0$ provides significantly worse results than $\alpha = 0.5$. Furthermore, $\alpha = 0.5$ performs significantly worse than $\alpha = 0.1$, $\alpha = 0.2$, and $\alpha = 0.3$. There are no significant differences between the results obtained with the three middle values $0.1, 0.2$, and 0.3. Since SVNS with $\alpha = 0.1$ has the lowest average rank, by taking into account the data from Table 1, we decided to set the value of α to 0.1 in our final computational experiments.

4.3. CPNCP results

In this subsection, we present summarized results of the CPLEX solver and the proposed SVNS algorithm for the CPNCP. Detailed results of both CPLEX and SVNS for all considered CPNCP instances are presented in Appendix A (Tasić et al., 2024a).

In our computational experiments, SVNS was executed 20 times on each test instance, starting from a different random initial solution. The maximum number of SVNS iterations was used as a termination criterion: 500 for the set of small size instances and 5000 for all other sets. The value of the parameter α was set to 0.1 for all dataset groups. We set $k_{\max} = 5$ for all instances except for the Additional pmed instances with $p+q = 5$, for which we used $k_{\max} = 3$ for obvious reasons.

Table 3 contains the average results of CPLEX and SVNS for all 716 pmed instances of CPNCP. For a summarized presentation, the instances are grouped according to the number of nodes n. For example, five CPNCP instances with $n = 20$ nodes derived from pmed1–pmed5 form one group. The average results shown in Table 3 are calculated over the results obtained for the instances of the same group.

Each row of Table 3 contains: the name of the pmed instance group, followed by the total number of locations - n, the interval for the number of new centers to be opened - p, the interval for the number of existing centers - q, and the number of instances in the group - *#inst*.

In the next four columns of Table 3, the CPLEX results are presented as average values for all instances in the group. Column *#opt* contains the number of instances in the group for which the optimal solution was obtained. For some instances, the CPLEX solver provided only a feasible solution within the given execution time, but failed to determine the optimal solution. The number of such instances is given in $\#feas.only$ column. For a certain number of instances, the CPLEX solver could not provide even a feasible solution due to the "Out of memory" (OOM) error message. The number of such instances is reported in the *#oom* column for each group. The last column regarding the CPLEX results, denoted as $avg.t$, contains the average running time used by CPLEX to complete its work (in seconds). The values in this column are marked with "*" to indicate that at least one CPLEX solution in this group is not optimal. Similarly, for groups of instances with *#oom* cases the sign "**" is used to indicate that only the instances with obtained optimal and/or feasible solution were considered. If CPLEX has reached neither an optimal nor a feasible solution for each instance in a group, a mark "–" is written in the corresponding column. Note that the CPLEX running time presented for an instance is the sum of the execution times collected from all threads (the result returned by the built-in function cplex.getTime() of class IloCplex).

The next four columns of Table 3 refer to the SVNS results: *#opt/#bk* - the total number of optimal (if known) or best known solutions from the given group, $avg.t$ - the average total execution time required to reach the best SVNS solution (in seconds), $avg.t_{tot}$ - the average total execution time (in seconds), $agap$ - the average gap (in %), where gap is the average gap between the best objective function value obtained in a single SVNS run, and the best objective function value obtained in 20 SVNS runs. If the SVNS results match the optimal or best known results for all instances of the same group, the values in the *#opt* columns are in bold.

The last column of Table 3 marked as $avg.dev$ contains the average ratio of the objective function values obtained by CPLEX and SVNS. For each instance, we calculate the deviation (in percent) using the formula:

$$\frac{obj_{CPLEX} - obj_{SVNS}}{obj_{SVNS}} \cdot 100\%.$$

Then, for each group of instances, we calculate and present the average deviation values. Please note that the deviation values are positive in cases when the SVNS solutions are better than the feasible ones obtained by CPLEX, and negative otherwise. Naturally, in cases when the objective function values of the SVNS and CPLEX solutions are the same, the deviation is equal to 0%.

The summarized results from Table 3 show that the proposed SVNS has reached all optimal solutions for all small size instances. For the set of medium size instances, SVNS failed to achieve an optimal solution for only one instance, but it improved 4 instances where CPLEX obtained only a feasible solution. The average total SVNS running time on all small and medium size instances was significantly shorter than the CPU time required by CPLEX.

The advantages of SVNS become more obvious as the problem size increases, as shown in the last three sections of Table 3. For the large pmed dataset, SVNS reached all the optimal solutions previously obtained with CPLEX. For the remaining 6 instances, for which CPLEX provided only feasible solution, SVNS failed to improve or reach this upper bound in only one instance. For extra large instances, CPLEX obtained an optimal solution for only 4 instances, while the remaining instances were out reach for CPLEX due to the lack of memory. On the other hand, SVNS reached all 4 optimal solutions in a short CPU time. For the remaining extra large pmed instances, SVNS showed good stability as the average gap values of its solutions are 0% for 171 out of 180 instances in this set.

When considering the Additional pmed dataset, the CPLEX solver obtained an optimal solution for 43 out of 79 instances and provided a feasible solution for 14 instances. For the remaining 22 instances,

Table 3
Average results of the CPLEX solver and the SVNS algorithm on the groups of pmed1–pmed40 and Additional pmed instances.

Inst.	n	p	q	#inst	CPLEX average values				SVNS average values				$avg.dev$ (%)
					$\#opt$	$\#feas.only$	#oom	$avg.t$	$\#opt/\#bk$	$avg.t$	$avg.t_{tot}$	$agap$	
Small size instances													
pmed1–pmed5	20	5	5	5	**5**	0	0	0.07	**5**	0.00	0.01	0	0
	30	5	5	5	**5**	0	0	0.26	**5**	0.00	0.02	0	0
	40	[5,15]	[5,15]	20	**20**	0	0	0.35	**20**	0.00	0.04	0	0
	50	[5,15]	[5,15]	20	**20**	0	0	0.81	**20**	0.01	0.06	<0.01	0
Medium size instances													
pmed1–pmed5	60	[5,25]	[5,10]	25	**25**	0	0	1.13	**25**	0.01	0.83	0	0
	70	[5,25]	[5,10]	25	**25**	0	0	5.81	**25**	0.03	1.03	0	0
	80	[5,25]	[5,10]	25	**25**	0	0	17.57	**25**	0.08	1.23	0.02	0
	90	[5,45]	[5,15]	40	**40**	0	0	25.16	**40**	0.09	1.87	0.02	0
	100	[5,45]	[5,15]	50	**50**	0	0	72.04	49	0.15	2.08	0.10	−0.02
1.03 pmed6–pmed10	150	[10,75]	[5,20]	60	**60**	0	0	1691.89	**60**	0.18	4.78	0.07	0
	200	[10,95]	[5,20]	80	76	4	0	5441.34*	**80**	0.76	8.69	0.17	0.22
Large size instances													
pmed11–pmed12	250	[27,88]	[2,9]	22	18	4	0	13 680.85*	21	2.43	10.91	0.74	0.11
	300	[54,148]	[2,15]	26	**26**	0	0	161.76	**26**	0.14	20.53	0	0
pmed16–pmed17	350	[36,118]	[2,12]	24	22	2	0	6433.69*	**24**	4.24	22.61	1.04	0.79
	400	[72,198]	[2,20]	30	**30**	0	0	615.67	**30**	3.67	55.38	0.45	0
Extra large size instances													
pmed35–pmed37	800	[144,398]	[2,40]	63	4	0	59	2569.40**	**63**	3.65	312.63	0	0**
pmed38–pmed40	850	[81,288]	[2,29]	54	0	0	54	–	**54**	10.28	230.15	0.28	–
	900	[162,448]	[2,40]	63	0	0	63	–	**63**	14.07	428.04	0.06	–
Additional pmed instances													
pmed1–pmed5	100	[3,31]	[2,3]	6	**6**	0	0	2655.45	**6**	0.06	1.79	0.00	0
pmed6–pmed10	200	[3,65]	[2,6]	8	7	1	0	18 406.39*	**8**	0.27	5.31	0.07	0.30
pmed11–pmed15	300	[3,98]	[2,10]	10	7	3	0	25 490.66*	**10**	1.06	12.17	0.32	5.14
pmed16–pmed20	400	[3,131]	[2,13]	11	8	2	1	14 312.88**	**11**	5.30	28.35	1.07	6334.52**
pmed21–pmed25	500	[3,165]	[2,16]	11	6	3	2	19 698.17**	**11**	4.77	46.06	0.77	4.41**
pmed26–pmed30	600	[3,198]	[2,20]	14	6	1	7	13 446.2**	**14**	1.63	67.42	0	0.30**
pmed31-pmed34	700	[3,138]	[2,14]	9	3	3	3	16 044.66**	**9**	5.78	51.88	0.12	14.52**
pmed35–pmed37	800	[3,78]	[2,8]	5	0	1	4	25 111.6**	**5**	1.77	47.15	0	0**
pmed38–pmed40	900	[3,88]	[2,9]	5	0	0	5	–	**5**	21.66	54.93	2.38	–

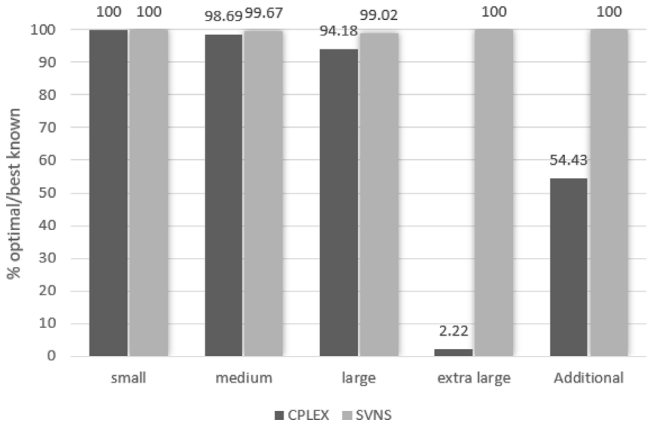


Fig. 4. Success rate (in %) of CPLEX and SVNS in reaching the optimal/best known value.

CPLEX could not even provide a feasible solution due to memory limitations. On the other hand, SVNS reached the optimal solutions obtained by CPLEX for all 43 instances. For the remaining 36 instances, the SVNS solution was better than the feasible one obtained with the CPLEX solver. Only for one instance (pmed35_800_5_2) does the SVNS solution match the feasible solution obtained by CPLEX. In the cases where CPLEX provided optimal solutions, it often required several hours of CPU time, while the average running time of SVNS was significantly shorter than that required by CPLEX.

Fig. 4 shows the success rate (in %) of CPLEX and SVNS in reaching the optimal/best known value for all five instance groups within the given total execution time for CPLEX and the maximum number of iterations for SVNS.

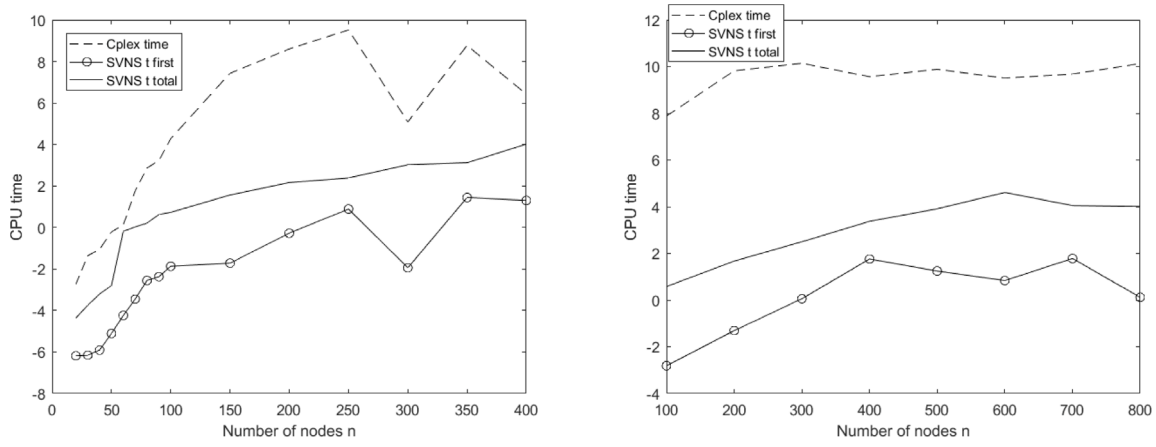
The summarized analysis of the CPU times required by CPLEX and SVNS to reach the optimal/best solution is shown in Fig. 5. The average CPLEX and SVNS running times (in seconds) for all instances with the same dimension n are shown in Fig. 5. Fig. 7(a) refers to the average CPLEX time, the average SVNS total execution time and the average SVNS time required to reach the best solution for the first time. These average values are calculated over all instances with the same dimension n belonging to small, medium and large datasets. Similarly, Fig. 7(b) refers to the running times on the subset of Additional pmed instances that could be solved by using CPLEX. Note that the X-axis indicates the problem dimension n, while the Y-axis represents the CPU time presented on a logarithmic scale to improve visualization.

To illustrate the quality of the solutions provided by SVNS, we show in Fig. 6 the percentage of instances from each group for which SVNS reached the best solution in each of the 20 runs ($agap = 0\%$). For the remaining instances (with $agap > 0\%$), Fig. 7 shows how the average $agap$ changes with increasing problem dimension n.

Note that the SVNS parameter values in our study were set to the same values for all instance groups considered. Our intention was to investigate the performance of the SVNS under the same conditions for all instances, i.e., without parameter adjustments for specific instance groups. We believe that fine tuning of parameters for each of the considered instance groups would lead to further improvements in the solution quality and shorter running times of the proposed SVNS.

4.4. Comparison with PNCP results

In this subsection, we compare the results of the CPNCP for given values p and q with the results of the PNCP when the number of centers to be established is $p+q$. Our motivation was to investigate two scenarios resulting from two different approaches to generate instances (which corresponds to two different approaches when planning service networks in real life situations). We also investigate the performance



(a) small, medium and large instances (b) Additional pmed instances

Fig. 5. Average CPU time required by CPLEX and SVNS algorithm (logarithmic scale).

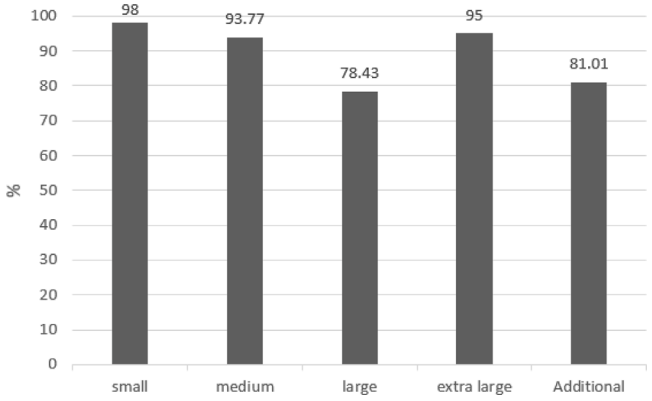


Fig. 6. Percentage of instances with $agap = 0\%$.

of the proposed SVNS approach in solving the PNCP problem. We run the SVNS with $q = 0$ on a set of PNCP instances from the literature and compare the obtained results with those obtained with recent PNCP solution methods.

4.4.1. Comparison of CPNCP and PNCP results

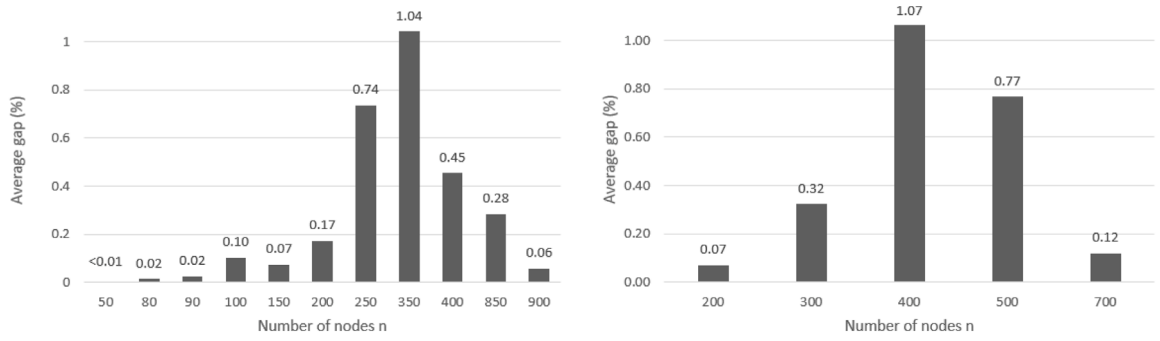
For a fair comparison of the two network design strategies, it was necessary to generate adequate CPNCP instances. We used two approaches to generate the CPNCP dataset: in the first approach, we start from the solutions of PNCP with q centers, while the second approach uses PNCP solutions with $p+q$ centers. The remaining parts of this subsection contain a detailed description of generating the CPNCP instances used for this purpose and the comparison of the best results obtained with SVNS for the CPNCP and the known optimal/best known solutions of the PNCP.

Generating instances for CPNCP from the optimal solution of PNCP with q centers

In the first approach, the q previously established centers for CPNCP are taken from the optimal solution of PNCP with q centers. It is obvious that the optimal solution value of CPNCP with q existing centers and p new centers to be chosen will not be lower than the optimal solution value of PNCP with $p+q$ centers. For example, for the instance pmed1_40_10_5, the five fixed centers for CPNCP are obtained from the optimal solution of PNCP with 5 centers: $\{0, 10, 28, 37, 38\}$. The optimal solution of CPNCP with the fixed centers in bold is $\{\mathbf{0}, \mathbf{10}, 17, 19, 20, 25, \mathbf{28}, 31, \mathbf{37}, \mathbf{38}\}$, and the objective function value is 116. This corresponds to the real situation of expanding a service network from 5 to 10 centers. On the other hand, the optimal solution of the PNCP problem with 10 centers is $\{6, 13, 14, 17, 20, 24, 28, 34, 37, 38\}$ with an objective function value of 111. This corresponds to a situation in which the entire network is designed from scratch, which is rarely the case in practice. Usually, companies expand their service network by adding new centers to the existing ones. The above example shows notice that centers 0 and 10 (the fixed centers in the CPNCP) are not included in the optimal solution of the PNCP with 10 centers. Thus, the solution of the CPNCP compared to the solution of the PNCP can either be a different solution but with the same objective function value or a solution having a greater objective function value, as in this example.

The presented results provide information on how much loss can be produced by keeping already existing facilities while expanding the network. Therefore, the objective value of a PNCP solution can be interpreted as the lower bound of the objective value of a CPNCP solution. In addition, the analysis of PNCP and CPNCP solutions and the comparison of the corresponding objective values can also be used to decide whether some of the already open facilities should be closed and replaced by another facility.

Table 4 shows the objective function value (*best*) of the optimal/feasible solution obtained by CPLEX when solving the PNCP on the set of small size instances from which the test instances for the CPNCP were generated. This is followed by the results of the SVNS algorithm for solving the CPNCP on the set of corresponding CPNCP instances generated by the first approach. Since our primary goal is to compare only the values of the obtained solutions, we present the best obtained objective function value (*best*) of the SVNS solutions for the CPNCP and the number of SVNS runs that provided this best value (#*best*). From the results shown in Table 4 for the set of 50 small size instances, it can be seen that for 35 instances the objective function value for the CPNCP is greater than the objective function value for the PNCP, while for the remaining 15 instances the values are equal. It can be noticed that even with the objective function values, the optimal solutions may contain different nodes. For example, in the case of the instance pmed1_20_10_5, the optimal solution of the PNCP with 10 nodes is $\{1, 3, 5, 7, 10, 11, 13, 15, 16, 17\}$ and the objective function value is 95. The fixed centers for CPNCP are $\{2, 6, 10, 11, 16\}$, while centers 2 and 6 are not included in the PNCP solution. On the other hand, the SVNS solutions obtained for CPNCP are $\{1, \mathbf{2}, 4, \mathbf{6}, \mathbf{10}, \mathbf{11}, 13, 15, \mathbf{16}, 17\}$, $\{\mathbf{2}, 3, 5, \mathbf{6}, \mathbf{10}, \mathbf{11}, 15, \mathbf{16}, 17, 19\}$, $\{1, \mathbf{2}, \mathbf{6}, 9, \mathbf{10}, \mathbf{11}, 14, 15, \mathbf{16}, 17\}$, etc., which all have the same objective function value 95. Therefore, in this case, nothing is lost by expanding the network compared to the design of the network from the beginning. On the other hand, the instance pmed2_40_10_5 has the result 112 for PNCP and 135 for CPNCP. It is



(a) Small, medium, large and extra large instances

(b) Additional pmed instances

Fig. 7. Average $agap$ values for instances with $agap > 0\%$.

obvious that already established centers prevent the full potential of the service network, so it is justified to consider the possible replacement of one or more already existing centers if this is possible.

Similar tables with the results on the sets of medium and large instances can be found in the Appendix B (Tasić et al., 2024b). For all these instances, a similar conclusion as for the instance $n \leq 100$ can be derived, i.e. the obtained objective function values for CPNCP are greater than or equal to the objective function values for PNCP.

Generating instances for CPNCP from solution of PNCP with $p + q$ centers

Following the second approach for generating test instances for CPNCP, we start from the solutions obtained by the heuristic (Tasić, 2024) for solving the PNCP on instances with $n \geq 100$ (note that these solutions do not necessarily coincide with the optimal ones). The results of the heuristic from Tasić (2024) were the only available solutions for which we had both the objective function values and the locations of opened centers, and these data were necessary to generate CPNCP instances by the second approach. When choosing the q random centers from the heuristic solutions of the PNCP (Tasić, 2024) as the locations of the previously established centers in the CPNCP, it is possible that the SVNS algorithm returns a CPNCP solution with a smaller objective function value compared to the initial PNCP heuristic solution (Tasić, 2024). In this situation, an improvement of the initial PNCP solution is provided.

We compared the results of CPNCP and PNCP on the set of instances with $n \geq 100$ generated by the second approach. Table 5 shows the average results for the set of Additional pmed instances, while the detailed results for the remaining instances can be found in Appendix B (Tasić et al., 2024b). Since these instances were generated from the heuristic solutions (Tasić, 2024), in order to provide a fair analysis of the two scenarios, we compare the CPNCP results obtained by SVNS with the heuristic solutions of PNCP from Tasić (2024). Note that the solution of the CPNCP is also a solution of the PNCP, so the CPNCP results with a lower objective function value than those for the PNCP obviously lead to an improvement of the existing PNCP solutions obtained by the heuristic from Tasić (2024).

In Table 5, the instances are grouped by dimension in the same way as in Table 3. For both CPNCP and PNCP, the table contains the number of optimal or best known solutions ($\#opt/\#bk$) from the same group. The percentage of the obtained optimal or best known solutions for the Heur (Tasić, 2024) is given i the ($\%opt/bk$) column. The last column $avg.savings(\%)$ shows the average decrease of the objective function value achieved by using SVNS for CPNCP, in respect to the objective function value of PNCP obtained by heuristic Heur.

From the data presented in Table 5, it can be seen that for 8 out of 79 instances, the SVNS for CPNCP reached solutions with a lower objective function value than the existing solutions of Heur for PNCP. For example, consider the instance pmed13_300_30 from the Additional pmed dataset. For this instance, the heuristic Heur (Tasić, 2024) for the

Table 4
Results of CPLEX for PNCP and the SVNS for CPNCP on the set of small-size instances generated by following the first approach.

Instance	n	p	q	CPLEX for PNCP	SVNS for CPNCP	
				best	*best*	*#best*
pmed1_20_10_5	20	5	5	95	95	20
pmed1_30_10_5	30	5	5	95	95	20
pmed1_40_10_5	40	5	5	111	116	20
pmed1_40_20_5	40	15	5	89	89	20
pmed1_40_20_10	40	10	10	89	89	20
pmed1_40_20_15	40	5	15	89	89	20
pmed1_50_10_5	50	5	5	110	112	20
pmed1_50_20_5	50	15	5	89	91	20
pmed1_50_20_10	50	10	10	89	94	20
pmed1_50_20_15	50	5	15	89	91	20
pmed2_20_10_5	20	5	5	99	100	20
pmed2_30_10_5	30	5	5	110	128	20
pmed2_40_10_5	40	5	5	112	135	20
pmed2_40_20_5	40	15	5	96	98	20
pmed2_40_20_10	40	10	10	96	98	20
pmed2_40_20_15	40	5	15	96	96	20
pmed2_50_10_5	50	5	5	140	145	20
pmed2_50_20_5	50	15	5	99	99	19
pmed2_50_20_10	50	10	10	99	104	20
pmed2_50_20_15	50	15	15	99	99	20
pmed3_20_10_5	20	5	5	77	92	20
pmed3_30_10_5	30	5	5	122	122	20
pmed3_40_10_5	40	5	5	105	122	20
pmed3_40_20_5	40	15	5	77	85	20
pmed3_40_20_10	40	10	10	77	85	20
pmed3_40_20_15	40	5	15	77	82	20
pmed3_50_10_5	50	5	5	125	128	20
pmed3_50_20_5	50	15	5	87	87	20
pmed3_50_20_10	50	10	10	87	88	20
pmed3_50_20_15	50	5	15	87	93	20
pmed4_20_10_5	20	5	5	125	125	20
pmed4_30_10_5	30	5	5	122	126	20
pmed4_40_10_5	40	5	5	122	126	20
pmed4_40_20_5	40	15	5	85	91	20
pmed4_40_20_10	40	10	10	85	91	20
pmed4_40_20_15	40	5	15	85	85	20
pmed4_50_10_5	50	5	5	126	141	20
pmed4_50_20_5	50	15	5	91	99	20
pmed4_50_20_10	50	10	10	91	108	20
pmed4_50_20_15	50	5	15	91	105	20
pmed5_20_10_5	20	5	5	91	91	20
pmed5_30_10_5	30	5	5	120	126	20
pmed5_40_10_5	40	5	5	127	137	20
pmed5_40_20_5	40	15	5	91	91	20
pmed5_40_20_10	40	10	10	91	97	20
pmed5_40_20_15	40	5	15	91	96	20
pmed5_50_10_5	50	5	5	121	121	20
pmed5_50_20_5	50	15	5	89	90	20
pmed5_50_20_10	50	10	10	89	91	20
pmed5_50_20_15	50	5	15	89	90	20

Table 5
Average results of Heur. (Tasić, 2024) for PNCP and the SVNS results for CPNCP on the set of instances generated following the second approach.

Instance	n	p	q	#inst	Heur. (Tasić, 2024) for PNCP		SVNS for CPNCP	
					$\#opt/\#bk$	$\%opt/bk$	$\#opt/\#bk$	$avg.savings$ (%)
pmed1–pmed5	100	[3,31]	[2,3]	6	6	**100**	6	0
pmed6–pmed10	200	[3,65]	[2,6]	8	8	**100**	8	0
pmed11–pmed15	300	[3,98]	[2,10]	10	8	80	10	0.42
pmed16–pmed20	400	[3,131]	[2,13]	11	8	72.73	11	1.70
pmed21–pmed25	500	[3,165]	[2,16]	11	9	81.82	11	0.52
pmed26–pmed30	600	[3,198]	[2,20]	14	14	**100**	14	0
pmed31–pmed34	700	[3,138]	[2,14]	9	9	**100**	9	0
pmed35–pmed37	800	[3,78]	[2,8]	5	5	**100**	5	0
pmed38–pmed40	900	[3,88]	[2,9]	5	4	80	5	0.83

Table 6
Literature review for PNCP instances.

Reference	Year	#inst	Stopping criterion	Solution data
Albareda-Sambola et al. (2015)	2015	132	max CPU = 4 h	Average results, CPU
López-Sánchez et al. (2019)	2018	132	The number of generated solutions	Best solution, CPU, gap
Londe et al. (2021)	2021	413	max CPU = 7 days	Best solution, CPU first
Zhang et al. (2022)	2022	413	CPU = 60 s	% of hits best solution, normalized CPU first
Ristić et al. (2021)	2023	104	max CPU = 5n	Best solution, avg.sol, worst sol, CPU first, #best (in 20 runs), gap
Ristić et al. (2023a)	2023	264	max CPU = 1800 s	Best sol., avg.sol., worst sol., CPU first, #best (in 20 runs)
Tasić (2024)	2023	285	max 5000 iterations	Best solution, #best (in 20 runs), CPU total, CPU first, gap, st.dev.

PNCP returned the best objective value of 49 (see Appendix B Tasić et al., 2024b). On the other hand, SVNS for CPNCP obtained the best solution with an objective function value of 48 for pmed13_300_30_2 and for pmed13_300_30_3. Note that the best SVNS solutions for CPNCP on these two instances are also the solution of PNCP for the instance pmed13_300_30. Similar situation occurs for the instances pmed19_400_80 and pmed23_500_50, as well as for pmed40_900_90. For the remaining 71 instances, SVNS for the CPNCP returned the same best objective values as for the ones obtained when solving the PNCP.

The results presented in this subsection show that comparing CPNCP and PNCP solutions can help decision makers to expand a service network. If locating a service facility requires significant installation costs, the decision maker will most likely opt for the CPNCP model and keep the installed facilities, as the losses generated by this model are likely to be lower than the costs of closing some of the existing facilities and opening new ones. In situations where opening facilities is cheap, the decision maker may consider designing the network from scratch with the required number of facilities. Therefore, the choice of model depends on the information about how much damage would be caused otherwise.

4.4.2. Comparison with recent approaches for solving PNCP

If the number of fixed centers is set to zero (q = 0), the CPNCP becomes the PNCP. Recently, many methods have been proposed for solving PNCP, but their direct comparison is challenging. The difficulties in providing direct comparison of solution methods for PNCP arise from several facts: Different sets of instances have been used in the different papers dealing with PNCP, the proposed methods use different stopping criteria, and the results are presented in different ways.

Table 6 presents our attempt to summarize the existing results for PNCP from the literature. For each paper dealing with PNCP, we indicate: the year in which the paper was published, the number of instances tested, the stopping criterion used for the proposed method, and the solution data available in the respective reference.

The main goal of these numerical experiments was to investigate the behavior of our SVNS algorithm for CPNCP when applied to solve PNCP. For this purpose, we used pmed instances, which represent a standard benchmark set in the literature commonly used for the p-center problem and its variants. Based on the available data, we divided all pmed instances used to test PNCP solution methods from the literature into 5 groups (453 instances in total), as shown in Table 7. In our experiments, we used the same parameter values as in the case of CPNCP, which means that these results should not be considered as the best possible results that can be obtained with SVNS when solving PNCP.

Table 8 shows the number of best solutions obtained by each method from all five groups and the percentage of best solutions obtained. (in the cases where this information was available).

It can be seen that the results obtained with our SVNS are comparable to seven recent methods from the literature. In 25 out of 453 cases (5.52%), the SVNS did not reach the best known PNCP solution. Although the papers (Zhang et al., 2022; Ristić et al., 2023a) show the performance of 100% for the tested instances, it should be taken into account that none of the mentioned papers in the literature used all pmed instances. For example, Group 5 was not used in Zhang et al. (2022), while Groups 3 and 4 were not used in Ristić et al. (2023a) (see Table 7). As can be seen from Section 4.3 and Table 3, exactly these instances were difficult to solve with the exact solver. It should also be mentioned that in Londe et al. (2021) the experiments performed for some instances took up to a week to obtain optimal/best known solutions. From the results presented in Table 8, it can be concluded that the SVNS developed for solving the CPNCP can also be successfully applied to the PNCP.

5. Conclusion

In this study, we have introduced the conditional p-next center problem (CPNCP), as a variant of the p-next center problem (PNCP). The considered CPNCP has an important role in the design and optimization of emergency systems and other service networks. The CPNCP covers two key requirements that arise in practice: opening new service centers while keeping the existing service in function and ensuring that if a primary center fails, its users are redirected to an open center that is closest to the primary center. The goal of the CPNCP is to minimize the distance from each customer to its backup center, passing through a primary center. In this study, an integer linear formulation of the CPNCP is given, and it is indicated that the problem itself is NP-hard, as a generalization of the classical p-center problem.

Due to the complexity of the CPNCP under consideration, an optimization method that is able to solve effectively large, realistic sized problem instances is required. This study proposes an efficient metaheuristic based on Skewed Variable Neighborhood Search (SVNS) as a

Table 7
PNCP instances.

Group	# inst	Inst. name	Inst. dimension	Reference(s)
Group 1	64	pmed1–pmed4, pmed6–pmed8	$10 \leq n \leq 200$	Albareda-Sambola et al. (2015), Londe et al. (2021), López-Sánchez et al. (2019), Ristić et al. (2023a), Ristić et al. (2021), Tasić (2024) and Zhang et al. (2022)
Group 2	68	pmed1–pmed4	$50 \leq n \leq 100$	Albareda-Sambola et al. (2015), Londe et al. (2021), López-Sánchez et al. (2019), Ristić et al. (2023a), Tasić (2024) and Zhang et al. (2022)
Group 3	113	pmed5, pmed9–pmed19	$10 \leq n \leq 400$	Londe et al. (2021), Tasić (2024) and Zhang et al. (2022)
Group 4	168	pmed20–pmed40	$350 \leq n \leq 900$	Londe et al. (2021) and Zhang et al. (2022)
Group 5	40	Additional pmed	$100 \leq n \leq 900$	Ristić et al. (2023a), Ristić et al. (2021) and Tasić (2024)

Table 8
Comparison of SVNS results when solving the PNCP (case $q = 0$) with results from the literature.

	Group 1		Group 2		Group 3		Group 4		Group 5	
	#best	%best	#best	%best	#best	%best	#best	%best	#best	%best
Albareda-Sambola et al. (2015)	64	100	68	100	–	–	–	–	–	–
López-Sánchez et al. (2019)	46	71.88	52	76.47	–	–	–	–	–	–
Londe et al. (2021)	63	98.44	68	100	112	99.2	158	95.05	–	–
Zhang et al. (2022)	64	100	68	100	113	100	168	100	–	–
Ristić et al. (2021)	50	78.12	–	–	–	–	–	–	29	72.5
Ristić et al. (2023a)	64	100	68	100	–	–	–	–	40	100
Tasić (2024)	63	98.44	68	100	102	90.27	–	–	33	82.5
SVNS	**64**	**100**	**68**	**100**	**109**	**96.46**	**149**	**88.69**	**38**	**95**

solution method for the CPNCP. Appropriate solution representation, neighborhood structures, and fast solution evaluation are used. One of the key aspects of the proposed SVNS is the efficient implementation of the Fast Interchange (FI) within the Local Search phase.

The experimental study was conducted on the set of modified pmed and Additional pmed instances with up to 900 nodes. The adequate value of the SVNS parameter α was found through the set of parameter tuning tests and statistical analysis of the obtained results. The results of the final computational experiments on modified pmed instances show that the proposed SVNS heuristic approach was able to reach optimal solutions previously obtained by the CPLEX solver in less CPU time. Moreover, SVNS improved the best feasible solution provided by CPLEX in cases where no optimal solution was found. For most of the extra large size pmed instances and Additional pmed instances, CPLEX failed to provide even a feasible solution within the given time limit. On the other hand, the proposed SVNS showed high stability in returning good quality solutions for these instances in short CPU times.

To investigate the effects of two scenarios in service network planning (the first one based on the PNCP model and the second one based on the CPNCP model), we generated CPNCP test examples from the PNCP solutions for which we the list of established centers available. We compared the SVNS solutions for the obtained CPNCP dataset with the best known PCNP solutions for PNCP instances from which the CPNCP dataset was generated. The presented results show how high the losses can be if the existing facilities are kept when expanding a service network (which is the case in most real-world situations). Finally, we considered the case of $q = 0$ in the CPNCP leading to the PNCP and compared the solutions of the proposed SVNS algorithm with $q = 0$ with the results of seven recent PNCP solution methods from the literature. The data obtained indicate that the SVNS results for the PNCP are comparable to the results of recent solution approaches from the literature that have been developed specifically for the PNCP.

The results presented in this paper show that SVNS is a promising solution approach for the CPNCP under consideration, but also for the PNCP. Future work could focus on the hybridization of SVNS with LP-based methods for solving CPNCP and considering to capacitated variants of CPNCP. Another promising idea is to decompose the CPNCP into a set of decision subproblems, and then apply an efficient heuristic such as tabu search, SVNS or another VNS-based heuristic to the obtained subproblems.

CRediT authorship contribution statement

Jelena Tasić: Writing – original draft, Software, Methodology, Investigation, Formal analysis, Data curation. **Zorica Dražić:** Writing – review & editing, Writing – original draft, Validation, Resources, Methodology, Formal analysis. **Zorica Stanimirović:** Writing – review & editing, Validation, Supervision, Formal analysis, Conceptualization.

Acknowledgments

The research of the authors was partially funded by Faculty of Mathematics University of Belgrade (the contracts 451-03-47/2023-01/200104 and 451-03-66/2024-03/200104) through the grant by the Ministry of Science, Technological Development, and Innovation of the Republic of Serbia.

Data availability

Data will be made available on request.

References

Albareda-Sambola, M., Hinojosa, Y., Marín, A., Puerto, J., 2015. When centers can fail: A close second opportunity. Comput. Oper. Res. 62, 145–156.
Berman, O., Drezner, Z., 2008. A new formulation for the conditional p-median and p-center problems,. Oper. Res. Lett. 36, 481–483.
Berman, O., Simchi-Levi, D., 1990. The conditional location problem on networks. Transp. Sci. 24, 77–78.
Brimberg, J., Mladenović, N., Todosijević, R., Urošević, D., 2019. Solving the capacitated clustering problem with variable neighborhood search. Ann. Oper. Res. 272, 289–321.
Brimberg, J., Mladenović, N., Urošević, D., 2015. Solving the maximally diverse grouping problem by skewed general variable neighborhood search. Inform. Sci. 295, 650–675.
Brimberg, J., Salhi, S., Todosijević, R., Urošević, D., 2023. Variable neighborhood search: The power of change and simplicity. Comput. Oper. Res. 155, 106221.
Callaghan, B., Slhi, S., Brimberg, J., 2018. Optimal solutions for the continuous p-centre problem and related α-neighbour and conditional problems: A relaxation-based algorithm. J. Oper. Res. Soc. 70 (2), 192–211.
Celik Turkoglu, D., Erol Genevois, M., 2020. A comparative survey of service facility location problems. Ann. Oper. Res. 292, 399–468.
Chen, R., 1990. Conditional minisum and minimax location-allocation problems in Euclidean space. Transp. Sci. 22, 158–160.

Chen, D., Chen, R., 2010. A relaxation-based algorithm for solving the conditional p-center problem,. Oper. Res. Lett. 38, 215–217.
Chen, R., Handler, G.Y., 1993. The conditional p-center in the plane. Naval Res. Logist. 40, 117–127.
Demšar, J., 2006. Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. 7, 1–30.
Drezner, Z., 1984. The p-centre problem-heuristic and optimal algorithms. J. Oper. Res. Soc. 35 (8), 741–748.
Drezner, Z., 1989. Conditional p-center problems. Transp. Sci. 23, 51–53.
Drezner, Z., 1995. On the conditional p-median problem. Comput. Oper. Res. 22, 525–530.
Hakimi, S., 1965. Optimum distribution of switching centers in a communication network and some related graph theoretic problem. Oper. Res. 13 (3), 462—-475.
Handler, G., Mirchandani, P.B., 1979. *Location on Networks: Theory and Algorithms*. MIT Press, Cambridge.
Hansen, P., Mladenović, N., 1997. Variable neighborhood search. Comput. Oper. Res. 24 (11), 1097–1100.
Hansen, P., Mladenović, N., Moreno Pérez, J.A., 2010. Variable neighbourhood search: methods and applications. Ann. Oper. Res. 175, 367–407.
Iravan, C.A., Slhi, S., Drezner, Z., 2016. Hybrid meta-heuristics with VNS and exact methods: Application to large unconditional and conditional vertex p-centre problems. J. Heuristics 22 (4), 507–537.
Kariv, O., Hakimi, S., 1979. An algorithmic approach to network location problems. Part 1: The p-centers. SIAM J. Appl. Math. 37 (3), 513–538.
Lin, C.C., 1975. A note about the new emergency facility insertion in an undirected connected graph. In: Proceedings of the Sixth Annual Pittsburgh Conference on Modelling Simulation, Pittsburgh, Penn. Vol. 1, pp. 375–377.
Londe, M.A., Andrade, C.E., Pessoa, L.S., 2021. An evolutionary approach for the p-next center problem. Expert Syst. Appl. 175, 114728.
López-Sánchez, A., Sánchez-Oro, J., Hernandez-Dõaz, A., 2019. GRASP and VNS for solving the p-next center problem. Comput. Oper. Res. 104, 295–303.
Macedo, R., Alves, C., Hanafi, S., Jarboui, B., Mladenović, N., Ramos, B., De Carvalho, J.V., 2015. Skewed general variable neighborhood search for the location routing scheduling problem. Comput. Oper. Res. 61, 143–152.
Minieka, E., 1980. Conditional centers and medians on a graph. Networks 10, 265–272.
Mladenović, N., Alkandari, A., Pei, J., Todosijević, R., Pardalos, P.M., 2020. Less is more approach: basic variable neighborhood search for the obnoxious p-median problem. Int. Trans. Oper. Res. 27 (1), 480–493.
Mladenović, N., Labbe, M., Hansen, P., 2003. Solving the p-center problem with tabu search and variable neighborhood search. Networks 42 (1), 48–64.
Mladenović, N., Todosijević, R., Urošević, D., Ratli, M., 2022. Solving the capacitated dispersion problem with variable neighborhood search approaches: From basic to skewed VNS. Comput. Oper. Res. 139, 105622.
Mousavi, S.R., 2023. Exploiting flat subspaces in local search for p-center problem and two fault-tolerant variants. Comput. Oper. Res. 149, 106023.
Mrkela, L., Stanimirović, Z., 2022. A variable neighborhood search for the budget-constrained maximal covering location problem with customer preference ordering. Oper. Res. 22 (5), 5913–5951.
Pelegrin, B., 1991. Heuristic methods for the p-center problem. RAIRO-Oper. Res. 25 (1), 65–72.
Pullan, W., 2008. A memetic genetic algorithm for the vertex p-center problem. Evol. Comput. 16 (3), 417–436.
Quevedo-Orozco, D.R., Ríos-Mercado, R.Z., 2015. Improving the quality of heuristic solutions for the capacitated vertex p-center problem through iterated greedy local search with variable neighborhood descent. Comput. Oper. Res. 62, 133–144.
ReVelle, C.S., Eiselt, H.A., 2005. Location analysis: A synthesis and survey. European J. Oper. Res. 165 (1), 1–19.
Ristić, D., Mladenović, N., Ratli, M., Todosijević, R., Urošević, D., 2023a. Auxiliary data structures and techniques to speed up solving of the p-next center problem: A VNS heuristic. Appl. Soft Comput. 140, 110276.
Ristić, D., Mladenović, N., Todosijević, R., Urosević, D., 2021. Filtered variable neighborhood search method for the p-next center problem. Int. J. Traffic Transp. Eng. 11 (2), 294–309.
Ristić, D., Urošević, D., Mladenović, N., Todosijević, R., 2023b. Solving the p-second center problem with variable neighborhood search. Comput. Sci. Inf. Syst. 20 (1), 95–115.
Sánchez-Oro, J., López-Sánchez, A.D., Colmenar, J.M., 2022. A multi-objective parallel variable neighborhood search for the bi-objective obnoxious p-median problem. Optim. Lett. 104, 1–31.
Sheshkin, D.J., 2000. Handbook of Parametric and Nonparametric Statistical Procedures. CRC Press, Boca Raton.
Tansel, B.C., Francis, R.L., Lowe, T.J., 1983. State of the art - location on networks: A survey. part I: the p-center and p-median problems. Manage. Sci. 29 (4), 482–497.
Tasić, J., 2024. An efficient solution approach to the p-next center problem. Mat. Vesnik 76 (1–2), 66–83.
Tasić, J., Dražić, Z., Stanimirović, Z., 2024a. Complete results of the SVNS mehod for the CPNCP. http://www.matf.bg.ac.rs/p/files/77-AppA.pdf. (Last Access: 25 August 2024).
Tasić, J., Dražić, Z., Stanimirović, Z., 2024b. Detailed comparison of the results for the CPNCP and PNCP on the sets of medium and large instances. http://www.matf.bg.ac.rs/p/files/77-AppB.pdf. (Last Access: 25 August 2024).
Tasić, J., Dražić, Z., Stanimirović, Z., 2024c. Parameter tuning. http://www.matf.bg.ac.rs/p/files/77-AppendixC.pdf. (Last Access: 18 April 2024).
Whitaker, R., 1983. A fast algorithm for the greedy interchange for large-scale clustering and median location problems. INFOR 21 (2).
Zhang, Q., Lü, Z., Su, Z., Li, C., 2023. A vertex weighting-based double-tabu search algorithm for the classical p-center problem. Comput. Oper. Res. 160, 106373.
Zhang, Q., Su, Z., Lü, Z., Yang, L., 2022. A weighting-based tabu search algorithm for the p-next center problem. In: Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (IJCAI-22). pp. 4828–4834.