Journal of Global Optimization (2021) 81:23–83
https://doi.org/10.1007/s10898-020-00962-4

The conditional *p*-dispersion problem

Marilène Cherkesly[1,2,3] · **Claudio Contardo**[1,2,3]

Received: 29 August 2019 / Accepted: 24 October 2020 / Published online: 21 November 2020

Abstract
We introduce the conditional p-dispersion problem (c-pDP), an incremental variant of the p-dispersion problem (pDP). In the c-pDP, one is given a set N of n points, a symmetric dissimilarity matrix D of dimensions $n \times n$, an integer $p \geq 1$ and a set $Q \subseteq N$ of cardinality $q \geq 1$. The objective is to select a set $P \subset N \setminus Q$ of cardinality p that maximizes the minimal dissimilarity between every pair of selected vertices, i.e., $z(P \cup Q) := \min\{D(i, j), i, j \in P \cup Q\}$. The set Q may model a predefined subset of preferences or hard location constraints in incremental network design. We adapt the state-of-the-art algorithm for the pDP to the c-pDP and include an ad-hoc acceleration mechanism designed to leverage the information provided by the set Q to further reduce the size of the problem instance. We perform exhaustive computational experiments and show that the proposed acceleration mechanism helps reduce the total computational time by a factor of five on average. We also assess the scalability of the algorithm and derive sensitivity analyses.

Keywords p-Dispersion · Clustering · Conditional p-dispersion · Exact methods

1 Introduction

The conditional p-dispersion problem (c-pDP) is defined on an undirected graph $G = (N, E)$, where N is the set of vertices with cardinality n and E is the set of edges with cardinality $n \times n$. The set of vertices N is composed of the union of two disjoint sets Q and R, where Q is the set of initial located vertices such that $|Q| = q \geq 1$, and where R is the set of potential additional locations such that $|R| \geq p \geq 1$. Each edge $(i, j) \in E$ is associated with a dissimilarity $D(i, j)$ satisfying $D(i, j) = D(j, i) \geq 0$ for every $1 \leq i, j \leq n$ and

✉ Claudio Contardo
contardo.claudio@uqam.ca

Marilène Cherkesly
cherkesly.marilene@uqam.ca

1 Department of Analytics, Operations and Information Technologies, ESG UQAM, Montreal, Canada

2 Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT), Montreal, Canada

3 Group for Research in Decision Analysis (GERAD), Montreal, Canada

$D(i,i) = 0$ for every $1 \leq i \leq n$. We also denote by $D = \{D(i,j) : (i,j) \in E\}$ the dissimilarity matrix. The objective is to select a subset of vertices $P \subseteq R$ of cardinality p such that $z(P \cup Q) := \min\{D(i,j), i, j \in P \cup Q\}$ is maximum. We denote the associated c-pDP for given input parameters N, Q, D and p as c-pDP(N, Q, D, p). The c-pDP is $\mathcal{NP}$-hard as an instance of the pDP—$\mathcal{NP}$-hard as noticed by Erkut [24]—can be reduced to an equivalent instance of the c-pDP by adding a single vertex u to N with sufficiently large dissimilarity to every other vertex in N, and thus by solving c-pDP(N′, Q′, D′, p), with $N' = N \cup \{u\}$, $Q' = \{u\}$ and D' built from extending the dissimilarity matrix D by one row and column associated with the new vertex u. Note that the c-pDP is also *slicewise polynomial* [18] in $O(n^p)$ time by exhaustive enumeration.

The c-pDP arises as an incremental variant of pDP in contexts where a decision planner desires to enlarge a previously existing network by locating additional facilities to improve its service. In location science, the pDP can be applied, for example, when locating harzardous facilities the furthest away from the population and when locating multiple stores to prevent cannibalism [29]. Belotti et al. [2] describe an application of the pDP in the location of waste disposal centres to reduce the undesirable effects on the surrounding populations. In multiobjective optimization, solving a pDP can be used, for example, to select p solutions with different characteristics when the Pareto frontier contains many solutions [36]. In portfolio optimization, selecting a diversified portfolio is important [38] and solving a pDP can be used to select p investment options with distinct features in order to minimize the risk associated with the portfolio [36]. Finally, in the maximin split clustering problem, which can be solved in polynomial time [17], one has to create p clusters in order to maximize the minimum dissimilarity between each pair of observation from different clusters. Similar applications can be found for the c-pDP. In particular, network design is often done in multiple iterations which can be separated by many years due, for example, to limited funding. Therefore, an initial set of facilities are located and additional locations are added later on. In portfolio optimization, this arises when initially selecting a set of investment options and then, when one has more money to invest, when expanding the portfolio.

The state-of-the-art solver for the pDP [13] relies on the clustering of the data to solve a series of small instances—each of which provides an upper bound of the problem—in a dynamic fashion. To the best of our knowledge, the incremental variant has not been previously studied and little is known regarding the potential of exact methods to handle this problem. Therefore, managerial insights can be derived by analyzing the impact of optimally locating from scratch compared to incrementally designing the network. In this paper, our main contributions are as follows. First, we adapt the state-of-the-art algorithm proposed by Contardo [13] and include new tailored acceleration mechanisms for the c-pDP that take advantage of the additional information provided by the set Q. Second, we show that such acceleration mechanisms outperform state-of-the-art algorithms for the pDP for which additional fixing constraints for the set Q could be added. Third, through an exhaustive computational campaign, we analyze the scalability of the proposed algorithm. In particular, instances with more than 100,000 vertices are solved to optimality which makes sense in practice, for example, in portfolio optimization problems where the number of investment options to choose from is often very large.

The remainder of this article is organized as follows. In Sect. 2, we present a review of the relevant scientific literature. In Sect. 3, we propose an exact solution algorithm for the c-pDP including new acceleration mechanisms tailored for this problem. In Sect. 4, we assess the effectiveness of the algorithm through our computational experiments, show that naive variants of state-of-the-art algorithms for the pDP are not efficient to solve the c-pDP, and derive sensitivity analyses. Finally, Sect. 5 concludes the paper.

2 Literature review

To the best of our knowledge, the `c-pDP` has not been previously studied, but related problems such as the discrete [24] and continuous [22,23] `pDP`, and conditional variants of p-center [6] and p-median [16] problems have already been subjects of scientific analysis. Our proposed decremental clustering algorithm is similar to relaxation mechanisms that have been proposed to solve minimax and maximin optimization problems to optimality [1,7–11,33] as well as routing problems [5,32,35]. This section presents the literature and state-of-the-art algorithms for related problems to the `c-pDP` as well as algorithmic variants of the proposed decremental clustering algorithm.

While several authors have studied the `pDP` and developed exact and heuristic solution approaches, we only highlight the most relevant contributions in exact methods. Kuby [29] introduced the first mathematical formulation for the `pDP` which was based on mixed-integer linear programming and used Big-M coefficients to compute the minimal distance between each pair of vertices. They obtained results for instances with exactly 25 vertices. Pisinger [33] later strengthened that formulation by introducing a quadratic formulation which is solved though different relaxations and embedding the upper bounds in a branch-and-bound algorithm. Their algorithm can solve instances with up to 100 vertices. More recently, Sayah and Irnich [37] introduced a binary compact formulation and proposed two enhancements to strengthen the formulation: bounds on the optimal distances and new valid inequalities. They propose a branch-and-cut algorithm to solve the problem and report solutions for instances with up to 1000 vertices. The state-of-the-art algorithm for the `pDP` was proposed by Contardo [13] and dynamically solves a series of relaxations of the problem in an incremental fashion. This algorithm optimally solves instances with more than 100,000 vertices in reasonable computational time.

Other related problems to the `c-pDP` are the conditional p-center and the conditional p-median problems. These two problems are said to be conditional if a set of facilities are already open and when the problem aims to locate p additional facilities. In the conditional p-median problem, the objective consists of minimizing the sum of the weighted distances between the demand points and their closest facility, whereas the objective of the p-center problem consists of minimizing the maximum distance between demand points and their closest facility. Drezner [20] show that the conditional p-center problem can be solved by solving $\mathcal{O}(\log n)$ p-center problems, where n is the number of potential facilities. Berman and Simchi-Levi [4] later propose an algorithm to solve both the conditional p-center and conditional p-median problems by solving a $(p{+}1)$-center (or median) problem and modifying the distance matrix. Berman and Drezner [3] have implemented an algorithm based on an alternative distance matrix and show that the proposed distance matrix outperforms the method of Berman and Simchi-Levi [4] for the the conditional p-center and the conditional p-median problems. Drezner [21] propose solution methods for the conditional 1-median problem on an instance with 100 vertices. Chen and Chen [9] suggest that reverse relaxation can be used to optimally solve the conditional p-center problem which iteratively solves p-center problem and show that the complexity of the proposed algorithm is $\mathcal{O}(n)$. More recently, Irawan et al. [27] have developed a heuristic multi-phase algorithm for the conditional p-median problem for large instances and are able to solve instances with up to 89,600 vertices within reasonable computational time. Drezner and Drezner [19] have also solved variants of the conditional location problem where the two facilities are sequentially located: the first one can either be randomly or optimally located, and the second one is optimally located according to the first one. They have tested three variants of the problem: the minisum, the minimax and the competitive problem. For the minisum and the minimax, their results show that randomly locating the first facility provides better results than optimally locating the first facility, which can be counter-intuitive.

As previously mentioned, the current state-of-the-art algorithm for the `pDP` consists of clustering the vertices to iteratively solve smaller instances [13]. Other decremental relaxation methods have been proposed to solve minimax and maximin optimization problems to optimality. We can note the decremental relaxation methods proposed by Chen and Chen [8] and Contardo et al. [15] to solve p-center problems. Their algorithm iteratively solves relaxed problems ignoring a few allocation constraints and by adding them only when needed. The algorithm proposed by Contardo et al. [15] solves instances with to 1,000,000 vertices to optimality within reasonable computation time. Aloise and Contardo [1] has also proposed a relaxation method to solve the minimax diameter clustering problem (MMDCP), where the objective consists of grouping the vertices in k clusters while minimizing the maximum intra-cluster dissimilarity. As for the p-center problem, their relaxation method iteratively and dynamically solves smaller MMDCP. Their algorithm can solve instances with up to 600,000 vertices to optimality within reasonable computational time. Recently, Contardo and Sefair [14] introduced a progressive approximation scheme for the solution of sparse large-scale binary interdiction games that is closely related to our algorithm.

Finally, reduction techniques have also been studied in the literature and are usually specifically designed for a solution-method or a problem-structure. In general, dual and primal information is leveraged to derive the conditions satisfied by an optimal solution. In mixed-integer programming, reduced cost fixing is a common technique which consists in fixing non-basic variables to some of their bounds after an iteration of the simplex method [12]. In column generation, variable fixing provides important speed-ups for vehicle routing [28]. For continuous facility location problems, Hooker et al. [26] present a general framework to identify a *finite dominating set*, which is a set of finite cardinality containing all potentially optimal locations. Network flows problems can also benefit from graph reduction techniques to decrease the complexity of finding augmenting cycles in the residual network [25, see, for instance,]. In minimax or maximin optimization problems, such techniques can have an even greater impact because the dual information can often be reduced to a single variable. This is the case, for instance, of p-center problems or minimax diameter clustering problems where a single arc may account for the total cost of a solution. As shown in Aloise and Contardo [1,30,31], assignments exceeding the cost of a primal bound can be discarded.

3 An exact decremental clustering algorithm

In this section, we describe the proposed exact decremental clustering algorithm to solve the `c-pDP`. The algorithm is initialized with an initial feasible solution for the `c-pDP` which is constructed through a heuristic algorithm. This solution yields a lower bound for the `c-pDP` and allows to discard multiple non-promising vertices, thus yielding a more compact—yet fully equivalent—input graph. Finally, we solve a `pDP` on the reduced graph using the algorithm proposed by [13]. Algorithm 1 provides an overview of our algorithm. In the next subsections, we describe the different steps of the algorithm.

Algorithm 1 Exact decremental clustering for the `c-pDP`(N, Q, D, p)

Require: N, Q, D, p
  $\underline{z}$ ←`heuristicCPDP`(N, Q, D, p)
  G', D' ← `graphReduction`(N, Q, D, $\underline{z}$)
  S ← `decrementalClusteringPDP`(G′, D′, p)
  return S

Please note that the case where $p = 1$ can be solved in linear time by finding a vertex $u^* \in \arg\max\{\min\{D(i, u) : i \in Q\} : u \in R\}$. Therefore, in the remainder of this article, we assume that $p \geq 2$.

3.1 Procedure heuristicCPDP(N, Q, D, p)

In this section, we describe a heuristic construction algorithm for the `c-pDP` which provides a non-trivial lower bound $\underline{z}$ to the problem. This procedure is not tailored to provide near-optimal solutions, but helps reducing the size of the graph which is given as an input for the decremental clustering algorithm. The set Q of initially selected vertices defines the initial solution $X \leftarrow Q$. The value of the objective function for this initial solution can be computed as $d \leftarrow \min\{D(u, v) : u, v \in Q\}$. Then, for each vertex $i \in N \backslash X$, we compute $d_{i^+} \leftarrow \min\{D(u, v) : u, v \in Q \cup \{i\}\}$ which is the minimal dissimilarity resulting from adding vertex i to X. The set X is then enlarged by adding vertex $i \in \arg\max\{d_{i^+} : i \in N \backslash X\}$. If multiple such vertices exist, the one with the smallest index is chosen to enlarge X. This procedure is repeated until $|X| = q + p$. This heuristic yields a valid lower bound which can be computed as $\underline{z} \leftarrow \min\{D(u, v) : u, v \in X\}$.

3.2 Procedure graphReduction(N, Q, D, $\underline{z}$)

In this section, we describe a procedure to reduce the input graph. Given the heuristic lower bound $\underline{z}$ found with the procedure `heuristicCPDP`(N, Q, D, p), we define $\overline{N} \leftarrow \{i \in N : \exists j \in Q, D(i, j) < \underline{z}\}$ as the set of vertices that lie at a distance strictly lower than $\underline{z}$ from at least one vertex in Q. Note that, by definition, the set $\overline{N}$ contains set Q. We define the set of vertices in the reduced graph G' as $N' \leftarrow N \setminus \overline{N}$. Once this is done, the dissimilarity matrix of G' is modified by setting the dissimilarity between two vertices i and j as the minimum between their dissimilarity and the dissimilarity between each of these two vertices and its closest vertex in Q, i.e., for $i, j \in N'$, we set

$$D'(i, j) \leftarrow \min\{D(i, j), \min\{D(i, k), D(j, k) : k \in Q\}, \min\{D(u, v) : u, v \in Q\}\}. \quad (1)$$

The dissmilarity matrix D' is defined as $D' = \{D'(i, j) : i, j \in N'\}$.

Proposition 1 *Given a valid lower bound* $\underline{z}$ *and the set* $\overline{N} \leftarrow \{i \in N : \exists j \in Q, D(i, j) < \underline{z}\}$, *no vertex* $i \in \overline{N}$ *is in the optimal solution of the* `c-pDP`.

Proof Let us define $\underline{z}$ as the lower bound found with the procedure `heuristicCPDP`(N, Q, D, p) and the set of vertices within a maximal dissimilarity of $\underline{z}$ and one of the initially located vertices $j \in Q$ defined as $\overline{N} \leftarrow \{i \in N : \exists j \in Q, D(i, j) < \underline{z}\}$. If one of the vertex in $\overline{N}$ is in the optimal solution, this implies that $z^* \leq \underline{z}$. Because the objective function is maximized, this is a contradiction and this solution cannot be optimal. □

Proposition 2 *Solving the* c-pDP *with the dissimilarity matrices D' and D yields the same optimal solution value.*

Proof Let us define z^*_D as the value of the optimal solution of the c-pDP when solving it with the original dissimilarity matrix D, and $z^*_{D'}$ as the value of the optimal solution of the c-pDP when solution with the dissimilarity matrix D'. Let us also define $X = (x_1, \ldots, x_q, x_{q+1}, \ldots, x_{q+p})$ as an optimal solution obtained when solving the c-pDP with dissimilarity matrix D. We aim to prove that $z^*_D = z^*_{D'}$.
Case 1 If the minimal dissimilarity between two vertices in X is equal to the minimal dissimilarity between two vertices in Q, i.e., $z^*_D = \min\{D(i, j), i, j \in X\} = \min\{D(i, j), i, j \in Q\}$. Then, this implies that there are at least p vertices further than z^*_D from the set Q and themselves, i.e., $\min\{D(i, u) : i \in X\backslash Q, u \in Q\} \geq \min\{D(u, v) : u, v \in Q\}$. In that case, by defining $D'(i, j)$ with Eq. (1), we know that $D'(i, j) = \min\{D(u, v) : u, v \in Q\}, \forall i, j, \in X\backslash Q$. Therefore, $z^*_{D'} = \min\{D(u, v) : u, v \in Q\}$ and $z^*_D = z^*_{D'}$.
Case 2 If the minimal dissimilarity is between a vertex $i \in Q$ and a vertex $j \in X\backslash Q$. This implies that, $D(i, k) \geq D(i, j), \forall k \in X\backslash\{i, j\}$ and that $D(j, k) \geq D(i, j), \forall k \in X\backslash\{i, j\}$. As defined by Eq. (1), this implies that $D'(j, k) = D(i, j), \forall k \in X\backslash Q, k \neq j$. Therefore, because $z^*_D = D(i, j)$ and $z^*_{D'} = D'(j, k) = D(i, j)$, then $z^*_D = z^*_{D'}$.
Case 3 If the minimal dissimilarity is between two vertices $i, j \in X\backslash Q$. This implies that, $D(i, k) \geq D(i, j), \forall k \in Q$ and that $D(j, k) \geq D(i, j), \forall k \in Q$. As defined by Eq. (1), this implies that $D'(i, j) = D(i, j)$. Therefore, because $z^*_D = D(i, j)$ and $z^*_{D'} = D'(i, j) = D(i, j)$, then $z^*_D = z^*_{D'}$. □

3.3 Procedure decrementalClusteringPDP(G', D', p)

In this section, we describe the decremental clustering procedure to solve the c-pDP on the reduced graph G' and the dissimilarity matrix D'. As explained in Sect. 3.2, the set of vertices N' in the reduced graph G' does not contain the set Q. Therefore, solving the c-pDP on the reduced graph G' and the dissimilarity matrix D' implies solving a pDP. The proposed decremental clustering procedure uses that of Contardo [13] which is the current state-of-the-art of the pDP. In this procedure, the first step consists of computing valid upper and lower bounds. Given these bounds, initial clusters are created which are used to solve to optimality the problem through decremental clustering. In order to provide a self-contained paper, we explain each step of this procedure in the next subsections.

3.3.1 Step 1: Valid upper and lower bounds

At this step, a valid and non-trivial upper bound $\overline{z}$ and lower bound $\underline{z}$ are computed for the pDP. The upper bound $\overline{z}$ is computed as the maximal dissimilarity between any pair of vertices in the reduced graph G', i.e., $\overline{z} = \max\{D'(i, j) : i, j \in N'\}$. The lower bound is computed using the greedy construction heuristic described in Sect. 3.1 starting with $X = \{u, v\}$ where u and v are the vertices in N' that maximize $\{D'(u, v) : u, v \in N'\}$. The value of the objective function for this heuristic solution is computed as $\underline{z} \leftarrow \min\{D'(i, j) : i, j \in X\}$.

3.3.2 Step 2: Creation of an initial cluster

A this step, a *sufficiently refined clustering* of the vertices in N' is created. The concept of *sufficiently refined clusters* was introduced by Contardo [13] and can be stated as follows. Let

$\mathcal{C} = \{C_i : i = 1, \ldots, m\}$ define a clustering such that each vertex is contained in exactly one cluster, i.e., $C_i \cap C_j = \emptyset, \forall\, 1 \le i < j \le m$ and $\cup\{C_i : i = 1, \ldots, m\} = N'$. That cluster is said to be *sufficiently refined* if the maximal dissimilarity between any pair of vertices in that cluster is lower than z^*, the optimal solution value of the pDP, that is, $\max\{D'(u, v) : u, v \in C_i\} < z^*$. Considering that $\underline{z} \le z^*$, a clustering for which $\max\{D'(u, v) : u, v \in C_i\} < \underline{z}$ is *sufficiently refined*.

To build a *sufficiently refined clustering* of the vertices in N', p initial clusters are created with a k-means algorithm, where $k = p$. If this initial clustering is *sufficiently refined*, then the algorithm stops and returns the clustering $\mathcal{C}$ as well as its associated dissimilarity matrix $D^{\mathcal{C}}$, where $D^{\mathcal{C}}(i, j) = \max\{D'(u, v) : u \in C_i, v \in C_j\}$. Otherwise, additional clusters are iteratively created until the clustering is *sufficiently refined*. At each iteration, one additional cluster is created by dividing the cluster with the maximal dissimilarity C_{i^*}, i.e., $i^* \leftarrow \arg\max\{D^{\mathcal{C}}(i, i) : i = 1, \ldots, m\}$, into two smaller clusters using a k-means algorithm where $k = 2$. To reduce the computational time, upon removing a cluster and adding two clusters, only the new rows and columns of the dissimilarity matrix are computed.

3.3.3 Step 3: Optimal decremental clustering

To solve the pDP to optimality, we use the decremental clustering procedure proposed by Contardo [13]. Let us denote by W the complete set of clusters in the optimal solution, and by S the set of optimal clusters with more than one vertex, i.e., $S = \{w \in W : |C_w| \ge 2\}$. This procedure consists of iteratively solving to optimality the pDP on a reduced graph consisting of clusters and creating new clusters when at least one cluster of the optimal solution has more than one vertex. When all the clusters of the optimal solution have exactly one vertex, the procedure stops and returns an optimal solution. We hereby explain in more detail this procedure.

Solving to optimality the pDP

In decremental relaxation schemes, there is often degeneracy due to the fact that the optimal value of the problem does not decrease from one iteration to the next [1,13,15]. Therefore, the pDP is first solved heuristically. If an optimal solution is found, the exact solver is not executed. Otherwise, it is. Note that at the first iteration, the heuristic solver is not executed.

Given a clustering $\mathcal{C}$ for which one cluster has been splitted at the previous iteration, the heuristic solver consists in identifying p vertices from the set W', where W' denotes the optimal solution of the previous iteration for which one of the optimal clusters has been splitted into two clusters. If it is possible to find a solution for which the value is equal to $\overline{z}$, i.e., the upper bound of the previous iteration, then this solution is optimal and the exact solver is not executed. Otherwise, the exact solver is executed.

The exact solver consists of embedding the branch-and-cut algorithm proposed by Sayah and Irnich [37] within the double binary search algorithm proposed by Contardo [13] which exploit the fact that the upper bound $\overline{z}$ is monotonically decreasing. The first binary search consists of finding an optimal solution of the pDP for the given clustering. The algorithm starts by setting $\underline{z}', \overline{z}' \leftarrow \overline{z}$. Then, the pDP is solved through the branch-and-cut algorithm proposed by Sayah and Irnich [37]. This branch-and-cut uses a compact binary integer formulation containing two types of variables, i.e., binary variables to indicate if a vertex is used to locate a facility, and binary variables which indicate if a a location decision satisfies a minimal dissimilarity. This formulation is tightened with the lower bound $\underline{z}'$ and the upper bound $\overline{z}'$, and additional valid inequalities which consider the incompatibility of two locations within a minimal dissimilarity are added. We refer the reader to the paper of Sayah and Irnich [37] for details on the model as well as the inequalities and their separation procedure If no feasible solution is found, the bounds are modified as follows: $\overline{z}' \leftarrow \underline{z}' - 1$, $\underline{z}' \leftarrow \underline{z}' - 2^t$, where t represents the iteration number and is initially set to one, and the branch-and-cut algorithm is iterated. If a feasible solution is found, the lower bound is modified as follows: $\underline{z}' \leftarrow z'^*$, where z'^* is the value of that feasible solution. The second binary search is done to close the gap between $\underline{z}'$ and $\overline{z}'$ and is is repeated until $\underline{z}' \geq \overline{z}'$. Closing the gap is important as it reduces the dimension of the mathematical model and thus reduces the total computational time. First, we set $\underline{z}'' \leftarrow \lceil (\underline{z}' + \overline{z}')/2 \rceil$ and solving the problem through branch-and-cut given the lower bound $\underline{z}''$ and the upper bound $\overline{z}'$. If a feasible solution is found, we set $\underline{z}' \leftarrow \underline{z}''$, otherwise the value of the upper bound is reduced as follows: $\overline{z}' \leftarrow \underline{z}'' - 1$.

Creating and adding new clusters

Given the optimal solution W of the previous iteration, this step consists of selecting a cluster $s \in S$, $S \subseteq W$ such that $S = \{w \in W : |C_w| \geq 2\}$ and splitting it into two new clusters. To determine the splitted cluster, the pair of clusters in $S \times W$ with minimal dissimilarity is identified, i.e., $(s^*, w^*) \leftarrow \arg\min\{D^{\mathcal{C}}(s, w) : s \in S, w \in W\}$. Then, the vertex $u^* \in \{s^*, w^*\}$ with the maximal dissimilarity is identified, i.e., $u^* \leftarrow \arg\max\{D^{\mathcal{C}}(u, u) : u \in \{s^*, w^*\}\}$. Note that if $w^* \in W \backslash S$, then by definition $u^* \in s^*$. Given the vertex u^*, a k-means algorithm with $k = 2$ is executed to create two new clusters. The clustering $\mathcal{C}$ and its associated dissimilarity matrix $D^{\mathcal{C}}$ are then updated.

4 Computational results

Our algorithm has been implemented in Julia v1.1 using the JuMP interface v18.5. The general purpose MIP solver required to solve the restricted pDPs (see Sect. 3.3.3) is Gurobi v8.1. All tests were performed on an computer equipped with an Intel Xeon E5-2637 v2 (3.5 GHz) processor and with 128 GB of RAM. Given that this problem is strategic, the maximum computation time was set to 86,400 s, i.e., one day, and the number of threads is limited to one. Note that the computational time does not comprise the time needed to generate the set Q, as this is given for the c-pDP.

We have tested our algorithm on instances from the TSPLIB [34] containing between 1621 and 104,815 vertices. For each instance, we have computed the dissimilarity between two vertices according to the TSPLIB standards and round it to the nearest integer. For the instances for which vertices with identical coordinates exist, we remove all redundant vertices. To generate the set Q, three construction strategies have been tested: greedy, optimal and random construction (see “Appendix A”). On our set of instances, greedily constructing the set Q provides the best solutions with the lowest computational time. Therefore, our computational analysis is derived for the instances where the set Q is greedily constructed. We have tested four values of q and p, i.e., $q, p \in \{5, 10, 15, 20\}$, for a total of 16 combinations.

We have designed a series of experiments with the aim of performing a sensitivity analysis of the c-pDP to the input parameters and to assess the quality of the proposed algorithm. In the first subsection, we analyze the sensitivity of the model to the different values of p and q. In the following subsection, we determine the performance of our algorithm. We compare our algorithm against a naive adaptation of the state-of-the-art algorithm for the pDP to the conditional case, and show that our proposed algorithm reduces by a factor of five on average the total computational time compared to a more naive approach. We also test higher values of q and p, and show that our algorithm performs well even when increasing q and p. For

Table 1 Average deviation from the best known solution

	$p = 5$	$p = 10$	$p = 15$	$p = 20$
$q = 5$	0.0	− 19.8	− 31.8	− 40.6
$q = 10$	− 28.3	− 37.7	− 44.1	− 49.4
$q = 15$	− 41.4	− 48.8	− 53.2	− 56.4
$q = 20$	− 51.2	− 55.5	− 58.6	− 60.8

Table 2 Average CPU time (s)

	$p = 5$	$p = 10$	$p = 15$	$p = 20$
$q = 5$	186.7	1095.0	2913.6	8893.7
$q = 10$	63.4	331.6	944.5	2072.3
$q = 15$	19.4	46.1	171.6	328.0
$q = 20$	19.1	41.0	97.6	298.9

Table 3 Average percentage of eliminated vertices

	$p = 5$	$p = 10$	$p = 15$	$p = 20$
$q = 5$	75.6	51.4	38.2	28.9
$q = 10$	93.8	79.3	63.0	52.9
$q = 15$	97.9	90.2	81.3	71.8
$q = 20$	98.9	94.9	88.2	80.3

conciseness reasons, we only report summarized results in the manuscript. Detailed results can be found in "Appendices B–D".

4.1 Sensitivity analyses on the sizes of q and p

In this section, we analyze the impact of increasing the value of q and p on the quality of the solutions as well as on the total computational time and the number of eliminated vertices with our graph reduction technique. Detailed results are reported in "Appendix B". Tables 1, 2 and 3 present the average deviation from the best known solution, the average time in seconds and the average percentage of eliminated vertices according to the value of q and p, respectively. These tables are not discussed in this section, but will be used for the analysis conducted in the next subsections.

4.1.1 Analysis on modifying the value of p while fixing the value of q

In this section, we focus our analysis on the impacts of modifying the value of p while fixing the value of q. Therefore, Figs. 1, 2 and 3 illustrate the impact on the average deviation from the best known solution, on the average time (in seconds), and on the average percentage of eliminated vertices, respectively, when modifying the value of p and fixing $q = 5$. In each figure, the x-axis represents the number of vertices in each instance and the y-axis the deviation with the best known solution, the average time (in seconds), or the average percentage of eliminated vertices, respectively. Note that the best known solution is the solution with the highest objective function value at optimality when setting $q = 5$. Similar results are obtained when fixing q to 10, 15, and 20.

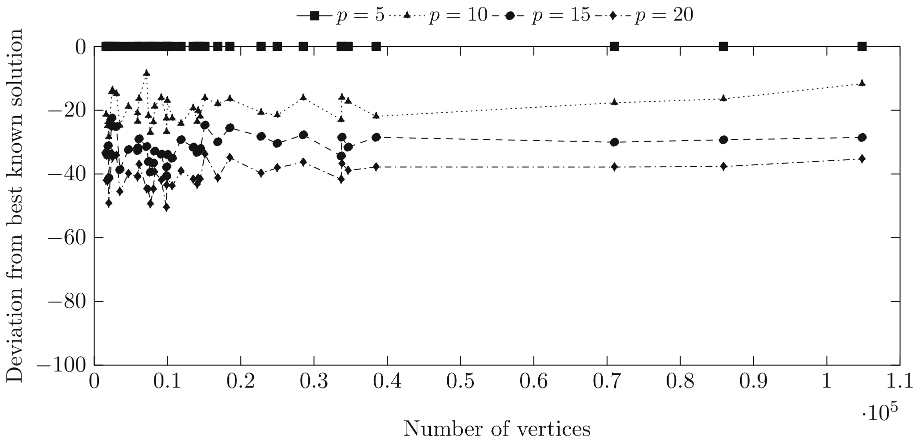

Fig. 1 Impact of increasing the value of p on the deviation from the best known solution ($q = 5$)

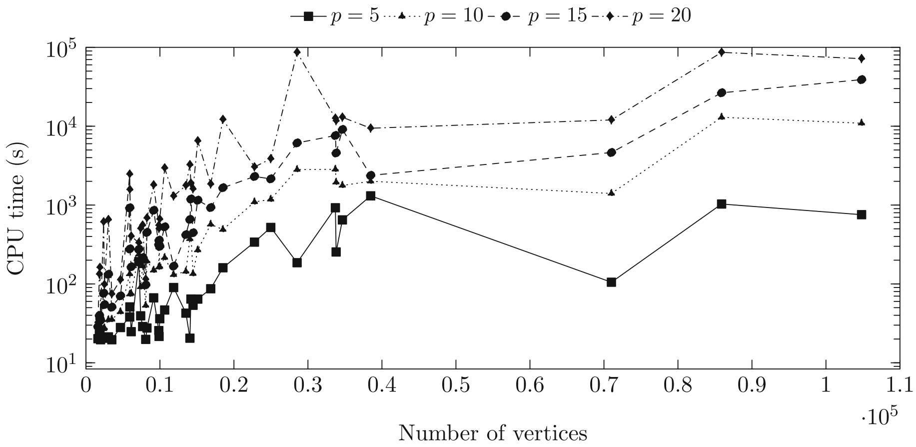

Fig. 2 Impact of increasing the value of p on the computational time ($q = 5$)

First, we can realize that the quality of the solution decreases as the value of p increases. In fact, when the value of p increases, the problem becomes more constrained which typically yields in a decrease of the value of the objective function at optimality. Second, we can realize that the total computational time increases as the size of p increases. The increase in computational time can be explained by two factors. On one hand, when increasing the value of p, the value of the initial lower bound decreases which decreases the number of eliminated vertices. On the other hand, when increasing the value of p, the mathematical model has more variables and is therefore more complex to solve.

4.1.2 Analysis on modifying the value of q while fixing the value of p

In this section, we analyze the impacts of modifying the value of q for a fixed value of p. Therefore, Figs. 4, 5 and 6 illustrate the impact on the average deviation from the best known solution, on the average time (in seconds), and on the average percentage of eliminated vertices, respectively, when modifying the value of q and fixing $p = 5$. Note that the best

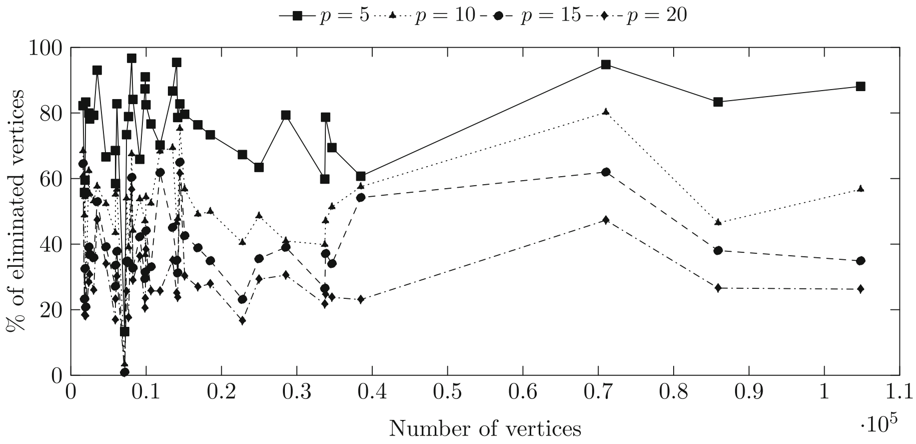

Fig. 3 Impact of increasing the value of p on the percentage of eliminated vertices ($q = 5$)

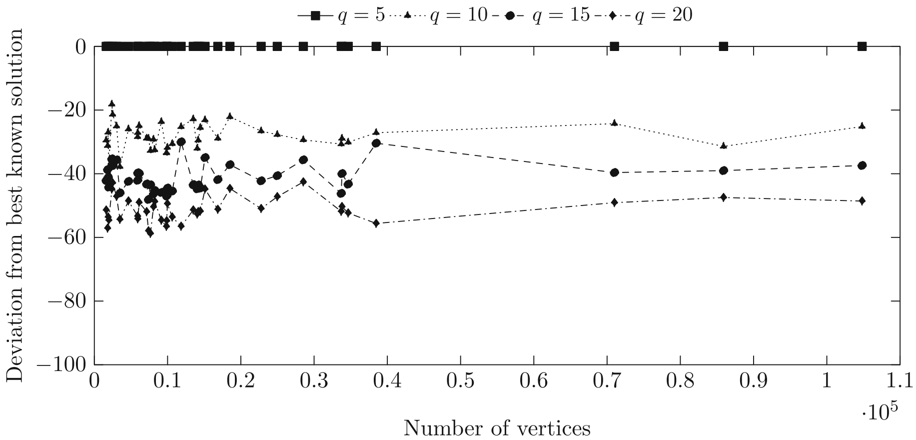

Fig. 4 Impact of increasing the value of q on the deviation from the best known solution ($p = 5$)

known solution is the solution with the highest objective function value at optimality when setting $p = 5$. Similar results are obtained when fixing p to 10, 15, and 20.

First, our analysis shows that the quality of the solution decreases as the value of q increases. In fact, when setting $p = 5$, increasing the value of q from 5 to 10 decreases the quality of the solution by 28.3%. In practice, when increasing the value of q, this makes the problem more constrained and therefore decreases the value of the objective function at optimality. Second, the computational time decreases as the size of q increases. In particular, when q = 5, the algorithm can be as much as four times slower on average (when p = 20). In fact, when increasing the value of q, the number of eliminated vertices also increases because more vertices are within a range of an initially located facility, i.e., there are more initially located facilities, and this reduces the total computational time.

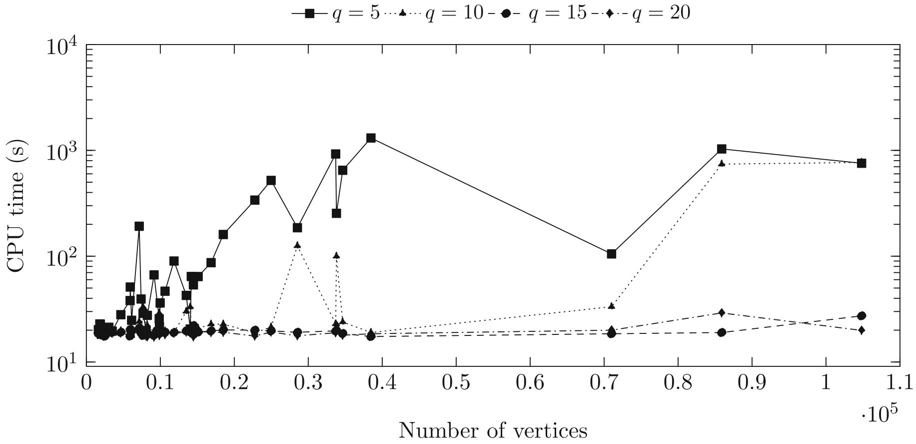

Fig. 5 Impact of increasing the value of q on the total computational time ($p = 5$)

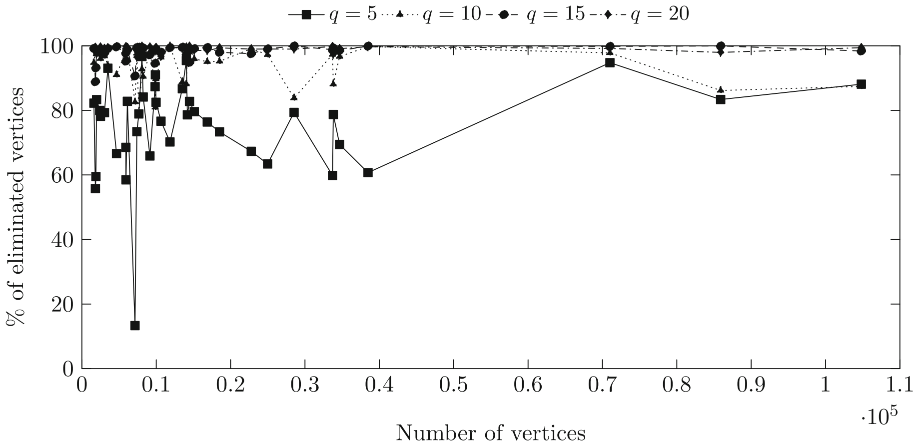

Fig. 6 Impact of increasing the value of q on the percentage of eliminated vertices ($p = 5$)

4.1.3 Analysis on modifying the values of q and p while fixing $q + p$ to a given value

In this section, we fix the value of $q + p$ and analyze the impact of modifying the values of p and q. Figures 7, 8 and 9 illustrate the impact on the average deviation from the best known solution, on the average time (in seconds), and on the average percentage of eliminated vertices, respectively, when modifying the value of p and q and fixing $q + p = 25$. Similar results are obtained when fixing $q + p$ to 15, 20, 30, and 35.

First, our analysis shows that the quality of the solution increases as the size of q decreases and size of p increases. In fact, for all instances and for a fixed value of $q + p$, increasing the size of q always decreases the quality of the solution. In practice, this shows that the added flexibility gained by having a higher value of p, i.e., locating more facilities at a later stage rather than locating more facilities at an earlier stage, allows to increase the value of the objective function. Second, the computational time increases as the size of q decreases and the size p increases. In fact, when q decreases and p increases, the percentage of eliminated

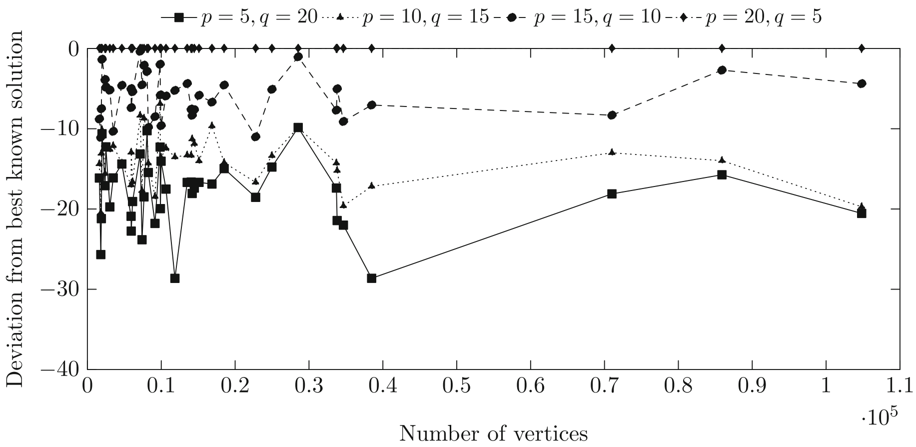

Fig. 7 Impact of modifying the values of p and q on the deviation from the best known solution ($q + p = 25$)

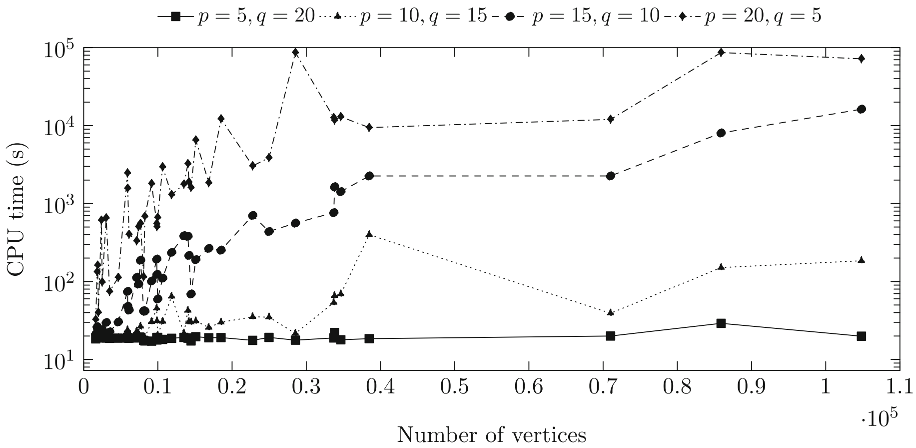

Fig. 8 Impact of modifying the values of p and q on the computational times ($q + p = 25$)

vertices decreases which can then increase the total computational time. In addition, when p increases, the mathematical model has more variables and therefore takes more time to solve.

4.2 Performance of the algorithm

In this section, we determine the performance of the proposed algorithm. First, we compare our algorithm with a naive algorithm to determine how the new features proposed in our algorithm, e.g., the graph reduction technique, impact its computational performance. Second, we analyze how increasing the values of q and p impact the computational performance of our algorithm.

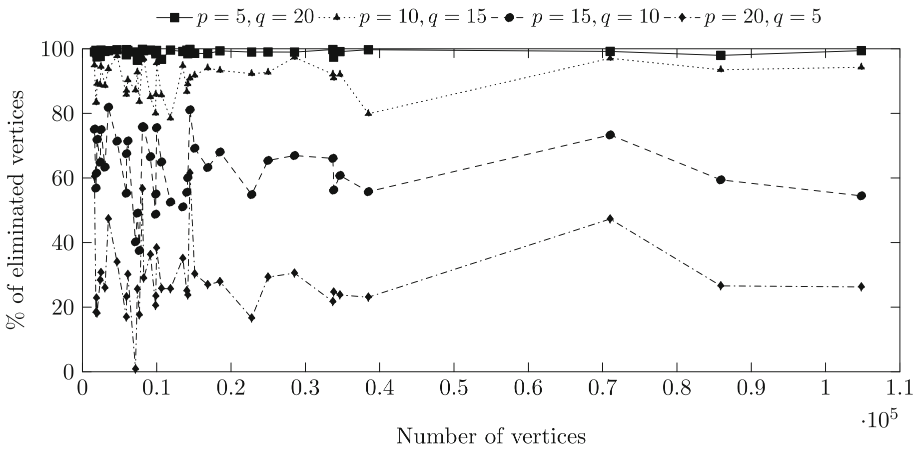

Fig. 9 Impact of modifying the values of p and q on the percentage of eliminated vertices ($q + p = 25$)

4.2.1 Comparison with a naive algorithm

In this section, we assess the performance of the proposed algorithm, denoted as the *ad hoc* algorithm, against a naive implementation of the state-of-the-art algorithm proposed by Contardo [13] for the pDP, denoted as the naive algorithm. The algorithm has been adapted as follows. When building the initial sufficiently refined clustering, the vertices in Q are in singleton clusters. We fix the binary location decisions associated with these clusters to one (i.e., ensuring that they are consistently chosen across the different stages of the algorithm). The pDP is then solved for $p' = p + q$ with the additional fixing constraints. Note that the naive algorithm does not use our proposed graph reduction technique. As the greedy construction strategy for the set Q is shown to be the most effective in our previous experiments, we only consider this construction strategy for the comparison. The initial set Q is the same for both algorithms and therefore the associated optimal solution values are consistently equivalent. For this reason, we only analyze the impact on the computational times. Note that the naive algorithm did not prove optimality for three problems, and for only two of those three with our *ad hoc* algorithm. In addition, all of the problems solved to optimality by the naive algorithm are also solved to optimality with the *ad hoc* algorithm. We restrict the analysis to the problems that could be solved to proven optimality by both algorithms, which excludes these three problem instances, i.e., instances fyg28534, pla85900 and sra104815 with $p = 20$ and $q = 5$. Detailed results are presented in “Appendix C”.

Tables 4 and 5 show the average proportional increase on the total computational time for the different values of p and q for instances where $|N| \leq 10{,}000$ and for instances where $|N| > 10{,}000$, respectively. The average proportional increase has been computed as $\text{Sec}_{\text{naive}}/\text{Sec}_{\textit{ad hoc}}$, where $\text{Sec}_{\text{naive}}$ and $\text{Sec}_{\textit{ad hoc}}$ are the total time in seconds to solve the instance with the naive and the *ad hoc* algorithms, respectively. We also show more detailed time profiles in Figs. 10, 11, 12 and 13 for the following configurations of (p, q): (5, 5), (5, 20), (20, 5), and (20, 20). For conciseness reasons, we do not show all configurations, but believe that the four identified ones are representative of the results.

We can easily see that independently on the values of p and q, our algorithm is always faster by a factor of at least 1.2 on average and at most 596.2 on average. By considering the average total computational time over all instances, our algorithm is faster by a factor of five.

Table 4 Average proportion increase of time with the naive algorithm ($|N| \leq 10{,}000$)

	$p = 5$	$p = 10$	$p = 15$	$p = 20$
$q = 5$	7.3	2.4	1.5	1.2
$q = 10$	10.9	6.6	4.1	2.7
$q = 15$	14.5	10.8	7.8	5.1
$q = 20$	15.9	13.4	10.0	7.4

Table 5 Average proportion increase of time with the naive algorithm ($|N| > 10{,}000$)

	$p = 5$	$p = 10$	$p = 15$	$p = 20$
$q = 5$	32.0	4.6	2.3	1.6
$q = 10$	161.0	40.5	6.4	2.9
$q = 15$	517.2	131.9	27.4	12.9
$q = 20$	596.4	190.9	80.2	38.6

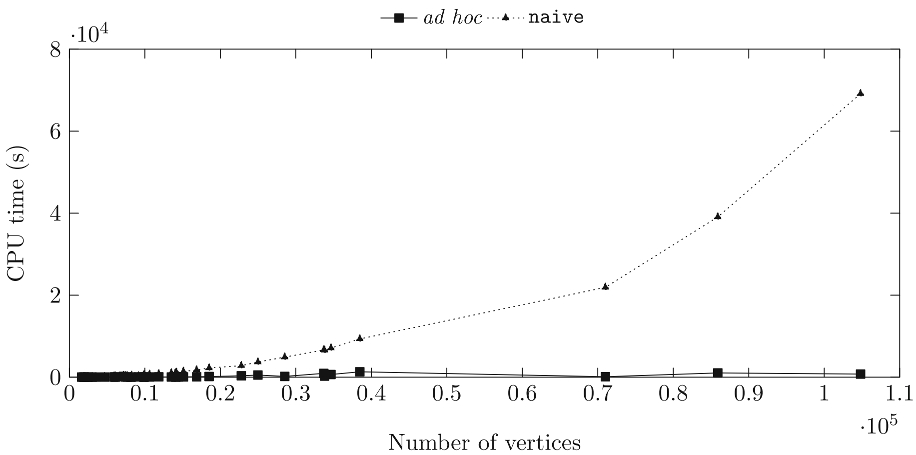

Fig. 10 Time profiles of the naive and the *ad hoc* algorithms ($p = 5, q = 5$)

Table 6 Number of instances solved to optimality within the prescribed time limit

	$p = 10$	$p = 20$	$p = 40$	$p = 80$
$q = 10$	21	21	19	7
$q = 20$	21	21	21	12
$q = 40$	21	21	21	20
$q = 80$	21	21	21	21

We can also note that increasing the value of q and decreasing the value of p is favorable to our algorithm as compared to the naive algorithm. This can also be observed in the time profiles depicted in Figs. 10, 11, 12 and 13. In addition, the differences become more apparent as the sizes of the instances grow. Our algorithm, including our proposed graph reduction technique, is most of the time orders of magnitude faster than the naive setting.

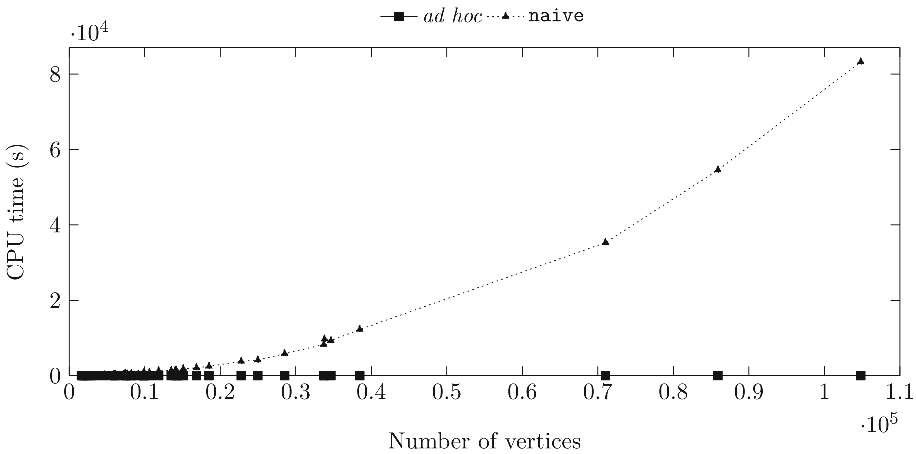

Fig. 11 Time profiles of the naive and *ad hoc* algorithms ($p = 5, q = 20$)

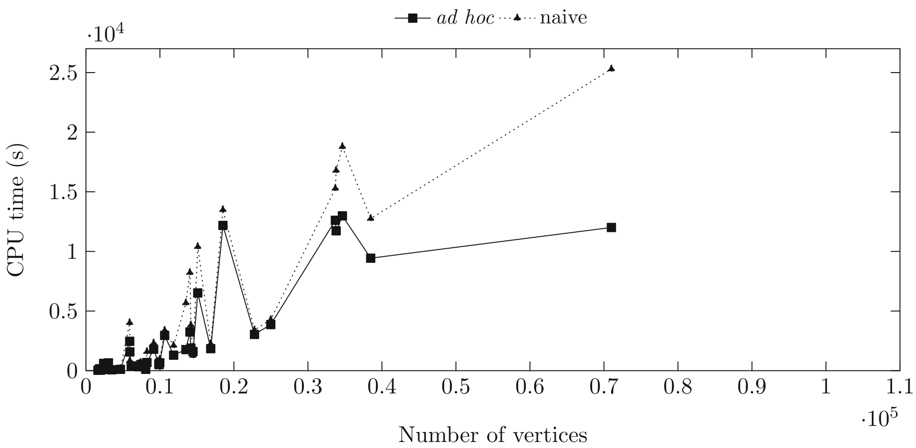

Fig. 12 Time profiles of the naive and *ad hoc* algorithms ($p = 20, q = 5$)

Table 7 Average CPU time (s)

	$p = 10$	$p = 20$	$p = 40$	$p = 80$
$q = 10$	33.9	99.1	5905.2	6926.9
$q = 20$	19.9	33.7	216.5	4463.4
$q = 40$	35.0	37.2	54.3	866.4
$q = 80$	34.7	35.3	39.6	115.7

4.2.2 Performance of the algorithm with increasing values of *q* and *p*

In this section, we assess the performance of our algorithm for increasing values of q and p. For these results, we have tested values of $q, p \in \{10, 20, 40, 80\}$ and have selected a subset of instances, that is, those with $|N| \leq 10{,}000$. Tables 6 and 7 present, for each tested value of q and p, the number of instances solved within the prescribed time limit as well as the

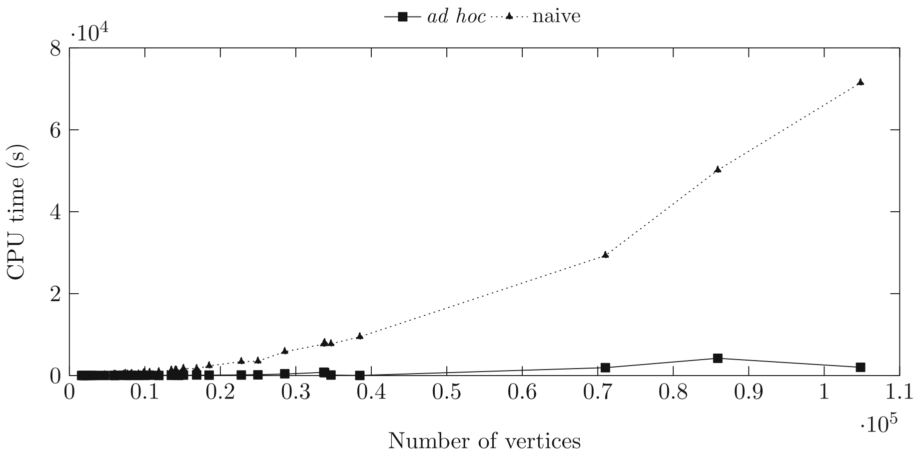

Fig. 13 Time profiles of the naive and the *ad hoc* algorithms ($p = 20$, $q = 20$)

average computational time for these instances, respectively. Note that there are 21 instances with $|N| \leq 10{,}000$. Detailed results are presented in "Appendix D".

Our results show that when increasing the value of p, less instances are solved to optimality and the average computational time increases. In fact, when $p = \{10, 20\}$, all instances are solved to optimality within the prescribed time limit and the average computational time is under one minute. On the other hand, when $p = \{40, 80\}$, not all instances are solved for lower values of q. In particular, when $p = 80$, only seven and 12 instances out of 21 are solved to optimality with $q = 10$ and $q = 20$, respectively. In addition, the average computational time are more than one hour when $p = 40$ and $q = 10$, and when $p = 80$ and $q = \{10, 20\}$. These results can be explained by the fact that for lower values of q, less vertices are eliminated with our graph reduction technique. We can also explain these results as, for a fixed value of q, increasing the value of p increases the complexity of the algorithm. Interestingly, our results show that for higher values of q and p, i.e., when q and p are equal to 80, all instances can be solved with an average computational time limit of less than two minutes. Therefore, our algorithm seems to perform well for higher values of q, independently of the value of p, while the computational complexity of our algorithm increases for higher values of p and lower values of q.

5 Conclusions

In this paper, we have introduced the `c-pDP` and have implemented an *ad hoc* adaptation of the exact decremental clustering proposed by Contardo [13]. The particularity of the `c-pDP` allows us to implement a graph reduction technique which uses valid lower bounds found through a greedy heuristic construction algorithm. We have shown that this graph reduction technique allows to reduce the total computational time and is therefore important for this problem. We show that for a fixed value of q, increasing the value of p decreases the quality of the solution and increases the total computational time. For a fixed value of p, increasing the value of q decreases the quality of the solution, but also decreases the total computational time. In addition, we show that for a fixed value of $q + p$ as the sizes of p increases and q decreases, the quality of the solutions are better even though the total computational time increases. This suggests that when designing a network, it is more interesting to allow for more flexibility at a later stage, i.e., locating less facilities now to allow locating more facilities later on. From an algorithmic viewpoint, we show that the different enhancements introduced in this article are very efficient to reduce the computational time compared to a naive adaptation of Contardo [13]'s algorithm for the pDP to the conditional case.

Acknowledgements The authors thank the Canadian Natural Sciences and Engineering Research Council (NSERC) under Discovery Grants 2017-06106 and 2020-06311 for its financial support. We also thank the GERAD for providing access to its computing infrastructure. Finally, we thank the referees for their valuable comments.

A Impact of the choice of construction strategy for the set Q

In this section, we present three construction strategies (greedy, optimal and random) for the initial set Q. The greedy strategy consists of selecting the set Q in an iterative manner such that, at every iteration, one vertex is added to the solution according to its impact on the objective function. The optimal strategy consists of solving a pDP by setting $p = q$ using the algorithm proposed by Contardo [13]. Note that we use one of the optimal solutions even though there might be alternative optimal solutions if there is a lot of symmetry. The third strategy consists of selecting the set Q randomly.

Table 8 presents summarized results for the construction strategies of the set Q. For each construction strategy (greedy, optimal and random), we report the average deviation in percentage from the best known solution computed as $(z^* - z^*_{best})/z^*_{best}$, where z^* is the optimal solution found according to a given initial set Q and z^*_{best} is the best solution for a given instance independently on how Q is generated (*Average* $\Delta\, z^*$ *(%)*), the worst deviation in percentage from the best known solution (*Worst (%)*), the number of best known solutions (*# Best*), the average computational time in seconds (*Average CPU time (s)*), the number of instances solved to optimality within the prescribed time limit (*# Solved*), and the average percentage of eliminated vertices with our graph reduction technique (*Average elim. (%)*). In addition, Figs. 14, 15 and 16 present the average time, the average deviation with respect to the best known solution value, and the average percentage of eliminated vertices according to the construction strategy as well as the number of instances, respectively.

By analyzing the results, we can note that randomly constructing the set Q yields the worst solutions (− 74.4% on average compared to less than − 2% on average for the two other strategies). In addition, the total computational time is the slowest which is due to the symmetry between the solutions because the value of the initial lower bound is the worst one. By comparing the greedy and the optimal construction strategies, one can realize that the greedy construction strategy yields better results for our set of instances. In fact, it yields the best average deviation from the best known optimal solution (− 1.4% on average compared to − 1.9% on average with the optimal construction) and its worst solution is better than the worst solutions with the two other construction strategies. In addition, this strategy has the lowest computational time and the highest number of instances solved to optimality within the prescribed time limit. Finally, our graph reduction technique performs best with this strategy. With our set of instances, greedily constructing the set Q outperforms both the optimal construction and the random construction as it yields the best solutions and has the lowest computational time. In practice, obtaining the set Q with a greedy construction algorithm is much faster than obtaining the set Q by solving to optimality the pDP. Therefore, it seems more practical the generate the set Q greedily.

Table 8 Impact of the construction strategy for the set Q

	Greedy	Optimal	Random
Average Δz^* (%)	− 1.4	− 1.9	− 74.4
Worst (%)	− 12.8	− 24.0	− 99.5
# Best	368	280	0
Average CPU time (s)	1095.2	1929.5	9067.1
# Solved	638	543	543
Average elim. (%)	74.2	68.5	21.4

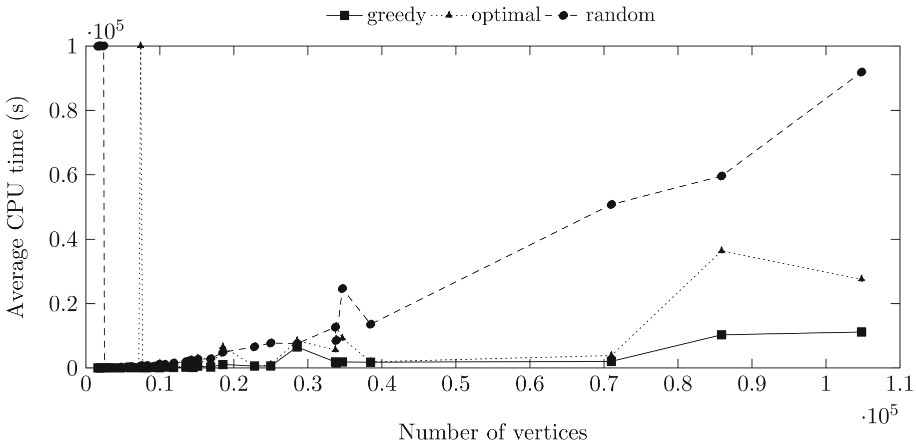

Fig. 14 Impact of the construction algorithm on the average time

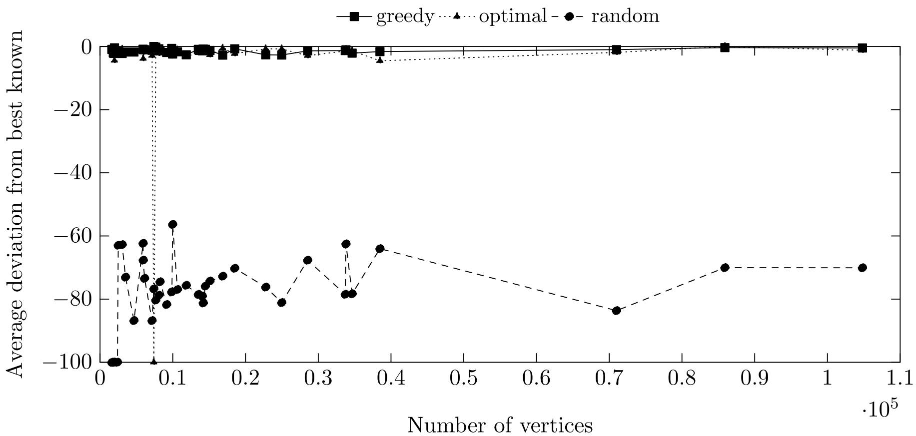

Fig. 15 Impact of the construction algorithm on the average deviation from the best known solution

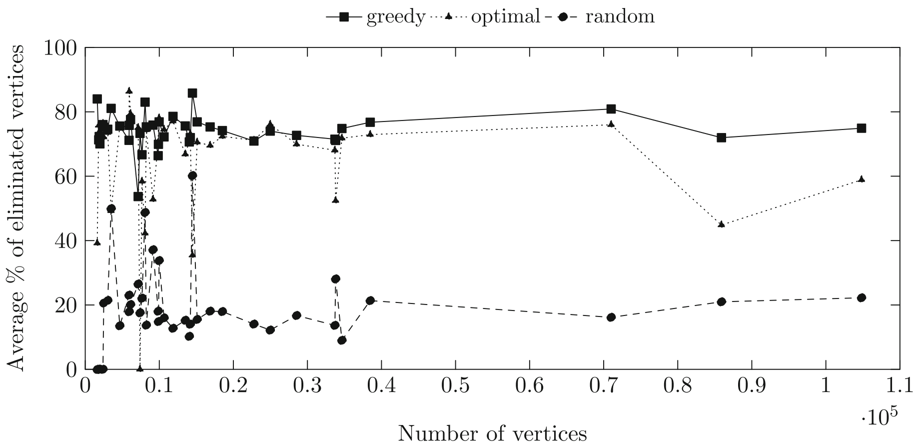

Fig. 16 Impact of the construction algorithm on the percentage of eliminated vertices

B Detailed results

In this section, we presented detailed results. Tables 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23 and 24 present the detailed results for the values of $p = \{5, 10, 15, 20\}$ and $q = \{5, 20, 15, 20\}$. In each table, the first column contains the name of the instance (*Instance*). Then, for each construction strategy for the set Q, i.e., greedy construction, optimal construction, and random construction, we report the total computational time in seconds (*Sec*), the optimal solution value (z^*), and the number of eliminated vertices with our graph reduction technique (*Elim*). If no solution was found within the prescribed time limit, the cells are left blank.

Table 9 Detailed results with $p = 5$ and $q = 5$

Instance	Greedy			Optimal			Random		
	Sec	z^*	Elim	Sec	z^*	Elim	Sec	z^*	Elim
ar9152	66.6	7329	6029	3.4	7142	5759	537.1	985	2979
bby34656	648.5	342	24,064	288.4	330	26,815	25,691.6	12	117
bm33708	924.6	3482	20,169	623.3	3649	21,340	4793.3	1275	11,133
brd14051	20.5	2287	13,410	151.7	2268	8926	2738.3	241	784
ca4663	28.0	17,154	3106	12.6	19,090	2972	149.4	3022	1202
ch71009	105.1	12,483	67,295	20.0	12,549	69,084	11,501.4	4131	34,320
d15112	64.2	6239	12,030	58.4	6676	11,563	2081.7	1689	2109
d15112-2500	20.5	6077	1954	2.4	6556	1858			
d18512	160.3	2258	13,577	61.4	2347	15,031	6196.3	177	258
eg7146	191.9	2599	952	0.6	2821	6696	505.3	218	1409
ei8246	27.6	1291	6938	44.5	1265	5726	94.1	753	4492
fi10639	46.7	3451	8154	0.3	3649	10,459	785.3	860	2501
fyg28534	185.8	303	22,644	116.6	296	23,295			
gr9882	21.6	2448	8994	8.4	2382	8245	913.6	490	1171
ho14473	53.6	1300	11,979	20.1	1254	5222	16.1	768	13,410
it16862	86.9	2803	12,880	95.4	2873	12,502	2058.8	703	3392
ja9847	25.6	5045	8601	29.4	4524	7508	963.7	738	1142
kz9976	36.2	7516	8230	1.4	7910	9376	177.1	3635	4952
mo14185	64.3	2538	11,152	4.0	2639	13,362	54.0	1590	11,300
mu1979	19.6	2057	1649	0.3	2066	1716			
nu3496	19.6	1424	3253	0.1	1366	2316	102.3	85	1226

Table 9 continued

Instance	Greedy			Optimal			Random		
	Sec	z^*	Elim	Sec	z^*	Elim	Sec	z^*	Elim
pba38478	1309.1	365	23,360	0.2	399	38,392	14,720.5	79	4337
pcb3038	21.3	1169	2409	1.2	1297	2568	52.1	518	910
pla33810	254.2	232,371	26,614	403.4	247,329	23,938	1177.2	132,847	20,304
pla7397	39.4	223,567	5429				37.2	132,687	5050
pla85900	1031.3	312,216	71,608	819.2	313,348	73,068	14,737.2	135,329	42,531
pm8079	19.8	1202	7812	0.7	1207	4473	332.2	118	3467
pr2392	20.6	3991	1913	0.6	4577	2114			
rl11849	90.0	5559	8320	0.2	6122	11,725	640.9	2023	3774
rl1889	22.8	5208	1124	0.1	5768	1859			
rl5915	38.1	5527	4054	4.7	5137	4943	20.4	3963	4593
rl5934	51.1	5378	3470	0.2	5915	5790	450.3	800	345
rw1621	20.2	492	1333	0.7	474	630			
sra104815	755.4	564	92,352	4128.1	602	80,723	96,546.1	134	15,017
sw24978	520.3	3553	15,837	8.6	3931	23,729	9656.0	337	794
tz6117	24.8	3239	5065	5.8	3300	5184	263.1	896	1391
u1817	22.9	830	1013	0.1	867	1790			
usa13509	42.6	117,178	11,711	148.1	121,128	8126	316.6	54,638	7090
vm22775	339.0	2489	15,329	465.4	2567	13,714	3023.8	854	5367
ym7663	28.8	2565	6046				32.8	1143	5884

Table 10 Detailed results with $p = 5$ and $q = 10$

	Greedy			Optimal			Random		
Instance	Sec	z^*	Elim	Sec	z^*	Elim	Sec	z^*	Elim
ar9152	19.4	5594	8987	2.5	5048	6035	619.7	574	3134
bby34656	23.9	239	33,495	0.1	225	34,643	13,743.5	51	3349
bm33708	22.9	2414	32,791	3.3	2354	32,759	14,490.3	399	3338
brd14051	33.1	1554	12,387	0.9	1593	13,644	2463.9	243	1276
ca4663	19.7	12,698	4243	0.8	12,207	4209	402.3	555	189
ch71009	33.3	9454	69,446	513.2	8828	59,987	99,173.0	421	1641
d15112	20.8	4797	14,501	0.4	4392	14,861	5126.5	251	239
d15112-2500	18.8	4778	2400	0.1	4424	2424			
d18512	22.8	1758	17,620	0.4	1639	18,292	8477.5	73	125
eg7146	24.1	1848	5901	0.2	1883	6994	312.9	238	2460
ei8246	22.2	871	7460	0.2	890	8149	1187.0	92	284
fi10639	19.6	2395	10,257	0.7	2227	10,301	1564.8	331	801
fyg28534	125.2	214	23,915	33.8	222	25,634	5524.3	72	6525
gr9882	34.2	1628	7998	0.1	1784	9820	1080.9	307	957
ho14473	18.8	968	14,452	0.8	854	6731	412.1	224	8963
it16862	22.8	1993	16,029	4.2	2166	15,743	2579.0	437	3141

Table 10 continued

	Greedy			Optimal			Random		
Instance	Sec	z^*	Elim	Sec	z^*	Elim	Sec	z^*	Elim
ja9847	29.0	3360	8651	0.1	3275	9822	1266.9	364	924
kz9976	24.9	5128	8951	1.3	5217	9548	172.5	2485	5749
mo14185	19.2	1790	13,764	10.5	1766	12,684	2389.8	278	1692
mu1979	19.2	1280	1929	0.1	1267	1939			
nu3496	19.7	886	3284	0.8	880	2029	79.7	118	1402
pba38478	19.0	266	38,381	14.8	238	36,605	12,232.1	76	7589
pcb3038	19.0	876	2941	0.2	894	2892	77.7	289	678
pla33810	99.8	165,217	29,788	40.6	156,765	30,726	3834.2	69,657	15,291
pla7397	19.4	158,899	7223				709.4	20,881	444
pla85900	741.1	214,133	74,012				79,080.2	46,681	12,171
pm8079	21.0	851	7493	0.4	786	4670	554.3	33	3176
pr2392	19.1	3265	2372	0.4	2909	2197			
rl11849	19.1	4155	11,837	1.0	3639	11,428	1544.6	861	1555
rl1889	17.3	3802	1867	0.3	3532	1733			
rl5915	19.0	3957	5825	0.1	3575	5840	241.4	1371	1528
rl5934	18.9	3920	5910	0.4	3538	5716	29.7	2788	4238
rw1621	18.7	347	1536	0.1	324	803			
sra104815	769.6	422	91,625	225.7	414	98,566	101,152.4	98	17,681
sw24978	21.4	2568	24,284	1.9	2655	24,384	6600.8	485	3447
tz6117	17.7	2432	6076	0.3	2327	5953	461.3	353	617
u1817	18.9	571	1797	0.2	540	1711			
usa13509	30.1	90,431	11,999	1.8	82,279	12,908	2577.6	10,925	853
vm22775	18.9	1827	22,572	0.5	1684	22,472	6267.3	297	2516
ym7663	33.2	1725	6011				242.7	480	3266

Table 11 Detailed results with $p = 5$ and $q = 15$

	Greedy			Optimal			Random		
Instance	Sec	z^*	Elim	Sec	z^*	Elim	Sec	z^*	Elim
ar9152	17.9	3967	8896	0.1	3996	6709	414.5	1019	3866
bby34656	18.6	194	34,141	2.0	183	33,958	17,560.6	32	2164
bm33708	19.7	1874	33,300				21,227.2	158	887
brd14051	19.3	1264	13,668	6.6	1217	13,011	2708.4	193	1237
ca4663	19.1	9879	4648	0.1	10,107	4616	400.8	472	221
ch71009	18.5	7536	70,876	1.1	7074	70,596	68,138.4	889	7115
d15112	19.1	4060	14,978	0.2	3751	15,009	3156.6	613	1462
d15112-2500	17.5	3797	2462	0.1	3780	2464			
d18512	20.1	1419	18,139	0.5	1386	18,266	20,800.6	11	15
eg7146	21.0	1476	6477	1.1	1438	6552	246.5	243	2792
ei8246	18.7	706	8184	0.4	693	7999	598.1	235	1617
fi10639	19.1	1881	10,422	0.3	1903	10,461	497.3	648	4510
fyg28534	19.1	195	28,517	0.7	167	28,146	5347.4	66	7515
gr9882	19.2	1298	9757	0.2	1213	9770	1982.5	83	259
ho14473	22.1	721	13,726	0.4	734	6866	758.1	94	7909
it16862	19.7	1630	16,772	3.6	1621	15,873	5895.3	101	306
ja9847	21.1	2696	9312	2.0	2431	9160	597.4	635	3031
kz9976	18.7	4171	9869	0.2	4206	9864	1322.0	647	1087

Table 11 continued

Instance	Greedy			Optimal			Random		
	Sec	z^*	Elim	Sec	z^*	Elim	Sec	z^*	Elim
mo14185	19.2	1433	13,881	0.9	1375	13,727	2808.2	186	1362
mu1979	18.2	1148	1952	0.1	1047	1959			
nu3496	19.0	768	3464	0.1	729	2336			
pba38478	17.4	254	38,445	0.1	193	38,447	14,106.9	61	6134
pcb3038	19.2	752	2969	0.1	694	2976	160.4	92	126
pla33810	19.8	139,533	33,553				19,324.5	16,049	1276
pla7397	19.0	116,087	7361				684.6	15,713	894
pla85900	19.0	190,403	85,854				81,063.8	36,501	12,439
pm8079	19.3	645	7979	0.2	655	4849	215.9	137	4214
pr2392	19.2	2581	2345	0.1	2486	2354			
rl11849	19.0	3892	11,779	0.2	3301	11,741	2993.7	227	191
rl1889	19.9	3066	1760	0.1	3185	1868			
rl5915	17.6	3202	5782	0.1	3044	5812	533.2	448	328
rl5934	20.4	3242	5652	0.1	3067	5907	415.5	692	732
rw1621	18.7	285	1609	0.1	275	855			
sra104815	27.3	353	103,136				16,240.0	171	63,251
sw24978	19.7	2110	24,739	2.0	2040	24,214	3107.6	719	8841
tz6117	19.5	1944	6036	0.3	1917	5917	539.1	248	712
u1817	19.2	509	1615	0.1	472	1808			
usa13509	19.8	66,284	13,450	0.2	62,636	13,414	1931.8	12,417	2377
vm22775	20.0	1438	22,220	0.1	1425	22,723	6085.5	284	3179
ym7663	17.8	1449	7627				1215.2	58	249

Table 12 Detailed results with $p = 5$ and $q = 20$

Instance	Greedy			Optimal			Random		
	Sec	z^*	Elim	Sec	z^*	Elim	Sec	z^*	Elim
ar9152	17.2	3326	9114	0.2	3244	6616	287.4	1022	4693
bby34656	17.9	163	34,350				12,031.0	47	6074
bm33708	18.9	1677	33,627	1.4	1535	33,023	16,240.5	264	3137
brd14051	18.8	1082	13,989	0.5	1030	13,826	4370.8	79	356
ca4663	18.8	8826	4649	0.1	8133	4621	483.5	271	163
ch71009	20.0	6356	70,428	0.9	6447	70,522	34,246.4	1363	22,313
d15112	19.6	3442	14,898	0.4	3064	14,843	4849.1	265	369
d15112-2500	18.9	3471	2490	0.2	3064	2423			
d18512	19.1	1250	18,393	1.5	1094	17,924	7811.0	87	336
eg7146	18.9	1249	7075	0.1	1148	7053	670.5	97	1175
ei8246	17.3	662	8197	0.1	614	8186	875.1	134	961
fi10639	18.1	1601	10,292	0.2	1525	10,503	1056.6	429	2079
fyg28534	17.7	174	28,241				12,749.5	25	1955
gr9882	17.8	1107	9728	0.4	1068	9645	2075.5	69	222
ho14473	17.3	627	14,445	0.1	627	7062	2050.2	17	7390
it16862	19.0	1368	16,610	0.3	1434	16,674	2907.1	297	3078
ja9847	19.1	2192	9759	0.2	2108	9731	1541.7	194	801
kz9976	19.2	3801	9924	0.1	3560	9883	1231.8	638	1266
mo14185	18.9	1220	13,967	0.1	1163	14,105	3883.2	98	447
mu1979	19.1	934	1929	0.1	837	1938			
nu3496	18.7	650	3475	0.1	591	2340			
pba38478	18.5	162	38,364				21,603.0	33	2740

Table 12 continued

Instance	Greedy			Optimal			Random		
	Sec	z^*	Elim	Sec	z^*	Elim	Sec	z^*	Elim
pcb3038	18.9	618	3013	0.1	619	3019	94.2	193	595
pla33810	22.2	115,413	32,928				5191.4	46,174	13,526
pla7397	19.4	94,022	7133						
pla85900	29.1	164,010	84,142				125,028.7	17,749	3188
pm8079	17.6	596	8065	0.2	537	4808	518.7	37	3278
pr2392	19.4	2202	2335	0.1	2057	2354			
rl11849	18.7	2417	11,800	0.3	2501	11,638	1670.1	657	1536
rl1889	19.2	2416	1865	0.1	2340	1837			
rl5915	19.1	2580	5805	0.1	2554	5874	310.3	878	1238
rl5934	18.8	2465	5918	0.1	2366	5908	630.5	310	234
rw1621	18.4	239	1605	0.1	240	860			
sra104815	19.9	290	104,204				126,650.3	60	14,945
sw24978	19.2	1875	24,728	1.4	1752	24,437	8426.5	285	2782
tz6117	18.8	1651	6085	0.1	1627	6076	580.4	217	568
u1817	19.2	356	1808	0.1	345	1766			
usa13509	19.2	56,954	13,396						
vm22775	17.6	1221	22,542	1.4	1249	22,225	10,244.1	105	719
ym7663	19.2	1058	7520				760.2	148	1052

Table 13 Detailed results with $p = 10$ and $q = 5$

Instance	Greedy			Optimal			Random		
	Sec	z^*	Elim	Sec	z^*	Elim	Sec	z^*	Elim
ar9152	148.9	6148	4919	67.2	5764	3378	76.2	2965	5855
bby34656	1772.5	283	17,792	5175.8	279	11,417	9868.9	89	3416
bm33708	2839.3	2681	13,433	4361.2	2672	11,722	15,189.3	308	1501
brd14051	373.1	1748	6546	712.1	1745	4900	1652.2	420	1755
ca4663	44.3	13,916	2438	44.5	14,950	1715	144.9	4779	374
ch71009	1403.3	10,278	56,932	5411.9	10,107	35,984	35,563.4	1929	14,305
d15112	268.9	5230	8578	722.1	5592	5452	293.8	3379	7736
d15112-2500	27.6	5243	1383	19.6	5468	1008			
d18512	488.8	1886	9242	976.2	1976	7465	405.8	1305	9457
eg7146	237.3	2377	241	15.2	2070	5318	556.8	151	734
ei8246	195.2	1048	3642	177.9	1030	3531	376.6	398	1685
fi10639	215.9	2674	5579	288.7	2644	3837	1352.0	369	579
fyg28534	2818.1	254	11,683	2510.1	257	11,284	1895.5	170	9123
gr9882	293.4	1793	3630	155.5	1851	5006	1646.8	100	178
ho14473	134.7	1015	10,887	137.7	1026	3186	526.0	180	7986
it16862	570.3	2298	8279	645.6	2203	7037	322.4	2253	9964
ja9847	163.0	3907	4633	76.4	3604	5510	576.2	999	2018
kz9976	165.9	6239	5425	190.9	5933	4849	92.0	5304	6803
mo14185	431.5	2027	6779	272.9	2005	7705	4805.2	52	24

Table 13 continued

Instance	Greedy			Optimal			Random		
	Sec	z^*	Elim	Sec	z^*	Elim	Sec	z^*	Elim
mu1979	34.7	1473	466	13.7	1419	476			
nu3496	35.5	1071	2014	13.6	998	1027	17.7	529	2525
pba38478	2005.4	285	22,124	4990.7	305	14,757	532.8	235	25,722
pcb3038	35.1	995	1614	40.7	1045	972	28.5	921	2134
pla33810	1947.4	195,151	15,908	4292.8	200,849	8614	492.4	181,080	25,089
pla7397	91.7	174,952	3994				318.2	42,427	1688
pla85900	12,980.6	260,855	39,922	21,041.8	269,374	33,295	41,199.2	85,374	18,500
pm8079	53.3	917	5452	42.2	917	2318	346.7	85	3281
pr2392	26.9	3422	1491	21.1	3662	918			
rl11849	130.8	4213	8094	358.8	4702	4583	748.9	1680	2796
rl1889	27.7	3974	1028	12.0	4385	705			
rl5915	133.8	4372	2571	79.3	4378	2493	382.5	907	430
rl5934	74.1	4112	3273	103.8	4428	2292	231.6	1516	944
rw1621	23.6	387	1109	2.8	393	359			
sra104815	10,912.5	498	59,428	23,832.2	488	45,701	124,421.9	73	5901
sw24978	1182.5	2788	12,128	1673.7	2932	9040	1767.6	1473	8391
tz6117	75.5	2709	3467	47.6	2791	3612	254.7	802	1114
u1817	27.3	623	887	12.4	656	689			
usa13509	144.0	94,515	9372	483.9	92,853	5138	1260.5	29,797	1829
vm22775	1095.3	1973	9207	1178.0	1973	9261	5275.3	384	1709
ym7663	168.7	1872	2996				126.6	703	3647

Table 14 Detailed results with $p = 10$ and $q = 10$

Instance	Greedy			Optimal			Random		
	Sec	z^*	Elim	Sec	z^*	Elim	Sec	z^*	Elim
ar9152	33.4	4516	7511	16.2	4396	5218	1224.4	83	2449
bby34656	337.8	213	27,443	1.2	216	34,214	15,175.6	41	2303
bm33708	268.7	2051	26,583	147.8	2087	28,279	8634.0	554	6231
brd14051	113.9	1384	10,363	31.6	1363	11,767	1765.9	318	2152
ca4663	23.7	11,488	3620	4.8	10,563	3675	74.2	2900	2049
ch71009	581.5	8009	61,090	1776.9	7901	52,639	82,183.2	492	2268
d15112	46.5	4164	13,155	5.9	4275	13,968	2000.6	1032	2468
d15112-2500	20.5	4033	2111	1.1	3932	2164			
d18512	97.1	1498	14,824	5.6	1501	17,526	2014.3	569	4517
eg7146	72.3	1615	3373	23.2	1618	4693	279.9	244	2143
ei8246	34.1	736	6660	6.2	833	7227	349.6	355	2079
fi10639	34.5	2066	8718	12.6	2027	9070	1670.5	227	506
fyg28534	177.4	199	23,233	165.2	191	23,187	11,618.9	26	729
gr9882	63.1	1409	7262	2.9	1445	9139	1626.6	97	245
ho14473	37.7	777	12,775	7.3	762	5903	416.1	209	8559
it16862	57.6	1680	14,043	254.1	1850	10,345	1086.5	684	6071
ja9847	72.4	2760	6647	24.2	2745	7590	1276.2	232	618
kz9976	36.2	4428	8224	5.2	4886	9041			

Table 14 continued

Instance	Greedy			Optimal			Random		
	Sec	z^*	Elim	Sec	z^*	Elim	Sec	z^*	Elim
mo14185	71.6	1551	10,901	95.8	1489	10,312	1776.1	337	2191
mu1979	19.8	1188	1828	0.4	1117	1805			
nu3496	20.5	784	3150	2.5	814	1747	23.5	291	2331
pba38478	20.1	254	37,994	34.2	232	35,592	17,285.0	46	2664
pcb3038	21.4	793	2449	2.1	783	2546	130.7	148	204
pla33810	340.1	146,697	26,119	39.7	152,675	30,488	2628.4	74,216	15,840
pla7397	48.9	128,577	5050				407.1	32,063	1420
pla85900	2112.6	197,511	66,261				47,012.1	59,572	19,705
pm8079	28.5	720	6685	10.8	694	3499	198.0	142	4146
pr2392	20.1	2695	1975	1.8	2713	1923			
rl11849	19.5	3892	11,581	2.0	3543	11,204	1223.8	1019	1289
rl1889	20.6	3295	1429	0.6	3293	1615			
rl5915	28.0	3465	4852	0.2	3475	5773	110.8	1819	2453
rl5934	26.1	3412	4854				384.7	700	560
rw1621	20.6	300	1425	0.3	296	703			
sra104815	7696.7	380	69,898	1123.6	375	89,193	191,920.6	26	1560
sw24978	155.0	2256	19,681	104.6	2249	20,215	6632.3	395	2648
tz6117	23.8	2169	5235	3.5	2004	5397	232.1	680	1441
u1817	21.4	509	1224	0.4	500	1602			
usa13509	178.8	71,317	8553	19.1	72,235	11,713	1956.1	12,623	1622
vm22775	204.8	1526	16,762	59.3	1560	19,409	3583.9	448	4661
ym7663	56.6	1528	5316				450.5	288	1770

Table 15 Detailed results with $p = 10$ and $q = 15$

Instance	Greedy			Optimal			Random		
	Sec	z^*	Elim	Sec	z^*	Elim	Sec	z^*	Elim
ar9152	30.5	3468	7787	0.4	3638	6511	649.3	494	2768
bby34656	69.1	168	31,905	5.2	177	33,550	15,993.8	31	1921
bm33708	53.7	1740	31,045	54.7	1715	30,142	24,292.7	83	283
brd14051	42.5	1125	12,185	19.6	1081	12,183	3287.3	113	524
ca4663	19.0	8835	4560	0.4	9091	4513	384.6	389	206
ch71009	39.2	6753	68,920	44.5	6273	67,828	38,730.1	1363	14,875
d15112	31.1	3552	13,869	2.6	3474	14,276	374.5	1978	8372
d15112-2500	19.6	3471	2362	0.4	3551	2325			
d18512	30.0	1261	17,272	6.0	1244	17,422	3746.4	289	2117
eg7146	23.0	1318	6230	6.8	1335	5699	286.1	177	2445
ei8246	19.2	671	7993	0.9	638	7834	450.0	230	1969
fi10639	30.7	1700	9114	1.7	1687	10,107	1302.5	300	1015
fyg28534	21.6	174	27,780	5.8	161	27,558	9838.4	33	2424
gr9882	31.2	1188	8485	2.3	1156	9198	1027.4	194	1343
ho14473	30.1	669	13,156	2.3	667	6412	975.3	50	7519
it16862	25.7	1487	15,856	4.7	1485	15,846	6468.9	69	173
ja9847	45.4	2327	7881	16.6	2226	8197	3100.6	37	34
kz9976	20.4	3821	9538	2.0	3754	9483	1322.5	485	771

Table 15 continued

Instance	Greedy Sec	Greedy z^*	Greedy Elim	Optimal Sec	Optimal z^*	Optimal Elim	Random Sec	Random z^*	Random Elim
mo14185	30.6	1321	12,639	28.0	1217	11,746	3921.3	75	183
mu1979	19.4	939	1766	1.4	865	1466			
nu3496	19.2	681	3278	0.7	650	2045	52.8	149	1687
pba38478	398.8	188	30,736	75.1	173	34,484	18,457.8	36	2779
pcb3038	19.7	673	2691	0.2	654	2918	153.3	84	113
pla33810	65.7	124,552	30,745				6829.8	42,802	9217
pla7397	21.6	101,560	6861				934.1	7212	200
pla85900	151.0	167,431	80,319				41,341.4	56,929	24,923
pm8079	19.5	596	7806	0.7	580	4605	249.8	105	3888
pr2392	20.8	2242	2126	0.1	2221	2282			
rl11849	64.5	2929	9301	25.6	2685	9812	1911.4	407	561
rl1889	20.9	2665	1577	1.1	2660	1520			
rl5915	23.9	2707	5078	2.3	2728	5321	487.2	440	266
rl5934	22.7	2778	5167				252.2	989	1450
rw1621	19.3	244	1539	0.2	245	794			
sra104815	184.6	293	98,810				29,404.7	138	47,204
sw24978	34.9	1906	23,152	9.5	1874	23,273	10,006.9	186	874
tz6117	21.2	1701	5525	0.7	1671	5783	710.1	85	251
u1817	20.7	381	1514	0.6	395	1536			
usa13509	21.7	59,273	12,803	1.6	57,000	12,990	1864.5	12,758	1824
vm22775	35.3	1249	21,018	1.1	1319	22,335	4633.8	300	3791
ym7663	26.6	1185	6408				618.4	170	1100

Table 16 Detailed results with $p = 10$ and $q = 20$

Instance	Greedy			Optimal			Random		
	Sec	z^*	Elim	Sec	z^*	Elim	Sec	z^*	Elim
ar9152	19.3	3110	9026	0.9	3034	6323	641.4	418	2983
bby34656	32.9	150	33,106				70,925.1	2	34
bm33708	25.4	1623	32,709				14,987.7	261	2748
brd14051	22.2	969	13,313	1.7	962	13,448	2135.5	210	1821
ca4663	20.9	7938	4325	0.2	7397	4509	361.2	414	279
ch71009	301.7	5732	63,007	38.9	5639	68,186			
d15112	20.4	3254	14,652				4390.9	281	408
d15112-2500	19.4	3045	2371	0.2	2889	2380			
d18512	24.2	1155	17,568						
eg7146	20.6	1120	6633	0.2	1081	6929	253.9	169	2722
ei8246	20.8	600	7720	1.0	566	7838	1248.6	59	205
fi10639	22.0	1441	9796	0.9	1406	10,225	866.7	403	2356
fyg28534	79.1	140	25,000				9509.7	31	2752
gr9882	19.8	1050	9464	3.5	970	9023	4160.6	17	20
ho14473	19.2	567	14,261	0.9	555	6709	570.2	112	8248
it16862	22.7	1281	16,057	5.9	1251	15,692	4786.0	123	674
ja9847	21.8	1817	9204	5.6	1940	8854	1508.6	149	569
kz9976	21.4	3265	9327	1.3	3356	9526	612.9	1077	2803

Table 16 continued

Instance	Greedy			Optimal			Random		
	Sec	z^*	Elim	Sec	z^*	Elim	Sec	z^*	Elim
mo14185	25.1	1088	13,193	1.3	1048	13,655	3051.6	122	862
mu1979	20.3	817	1815	0.1	725	1911			
nu3496	19.2	560	3260	0.2	544	2260	57.2	130	1628
pba38478	19.5	156	38,081				5708.1	80	14,234
pcb3038	19.2	563	2912	0.1	595	2983	72.6	215	743
pla33810	72.1	110,345	30,517				15,527.6	18,111	2359
pla7397	19.6	92,032	7097				1988.4	2000	21
pla85900	392.9	149,610	77,213				100,495.3	22,100	5819
pm8079	19.8	516	7772	2.6	501	4299	160.9	137	4535
pr2392	19.2	2007	2257	0.2	1908	2289			
rl11849	19.3	2398	11,779	1.0	2365	11,540	2359.4	289	357
rl1889	19.5	2278	1813	0.1	2264	1815			
rl5915	20.0	2328	5586	0.5	2386	5668	456.0	472	519
rl5934	19.1	2382	5863				222.2	914	1567
rw1621	20.3	206	1558	0.1	207	820			
sra104815	56.3	281	101,773				87,538.8	73	20,448
sw24978	26.1	1658	23,732	2.3	1650	24,191	3225.2	584	8093
tz6117	19.4	1460	5910	0.8	1489	5773	473.3	190	704
u1817	19.2	345	1798	0.1	330	1752			
usa13509	19.9	53,369	13,134				1825.8	9935	2477
vm22775	21.9	1101	21,944	26.4	1081	20,374	8527.3	121	1004
ym7663	20.0	990	7251				1139.1	50	243

Table 17 Detailed results with $p = 15$ and $q = 5$

Instance	Greedy			Optimal			Random		
	Sec	z^*	Elim	Sec	z^*	Elim	Sec	z^*	Elim
ar9152	858.0	4853	3867	337.3	4765	2219	206.7	2017	3907
bby34656	9091.4	234	11,800	15,220.3	237	11,005	2960.1	143	11,698
bm33708	7630.1	2286	8949	6055.4	2288	7841	8589.0	655	3684
brd14051	657.9	1527	4925	2219.5	1499	4288	2095.8	283	787
ca4663	70.5	11,607	1827	40.7	11,607	2261	305.5	765	93
ch71009	4638.1	8736	44,013	16,570.2	8464	20,816	71,455.5	695	2520
d15112	1151.8	4699	6431	4045.6	4610	4072	1733.4	1340	2118
d15112-2500	54.5	4557	921	190.6	4502	701	20.0	3243	1192
d18512	1659.8	1682	6468	2599.7	1631	5977	3491.6	346	1280
eg7146	275.3	1783	67	25.2	1791	4940	256.2	266	1995
ei8246	450.3	867	2696	568.0	863	2749	521.3	281	907
fi10639	529.5	2241	3518	514.2	2274	2713	949.1	523	1221
fyg28534	6134.2	219	11,138	7938.8	220	10,247	1412.2	199	15,323
gr9882	358.5	1524	3119	214.7	1541	4163	436.9	568	2372
ho14473	444.9	884	9407	213.0	891	2280	137.1	434	9947
it16862	930.1	1965	6556	1061.1	1929	5969	2379.2	490	1330
ja9847	317.5	2997	2906	204.4	2975	3565	402.6	1129	2314
kz9976	298.5	4971	4399	327.7	5024	4276	233.9	4777	4901

Table 17 continued

	Greedy			Optimal			Random		
Instance	Sec	z^*	Elim	Sec	z^*	Elim	Sec	z^*	Elim
mo14185	1194.8	1703	4430	480.1	1655	6909	2072.6	298	820
mu1979	34.3	1208	412	16.9	1202	390			
nu3496	50.7	874	1851	22.8	920	671	16.0	567	2534
pba38478	2379.1	261	20,857	6415.1	254	14,067	1962.3	233	23,557
pcb3038	132.6	875	1089	144.4	877	906	43.5	788	1690
pla33810	4579.0	166,242	12,556	8568.4	165,217	5726	976.7	162,113	22,247
pla7397	194.9	142,916	2576				54.0	132,509	5438
pla85900	26,555.8	220,778	32,678	37,165.1	221,516	25,079	4603.2	160,639	50,140
pm8079	96.9	762	4876	86.5	771	1300	125.5	260	4353
pr2392	76.2	3094	936	130.2	3035	807			
rl11849	168.3	3933	7329	712.1	3866	4431	803.3	1567	2217
rl1889	36.3	3587	616	16.6	3590	636			
rl5915	923.0	3770	1608	283.7	3644	1856	142.7	3239	2641
rl5934	278.1	3622	1994				230.8	1386	796
rw1621	29.0	328	1047	4.4	320	395			
sra104815	38,966.7	403	36,589	77,413.5	393	30,552	6026.1	346	74,643
sw24978	2150.4	2473	8884	2392.6	2625	8077	7757.3	324	586
tz6117	164.7	2301	2314	126.8	2246	2942	71.2	2098	3980
u1817	39.9	547	424	18.4	544	628			
usa13509	422.2	80,181	6095	975.7	81,319	4013	3169.3	4663	290
vm22775	2304.6	1787	5275	2013.7	1777	5341	543.6	1584	14,105
ym7663	215.6	1554	2631				536.8	276	961

C Detailed computational time with the naive algorithm

In this section, we present the total computational time for the naive and the *ad hoc* algorithms. Tables 25, 26, 27 and 28 present the detailed results for the values of $p = \{5, 10, 15, 20\}$, respectively. In each table, the first column contains the name of the instance (*Instance*). Then, for each value of q, i.e., $q = \{5, 10, 15, 20\}$, we report the total computational time in seconds used with the proposed *ad hoc* algorithm (*ad hoc*) and with the naive algorithm (*naive*). If no optimal solution was found within the prescribed time limit, the cells are left blank.

Table 18 Detailed results with $p = 15$ and $q = 10$

Instance	Greedy			Optimal			Random		
	Sec	z^*	Elim	Sec	z^*	Elim	Sec	z^*	Elim
ar9152	101.5	3892	6094	54.7	4122	3780	197.6	1285	4693
bby34656	1422.9	190	21,063	66.4	206	31,216	15,775.4	27	1188
bm33708	765.6	1874	22,271	1126.6	1957	20,637	13,589.9	302	1861
brd14051	379.6	1200	7810	215.3	1238	8585	1672.2	346	1600
ca4663	30.3	9837	3329	15.7	9927	2873	124.4	2335	1196
ch71009	2260.1	7117	52,065	3223.8	7259	48,966	46,376.2	1173	9768
d15112	190.9	3890	10,457	77.7	3841	11,611	1955.6	1064	1822
d15112-2500	24.0	3766	1874	11.6	3605	1331	49.2	1443	431
d18512	252.3	1403	12,593	314.9	1348	11,649	554.1	799	9510
eg7146	113.3	1433	2876	61.0	1318	3836	420.5	156	1282
ei8246	41.5	706	6236	68.2	756	4995	1289.6	53	91
fi10639	110.2	1826	6909	117.8	1909	6158	983.1	406	1455
fyg28534	561.2	191	19,101	450.5	178	20,306	12,159.4	18	372
gr9882	123.6	1303	5436	46.1	1323	7366	644.4	321	2078
ho14473	69.2	701	11,738	39.6	724	4892	802.5	75	7541
it16862	266.0	1536	10,662	575.4	1641	7709	1457.6	506	4776
ja9847	193.7	2450	4803	73.7	2460	6228	213.0	1062	4244
kz9976	59.7	3997	7540	99.9	4237	6308	390.4	1618	3210

Table 18 continued

Instance	Greedy Sec	Greedy z^*	Greedy Elim	Optimal Sec	Optimal z^*	Optimal Elim	Random Sec	Random z^*	Random Elim
mo14185	216.3	1365	8528	329.2	1431	6857	1953.8	237	1528
mu1979	21.6	1031	1424	1.3	1009	1618			
nu3496	22.6	695	2863	6.7	755	1427	64.4	134	1396
pba38478	2258.8	211	21,456	699.5	210	28,623	21,986.7	23	760
pcb3038	29.8	730	1924	4.6	724	2302	53.4	332	822
pla33810	1642.0	139,533	19,023	1003.6	137,252	21,929	9500.8	35,384	4090
pla7397	92.8	117,843	3632				579.4	17,205	562
pla85900	8033.0	189,381	51,042				44,002.0	58,320	17,463
pm8079	42.3	645	6133	23.8	663	2937	285.5	85	3570
pr2392	23.9	2553	1552	4.8	2558	1750			
rl11849	236.2	3210	6226	76.1	3304	8174	356.1	1936	4628
rl1889	25.0	2836	1161	4.4	2956	1293			
rl5915	74.7	3099	3269	36.2	3088	3545	204.1	1181	1352
rl5934	48.1	2956	4008				101.8	1659	2267
rw1621	21.3	260	1218	1.4	270	503			
sra104815	16,236.2	349	57,088	2289.4	342	80,550	56,516.1	120	24,788
sw24978	438.2	2088	16,338	408.2	2134	16,481	21,010.3	37	42
tz6117	43.0	1930	4371	32.0	1930	3989	75.6	1070	3047
u1817	26.3	426	1033	2.4	450	1299			
usa13509	387.7	65,374	6897	183.9	67,189	7958	2724.0	5819	613
vm22775	705.0	1334	12,496	327.6	1431	15,348	9392.5	94	309
ym7663	187.8	1271	2877				318.6	333	1918

Table 19 Detailed results with $p = 15$ and $q = 15$

Instance	Greedy			Optimal			Random		
	Sec	z^*	Elim	Sec	z^*	Elim	Sec	z^*	Elim
ar9152	43.0	3221	7359	3.6	3425	5941	791.5	219	2626
bby34656	210.5	160	29,052	29.2	169	32,296	26,030.3	8	154
bm33708	263.4	1639	27,780	114.1	1636	28,933	14,656.8	240	2035
brd14051	143.2	1040	9774	45.3	1051	11,436	3531.1	104	338
ca4663	20.7	8133	4030	0.6	8463	4389	140.3	1604	1200
ch71009	911.9	5976	59,375	212.7	6097	64,589	43,293.6	1013	11,996
d15112	42.2	3347	13,337	18.0	3349	13,281	4308.4	256	328
d15112-2500	20.9	3224	2077	1.1	3312	2098	76.2	695	242
d18512	113.1	1194	14,960	23.3	1196	16,276	4871.7	172	894
eg7146	34.9	1152	5608	21.7	1189	4646	822.5	56	320
ei8246	29.3	640	7008	1.7	627	7664	934.2	90	360
fi10639	75.9	1518	8058	15.7	1607	9041	754.0	490	2556
fyg28534	220.8	148	22,247	40.2	156	25,876	3928.9	69	8298
gr9882	41.0	1104	7830	10.9	1064	8554	684.0	273	1918
ho14473	46.8	617	12,579	10.9	611	5646	369.7	157	8954
it16862	37.3	1325	15,013	34.0	1424	14,190	1090.9	538	5892
ja9847	135.9	2101	5890	26.2	2103	7626	1589.9	142	333
kz9976	46.1	3516	8070	14.1	3489	8299	769.1	903	1846

Table 19 continued

	Greedy			Optimal			Random		
Instance	Sec	z^*	Elim	Sec	z^*	Elim	Sec	z^*	Elim
mo14185	116.0	1148	10,631	39.1	1137	11,251	3264.9	109	365
mu1979	19.9	897	1561	1.8	817	1417			
nu3496	22.5	584	2856	0.9	613	2002	101.5	50	1203
pba38478	549.3	169	29,776	93.0	172	34,264	28,248.4	11	289
pcb3038	21.9	598	2451	1.7	612	2592	106.4	152	317
pla33810	386.2	112,324	26,367				11,345.9	28,636	3982
pla7397	22.9	93,146	6801				471.6	24,167	1153
pla85900	1923.9	151,644	67,592				71,477.0	33,745	10,204
pm8079	29.3	537	6967	10.7	521	3523	328.0	67	3436
pr2392	22.0	2051	1956	0.5	2089	2147			
rl11849	86.7	2479	9243	34.9	2634	9404	770.9	1010	3010
rl1889	21.4	2432	1456	2.1	2442	1353			
rl5915	31.7	2534	4629	7.5	2564	4830	336.7	700	721
rl5934	25.2	2465	5012				470.8	379	280
rw1621	19.6	222	1449	0.5	227	710			
sra104815	791.7	288	92,886				79,848.1	84	19,489
sw24978	120.0	1784	20,497	26.1	1790	22,492	9207.5	170	957
tz6117	29.1	1540	4920	1.9	1615	5539	408.8	240	653
u1817	20.4	356	1489	1.0	364	1485			
usa13509	30.7	55,755	12,110	17.6	55,034	11,840	3050.2	4476	513
vm22775	94.5	1168	19,504	20.2	1250	20,788	6883.8	163	1396
ym7663	42.0	1055	5769				1104.8	49	194

Table 20 Detailed results with $p = 15$ and $q = 20$

Instance	Greedy			Optimal			Random		
	Sec	z^*	Elim	Sec	z^*	Elim	Sec	z^*	Elim
ar9152	22.0	2928	8455	2.7	2936	5992	686.5	300	2840
bby34656	51.7	144	32,349				21,304.7	13	481
bm33708	91.0	1452	30,276	50.3	1384	30,398	6321.0	463	9753
brd14051	39.3	880	12,171	3.0	937	13,142	2818.7	138	642
ca4663	21.9	7147	4003	0.3	7096	4401	243.0	886	605
ch71009	1077.1	5316	57,019	82.0	5493	66,604	66,248.8	589	6065
d15112	67.7	2908	12,597				6027.6	141	140
d15112-2500	21.7	2797	1987	0.7	2739	2252	56.0	890	513
d18512	61.5	1041	16,182				5258.7	144	766
eg7146	37.0	997	5377	7.8	968	5898	556.0	90	959
ei8246	27.2	560	7156	3.4	541	7497	1567.1	28	64
fi10639	40.0	1374	8956	2.0	1366	10,006	2582.5	69	104
fyg28534	121.0	137	24,277				11,581.6	19	1131
gr9882	36.0	919	8239	30.9	935	7682	668.0	279	2218
ho14473	20.6	543	14,003	2.6	541	6419	490.6	112	8435
it16862	48.3	1209	14,862	10.6	1213	15,247	4528.2	120	496
ja9847	30.9	1659	8366	18.7	1662	8285	991.8	283	1268
kz9976	30.1	3005	8564	2.7	3114	9263	419.8	1216	3442
mo14185	41.1	1018	12,428	7.3	1008	13,050	3442.2	89	409
mu1979	21.5	697	1444	0.3	707	1861			
nu3496	20.0	521	3088	0.8	517	2119	69.6	90	1422
pba38478	23.5	152	37,588				21,924.6	23	1580
pcb3038	20.0	534	2714	2.2	535	2423	170.2	61	72

Table 20 continued

Instance	Greedy			Optimal			Random		
	Sec	z^*	Elim	Sec	z^*	Elim	Sec	z^*	Elim
pla33810	201.2	102,026	28,569				22,995.6	6325	418
pla7397	23.2	89,405	6500				579.4	14,000	851
pla85900	1113.3	138,402	73,153				64,295.7	37,857	12,397
pm8079	22.8	461	7321	10.5	468	3596	675.9	17	3154
pr2392	19.7	1814	2096	0.4	1811	2214			
rl11849	19.0	2365	11,754	0.2	2354	11,750	2296.5	253	297
rl1889	19.3	2236	1763	0.4	2158	1741			
rl5915	21.9	2244	5330	0.9	2249	5555	621.5	169	85
rl5934	19.9	2294	5702				505.0	307	285
rw1621	19.4	200	1487	0.2	195	763			
sra104815	239.5	255	98,047				101,425.5	60	14,489
sw24978	56.6	1547	22,518	10.9	1576	23,556	9148.7	189	1046
tz6117	20.8	1402	5696	1.6	1419	5623	633.4	101	382
u1817	19.4	330	1738	0.2	325	1719			
usa13509	32.9	49,907	11,921				3989.0	2379	280
vm22775	60.0	1008	20,227	66.6	1036	19,285	12,847.2	47	129
ym7663	24.6	929	6574				601.7	138	1102

Table 21 Detailed results with $p = 20$ and $q = 5$

	Greedy			Optimal			Random		
Instance	Sec	z^*	Elim	Sec	z^*	Elim	Sec	z^*	Elim
ar9152	1799.5	4253	3319	2043.7	4124	984	467.9	717	2900
bby34656	12,980.8	209	8239	86,403.8	212[3]	8047	5076.7	198	15,137
bm33708	12,611.4	2030	7318	62,650.5	1983	5466	8459.7	601	3840
brd14051	3244.3	1298	3524	4242.1	1312	2607	544.1	776	4126
ca4663	112.8	10,310	1584	75.6	10,310	1179	173.3	1826	542
ch71009	12,007.7	7762	33,585	22,482.8	7765	15,284	22,965.1	2167	18,742
d15112	6517.1	4131	4579	33,350.9	4105	3259	669.0	2382	4586
d15112-2500	98.2	3956	768	708.7	4018	475	60.8	1174	255
d18512	12,180.1	1470	5164	86,402.5	1459[4]	4428	920.6	1397	8829
eg7146	332.2	1438	59	59.6	1396	4278	87.5	447	3464
ei8246	684.7	783	2391	1858.9	777	2111	971.7	84	105
fi10639	2954.4	1941	2742	2014.4	1970	2034	180.9	1213	4649
fyg28534	86,410.4	193[1]	8716	86,403.6	195[5]	8613	2318.3	113	8316
gr9882	503.8	1383	2316	358.3	1400	3487	119.5	926	4243
ho14473	1603.9	759	8908	2033.1	768	1495	128.1	417	9944
it16862	1846.7	1646	4546	2100.6	1708	4295	611.1	1036	5876
ja9847	552.7	2499	2020	392.7	2443	2938	517.0	954	1493
kz9976	663.0	4422	3826	595.6	4497	3423	269.7	4213	5521
mo14185	1892.4	1489	3369	1617.7	1479	4091	615.7	651	4335
mu1979	40.2	1045	359	24.7	1031	358			
nu3496	74.6	775	1655	110.8	771	403	35.4	264	1665
pba38478	9433.9	227	8881	6522.6	230	12,800	4528.3	207	18,552
pcb3038	653.7	770	789	2210.0	772	611	42.0	479	666

Table 21 continued

Instance	Greedy			Optimal			Random		
	Sec	z^*	Elim	Sec	z^*	Elim	Sec	z^*	Elim
pla33810	11,729.3	146,906	8342	86,402.3	147,649[6]	4683	10,930.1	31,501	1810
pla7397	503.9	123,435	1890				306.2	45,123	1125
pla85900	86,408.8	194,637[2]	22,844	86,402.2	197,773[7]	22,621	38,839.0	79,704	14,608
pm8079	114.6	664	4576	151.0	666	869	51.8	591	6059
pr2392	611.2	2656	678	1290.4	2670	616			
rl11849	1299.1	3387	3043	3633.4	3394	3608	1062.6	1137	1025
rl1889	161.7	3066	431	292.0	3083	465			
rl5915	2464.7	3262	1001	1620.7	3225	1421	129.4	1884	1470
rl5934	1571.1	3191	1376				1183.4	2974	2142
rw1621	32.5	285	980	6.3	280	250			
sra104815	71,884.2	365	27,531	86,403.0	367[8]	24,262	48,971.1	159	22,333
sw24978	3866.4	2200	7319	10,324.9	2229	5854	4116.8	709	3074
tz6117	402.7	2040	1843	942.0	2016	2535	588.6	142	209
u1817	133.0	479	334	90.4	479	562			
usa13509	1769.8	68,360	4738	5759.7	70,119	2175	361.9	46,139	5159
vm22775	3041.0	1499	3788	2950.0	1535	3088	1695.2	777	5939
ym7663	556.2	1298	1349				209.4	488	2339

The reported solutions marked with a number are not proven optimal but we give their gap with their lower bound below
[1]Gap of 10.9%
[2]Gap of 18.7%
[3]Gap of 21.1%
[4]Gap of 23.3%
[5]Gap of 9.6%
[6]Gap of 17.6%
[7]Gap of 14.8%
[8]Gap of 27.4%

Table 22 Detailed results with $p = 20$ and $q = 10$

	Greedy			Optimal			Random		
Instance	Sec	z^*	Elim	Sec	z^*	Elim	Sec	z^*	Elim
ar9152	107.4	3532	5677	198.8	3696	2532	1034.9	105	2527
bby34656	2638.8	176	17,529	3031.9	187	14,042	106,159.2	1	10
bm33708	2795.1	1686	18,311	3055.4	1746	15,344	1070.8	1081	20,015
brd14051	1355.6	1127	5589	743.7	1104	6313	785.5	519	3958
ca4663	38.0	8943	2696	31.0	8754	2646	298.4	639	207
ch71009	5911.5	6605	40,813	9992.6	6638	32,464	39,182.1	1216	10,120
d15112	308.9	3498	9762	659.9	3655	6513	1516.7	1427	2198
d15112-2500	29.9	3386	1514	13.6	3475	1178	39.5	1472	631
d18512	1746.0	1264	9976	1219.8	1296	8602	3647.3	289	1130
eg7146	160.4	1200	2319	66.6	1138	3158	185.0	231	2866
ei8246	96.6	657	5085	194.8	663	3211	328.9	310	2009
fi10639	228.1	1650	5774	269.0	1693	4920	1019.7	359	1174
fyg28534	6690.4	161	12,036	3674.6	170	12,212	10,136.5	27	940
gr9882	246.3	1184	4766	212.1	1180	4561	413.0	499	2383
ho14473	134.8	635	10,995	270.3	667	3099	482.0	141	8304
it16862	424.8	1443	9148	956.0	1464	6678	2240.5	380	2000
ja9847	287.4	2241	3403	197.5	2104	4255	49.1	1910	7513
kz9976	201.3	3689	5820	199.1	3789	5109	648.1	1102	1968
mo14185	531.3	1244	6489	599.6	1251	5817	1558.5	283	2314

Table 22 continued

Instance	Greedy			Optimal			Random		
	Sec	z^*	Elim	Sec	z^*	Elim	Sec	z^*	Elim
mu1979	23.0	918	1207	4.3	868	1162			
nu3496	32.7	630	2418	19.9	655	952	31.0	237	1908
pba38478	9253.6	184	20,278	3877.9	195	15,449	4812.8	102	12,392
pcb3038	47.8	652	1631	61.8	678	1263	80.7	218	396
pla33810	3363.2	127,201	14,715	2677.9	131,606	15,637	8764.7	36,361	4308
pla7397	132.1	106,902	3514				627.7	12,650	515
pla85900	15,816.8	173,042	37,704				17,189.7	84,403	34,853
pm8079	98.8	579	5026	77.3	555	1944	82.6	232	4985
pr2392	28.9	2223	1315	18.6	2301	1259			
rl11849	391.5	2848	6145	355.9	3008	4744	1853.7	403	331
rl1889	25.7	2630	1040	10.1	2753	822			
rl5915	95.6	2790	2705	98.5	2821	2708	126.9	1511	1812
rl5934	56.2	2605	3804				313.5	733	629
rw1621	23.8	243	1108	2.6	237	383			
sra104815	26,138.9	313	49,418	25,140.9	315	44,098	158,029.1	33	2318
sw24978	823.6	1888	13,425	1013.5	1959	12,385	11,728.4	97	122
tz6117	72.1	1761	3592	90.7	1769	3219	122.9	847	2206
u1817	27.2	381	992	10.1	423	720			
usa13509	967.2	59,273	6046	678.6	61,276	6243	1037.1	18,880	2904
vm22775	1289.1	1247	10,220	1312.2	1236	9392	2297.0	546	5596
ym7663	250.5	1185	2399	158.1	1124	2765	1179.3	37	51

Table 23 Detailed results with $p = 20$ and $q = 15$

Instance	Greedy			Optimal			Random		
	Sec	z^*	Elim	Sec	z^*	Elim	Sec	z^*	Elim
ar9152	92.8	3134	6058	68.3	3253	3590	469.7	507	3310
bby34656	341.8	153	27,686	389.2	162	27,391	16,555.1	26	1386
bm33708	799.7	1483	23,628	348.6	1576	26,419	16,196.9	180	1107
brd14051	323.3	967	7883	133.9	987	9346	3521.7	88	212
ca4663	25.6	7639	3632	7.2	7713	3658	142.0	1508	1147
ch71009	1637.1	5732	52,662	819.2	5602	58,724	40,719.4	1203	9756
d15112	220.5	3067	10,108	172.4	3028	9938	1933.2	841	2213
d15112-2500	28.5	2984	1535	6.3	2969	1708	73.2	622	230
d18512	271.2	1107	12,533	322.4	1092	11,677	593.2	658	9315
eg7146	57.4	1069	4255	79.1	1069	3179	249.5	170	2297
ei8246	51.6	580	6153	23.9	579	6411	734.2	132	624
fi10639	133.3	1441	6980	66.6	1461	7615	1180.1	297	924
fyg28534	340.8	142	21,211	182.4	150	23,127	8027.7	38	2369
gr9882	128.1	1000	6101	27.2	1017	7699	521.9	321	2378
ho14473	80.8	566	12,184	43.8	574	4656	787.9	60	7590
it16862	87.8	1269	13,521	346.8	1276	9811	3850.8	157	651
ja9847	205.5	1916	4969	156.4	1945	5128	731.7	430	1870
kz9976	95.7	3277	6909	35.2	3313	7674	519.2	1163	2726

Table 23 continued

Instance	Greedy			Optimal			Random		
	Sec	z^*	Elim	Sec	z^*	Elim	Sec	z^*	Elim
mo14185	184.8	1066	9264	169.4	1104	8866	1805.8	232	1925
mu1979	24.0	706	1187	4.6	742	1150			
nu3496	25.2	556	2640	2.7	555	1785			
pba38478	895.9	159	28,818	173.6	162	33,241	15,079.7	41	3610
pcb3038	27.9	544	2148	2.4	580	2504	149.4	79	86
pla33810	633.3	108,904	23,793				6556.3	38,053	7854
pla7397	29.7	92,000	6093				1055.6	4473	106
pla85900	3300.1	144,430	62,260				83,129.9	26,005	5716
pm8079	37.6	490	6380	15.8	485	3417	320.6	60	3390
pr2392	23.2	1937	1707	2.4	1953	1920			
rl11849	103.8	2408	9145	57.3	2529	8857	2528.4	192	127
rl1889	21.8	2322	1384	2.6	2382	1352			
rl5915	43.3	2328	4231	14.3	2418	4492	374.2	560	496
rl5934	32.7	2370	4769				374.7	518	488
rw1621	19.9	201	1362	0.8	210	629			
sra104815	2048.9	264	86,483				188,607.9	18	1103
sw24978	272.8	1582	18,342	127.6	1714	19,881	4264.8	434	5164
tz6117	35.8	1471	4601	8.5	1519	5130	218.4	560	1385
u1817	22.2	339	1413	2.5	345	1338			
usa13509	78.3	53,107	10,355	17.4	51,562	11,854	1456.0	12,658	2265
vm22775	263.8	1051	16,745	391.0	1120	15,240	7737.0	121	736
ym7663	74.3	983	4797	81.4	1017	3805	278.2	287	2386

Table 24 Detailed results with $p = 20$ and $q = 20$

	Greedy			Optimal			Random		
Instance	Sec	z^*	Elim	Sec	z^*	Elim	Sec	z^*	Elim
ar9152	30.4	2618	8010	6.3	2753	5746	630.7	300	2890
bby34656	127.7	139	30,872				19,810.7	14	589
bm33708	815.1	1335	23,444	142.2	1342	28,906	14,399.9	217	2147
brd14051	81.7	838	11,302	28.0	895	11,836	2005.9	206	1488
ca4663	25.0	6692	3702	1.6	6668	4130	256.7	709	434
ch71009	1922.8	4943	51,440	554.6	5269	61,000	61,940.5	587	6293
d15112	119.0	2737	11,953				2989.4	460	902
d15112-2500	23.7	2621	1874	1.7	2678	2070	44.7	1119	631
d18512	99.4	1000	15,173	12.9	1006	17,086	4209.5	177	1310
eg7146	82.8	897	3935	5.9	938	5968	449.8	105	1270
ei8246	76.3	529	5876	10.8	513	7000	768.7	117	666
fi10639	89.7	1268	7666	9.2	1315	9429	1386.3	224	743
fyg28534	438.8	131	22,115				7381.8	35	3817
gr9882	49.4	841	7419	27.1	850	7533	862.8	180	1500
ho14473	29.9	521	13,245	13.1	511	5548	406.7	134	8684
it16862	140.6	1161	12,338	49.9	1163	14,044	3819.8	152	951
ja9847	57.5	1567	7527	69.5	1586	6622	1711.3	93	291
kz9976	40.5	2896	8094	8.4	2983	8681	378.1	1301	3528
mo14185	56.7	947	11,972	29.4	952	11,914	1826.3	217	2071
mu1979	23.0	677	1253	2.7	653	1434			

Table 24 continued

Instance	Greedy			Optimal			Random		
	Sec	z^*	Elim	Sec	z^*	Elim	Sec	z^*	Elim
nu3496	22.6	488	2800	0.8	502	2081	45.1	130	1774
pba38478	26.8	151	37,438				14,194.5	42	4491
pcb3038	21.2	508	2517	4.4	501	2276	57.8	230	848
pla33810	692.8	96,933	25,154				10,561.4	23,410	4351
pla7397	39.2	81,104	5604				1876.9	2000	21
pla85900	4213.0	133,047	62,549				100,595.7	18,310	3665
pm8079	28.9	428	6964	11.4	445	3529	207.5	97	4063
pr2392	20.4	1765	2048	0.8	1779	2078			
rl11849	19.6	2343	11,609	0.7	2297	11,557	1943.9	334	491
rl1889	20.1	2124	1547	0.4	2100	1706			
rl5915	25.5	2110	5013	3.5	2194	5164	159.1	1011	1893
rl5934	23.6	2193	5194				808.0	80	47
rw1621	19.7	182	1423	0.4	185	711			
sra104815	2018.0	242	86,999				57,130.8	87	27,605
sw24978	167.7	1474	20,194	56.7	1514	21,879	7313.8	239	1884
tz6117	24.5	1344	5195	8.0	1335	5054	336.7	261	1042
u1817	19.4	321	1653	0.5	310	1628			
usa13509	61.0	47,096	10,855				2704.2	4649	822
vm22775	128.5	955	18,609	137.3	1005	18,196	16,332.7	33	78
ym7663	33.9	860	6173	5.3	851	6857	592.9	140	934

Table 25 Time profiles of the naive and *ad hoc* algorithms ($p = 5$)

	$q = 5$		$q = 10$		$q = 15$		$q = 20$	
Instance	Ad hoc	Naive	Ad hoc	Naive	Ad hoc	Naive	Ad hoc	Naive
ar9152	66.6	291.4	19.4	291.6	17.9	379.4	17.2	385.5
bby34656	648.5	7108.8	23.9	8221.0	18.6	8304.9	17.9	9290.0
bm33708	924.6	6659.0	22.9	7068.8	19.7	8214.4	18.9	8268.1
brd14051	20.5	1028.6	33.1	1464.2	19.3	1306.5	18.8	1429.8
ca4663	28.0	125.9	19.7	150.7	19.1	155.4	18.8	167.9
ch71009	105.1	21,903.2	33.3	25,703.1	18.5	28,974.8	20.0	35,281.5
d15112	64.2	1406.2	20.8	1458.0	19.1	1592.0	19.6	1811.1
d15112-2500	20.5	66.7	18.8	61.9	17.5	70.3	18.9	70.7
d18512	160.3	2242.4	22.8	2337.4	20.1	2497.6	19.1	2486.1
eg7146	191.9	398.4	24.1	295.0	21.0	380.3	18.9	320.3
ei8246	27.6	423.7	22.2	482.1	18.7	565.7	17.3	538.8
fi10639	46.7	680.4	19.6	655.9	19.1	786.6	18.1	811.4
fyg28534	185.8	4891.5	125.2	5949.3	19.1	5360.3	17.7	5842.3
gr9882	21.6	552.4	34.2	728.5	19.2	713.0	17.8	752.9
ho14473	53.6	291.0	18.8	313.4	22.1	396.3	17.3	394.5
it16862	86.9	1751.9	22.8	1881.2	19.7	1873.2	19.0	2114.2
ja9847	25.6	448.7	29.0	624.9	21.1	671.7	19.1	661.0
kz9976	36.2	622.5	24.9	665.0	18.7	739.7	19.2	762.5
mo14185	64.3	1155.6	19.2	1197.6	19.2	1314.8	18.9	1425.1
mu1979	19.6	44.2	19.2	44.9	18.2	42.4	19.1	46.8
nu3496	19.6	46.0	19.7	58.4	19.0	57.9	18.7	59.2
pba38478	1309.1	9329.2	19.0	9611.5	17.4	8158.0	18.5	12,259.6
pcb3038	21.3	80.3	19.0	91.2	19.2	89.5	18.9	98.9
pla33810	254.2	6553.3	99.8	7884.3	19.8	8058.4	22.2	9689.0
pla7397	39.4	351.1	19.4	346.9	19.0	452.9	19.4	599.2
pla85900	1031.3	39,068.7	741.1	53,222.4	19.0	49,334.1	29.1	54,530.6
pm8079	19.8	118.0	21.0	183.6	19.3	174.2	17.6	177.2
pr2392	20.6	58.2	19.1	62.2	19.2	68.0	19.4	71.3
rl11849	90.0	816.6	19.1	840.4	19.0	837.2	18.7	1272.2
rl1889	22.8	46.4	17.3	43.6	19.9	50.0	19.2	51.9
rl5915	38.1	248.7	19.0	242.8	17.6	259.8	19.1	308.4
rl5934	51.1	243.3	18.9	242.8	20.4	258.5	18.8	320.8
rw1621	20.2	26.8	18.7	25.7	18.7	25.4	18.4	26.1
sra104815	755.4	69,119.7	769.6	71,240.5	27.3	73,444.2	19.9	83,220.3
sw24978	520.3	3686.5	21.4	3913.2	19.7	3852.7	19.2	4139.1
tz6117	24.8	245.5	17.7	266.3	19.5	286.5	18.8	310.5
u1817	22.9	42.9	18.9	46.2	19.2	45.5	19.2	49.2
usa13509	42.6	1031.4	30.1	1135.7	19.8	1311.4	19.2	1391.8
vm22775	339.0	2801.4	18.9	3057.9	20.0	3000.5	17.6	3803.0
ym7663	28.8	295.1	33.2	400.8	17.8	363.3	19.2	443.0

Table 26 Time profiles of the naive and *ad hoc* algorithms ($p = 10$)

	$q = 5$		$q = 10$		$q = 15$		$q = 20$	
Instance	Ad hoc	Naive	Ad hoc	Naive	Ad hoc	Naive	Ad hoc	Naive
ar9152	148.9	305.7	33.4	313.5	30.5	382.0	19.3	333.0
bby34656	1772.5	7558.7	337.8	7517.3	69.1	8049.8	32.9	8992.0
bm33708	2839.3	7328.2	268.7	6957.4	53.7	7483.8	25.4	6955.4
brd14051	373.1	1183.2	113.9	1352.2	42.5	1379.0	22.2	1329.3
ca4663	44.3	145.7	23.7	118.3	19.0	133.4	20.9	148.5
ch71009	1403.3	20,580.6	581.5	24,159.0	39.2	24,813.4	301.7	31,975.0
d15112	268.9	1484.9	46.5	1454.5	31.1	1613.6	20.4	1522.2
d15112-2500	27.6	64.1	20.5	62.1	19.6	67.3	19.4	71.5
d18512	488.8	2267.9	97.1	2232.1	30.0	2341.6	24.2	2315.1
eg7146	237.3	245.4	72.3	257.4	23.0	272.7	20.6	289.3
ei8246	195.2	474.7	34.1	497.5	19.2	491.3	20.8	535.5
fi10639	215.9	709.0	34.5	650.5	30.7	713.9	22.0	800.0
fyg28534	2818.1	6074.0	177.4	5516.5	21.6	4522.7	79.1	6151.0
gr9882	293.4	607.4	63.1	657.5	31.2	654.9	19.8	647.9
ho14473	134.7	359.1	37.7	344.7	30.1	346.8	19.2	359.5
it16862	570.3	1613.3	57.6	1775.8	25.7	1673.4	22.7	1986.4
ja9847	163.0	482.6	72.4	576.6	45.4	625.9	21.8	645.9
kz9976	165.9	549.4	36.2	648.5	20.4	674.9	21.4	677.2
mo14185	431.5	1272.2	71.6	1228.8	30.6	1166.6	25.1	1270.4
mu1979	34.7	44.1	19.8	35.9	19.4	40.6	20.3	42.6
nu3496	35.5	57.2	20.5	59.2	19.2	58.3	19.2	61.6
pba38478	2005.4	9604.1	20.1	7330.5	398.8	10,045.3	19.5	10,871.3
pcb3038	35.1	84.8	21.4	87.3	19.7	89.6	19.2	95.8
pla33810	1947.4	7280.2	340.1	7769.5	65.7	7511.5	72.1	8473.5
pla7397	91.7	398.4	48.9	435.9	21.6	468.6	19.6	477.8
pla85900	12,980.6	41,836.4	2112.6	47,443.3	151.0	46,639.5	392.9	51,261.7
pm8079	53.3	167.4	28.5	179.3	19.5	163.1	19.8	179.6
pr2392	26.9	65.5	20.1	68.1	20.8	64.6	19.2	66.2
rl11849	130.8	937.7	19.5	796.7	64.5	964.2	19.3	1061.7
rl1889	27.7	57.6	20.6	44.1	20.9	52.9	19.5	51.5
rl5915	133.8	315.1	28.0	257.4	23.9	282.0	20.0	294.5
rl5934	74.1	266.7	26.1	230.7	22.7	273.0	19.1	280.9
rw1621	23.6	29.8	20.6	29.2	19.3	28.0	20.3	29.3
sra104815	10,912.5	64,531.1	7696.7	69,347.3	184.6	72,591.8	56.3	70,570.2
sw24978	1182.5	3778.1	155.0	3676.5	34.9	3776.4	26.1	3802.9
tz6117	75.5	268.0	23.8	237.5	21.2	253.5	19.4	307.9
u1817	27.3	49.1	21.4	44.7	20.7	48.8	19.2	46.5
usa13509	144.0	1055.6	178.8	1451.7	21.7	1155.9	19.9	1364.8
vm22775	1095.3	2976.6	204.8	3095.8	35.3	3143.4	21.9	3159.9
ym7663	168.7	314.8	56.6	342.0	26.6	353.4	20.0	424.4

Table 27 Time profiles of the naive and *ad hoc* algorithms ($p = 15$)

	$q = 5$		$q = 10$		$q = 15$		$q = 20$	
Instance	Ad hoc	Naive	Ad hoc	Naive	Ad hoc	Naive	Ad hoc	Naive
ar9152	858.0	679.2	101.5	340.3	43.0	359.4	22.0	318.2
bby34656	9091.4	16,154.8	1422.9	8341.4	210.5	7748.4	51.7	8267.3
bm33708	7630.1	10,562.7	765.6	6965.5	263.4	7092.4	91.0	7358.6
brd14051	657.9	1293.2	379.6	1671.9	143.2	1428.4	39.3	1365.0
ca4663	70.5	133.5	30.3	137.0	20.7	128.1	21.9	149.4
ch71009	4638.1	24,469.5	2260.1	26,286.9	911.9	28,453.1	1077.1	29,831.6
d15112	1151.8	1823.9	190.9	1489.5	42.2	1514.4	67.7	1672.6
d15112-2500	54.5	86.7	24.0	65.1	20.9	70.1	21.7	77.2
d18512	1659.8	2651.2	252.3	2123.7	113.1	2290.1	61.5	2512.3
eg7146	275.3	260.7	113.3	254.4	34.9	270.6	37.0	241.7
ei8246	450.3	521.3	41.5	487.1	29.3	495.1	27.2	486.8
fi10639	529.5	903.3	110.2	760.7	75.9	780.1	40.0	743.9
fyg28534	6134.2	8915.5	561.2	5179.8	220.8	5670.0	121.0	5703.1
gr9882	358.5	717.8	123.6	666.6	41.0	636.1	36.0	639.9
ho14473	444.9	530.4	69.2	342.6	46.8	352.2	20.6	365.4
it16862	930.1	1831.7	266.0	1827.3	37.3	1673.7	48.3	1781.9
ja9847	317.5	600.7	193.7	597.3	135.9	684.9	30.9	617.8
kz9976	298.5	675.0	59.7	637.4	46.1	667.0	30.1	707.4
mo14185	1194.8	1551.8	216.3	1303.0	116.0	1264.3	41.1	1289.2
mu1979	34.3	41.6	21.6	36.7	19.9	40.3	21.5	51.0
nu3496	50.7	76.5	22.6	59.0	22.5	61.4	20.0	57.3
pba38478	2379.1	8965.4	2258.8	9458.6	549.3	9389.7	23.5	9998.1
pcb3038	132.6	129.4	29.8	92.5	21.9	101.5	20.0	95.1
pla33810	4579.0	8775.7	1642.0	7478.3	386.2	8139.6	201.2	8256.6
pla7397	194.9	520.3	92.8	524.3	22.9	524.4	23.2	410.8
pla85900	26,555.8	66,566.2	8033.0	46,521.9	1923.9	47,950.9	1113.3	48,827.4
pm8079	96.9	185.8	42.3	177.1	29.3	173.5	22.8	177.7
pr2392	76.2	105.4	23.9	59.4	22.0	71.1	19.7	70.0
rl11849	168.3	968.3	236.2	995.9	86.7	1110.5	19.0	971.0
rl1889	36.3	50.4	25.0	52.3	21.4	49.6	19.3	47.8
rl5915	923.0	1151.9	74.7	280.4	31.7	278.3	21.9	267.7
rl5934	278.1	324.4	48.1	257.4	25.2	261.8	19.9	269.7
rw1621	29.0	33.0	21.3	29.1	19.6	27.9	19.4	28.5
sra104815	38,966.7	76,520.1	16,236.2	72,592.8	791.7	64,476.0	239.5	67,003.0
sw24978	2150.4	3733.7	438.2	3388.6	120.0	3648.4	56.6	3750.4
tz6117	164.7	307.2	43.0	249.7	29.1	277.8	20.8	273.8
u1817	39.9	60.0	26.3	45.8	20.4	45.9	19.4	46.8
usa13509	422.2	1449.6	387.7	1377.7	30.7	1243.4	32.9	1294.1
vm22775	2304.6	3065.4	705.0	3543.6	94.5	3023.7	60.0	3372.0
ym7663	215.6	417.9	187.8	415.4	42.0	384.5	24.6	393.7

Table 28 Time profiles of the naive and *ad hoc* algorithms ($p = 20$)

	$q = 5$		$q = 10$		$q = 15$		$q = 20$	
Instance	Ad hoc	Naive	Ad hoc	Naive	Ad hoc	Naive	Ad hoc	Naive
ar9152	1799.5	2314.1	107.4	301.2	92.8	347.5	30.4	356.8
bby34656	12,980.8	18,784.6	2638.8	8067.5	341.8	7768.0	127.7	7740.9
bm33708	12,611.4	15,287.7	2795.1	8236.2	799.7	7670.6	815.1	7677.7
brd14051	3244.3	8223.2	1355.6	1534.6	323.3	1434.5	81.7	1350.2
ca4663	112.8	161.7	38.0	123.6	25.6	139.4	25.0	156.3
ch71009	12,007.7	25,290.6	5911.5	29,321.6	1637.1	26,838.4	1922.8	29,306.5
d15112	6517.1	10,397.4	308.9	1805.6	220.5	1656.7	119.0	1732.3
d15112-2500	98.2	133.0	29.9	79.0	28.5	77.2	23.7	81.8
d18512	12,180.1	13,492.2	1746.0	2598.7	271.2	2358.0	99.4	2375.0
eg7146	332.2	325.7	160.4	284.6	57.4	243.0	82.8	277.3
ei8246	684.7	1593.4	96.6	486.5	51.6	483.2	76.3	495.4
fi10639	2954.4	3380.1	228.1	697.7	133.3	764.9	89.7	758.0
fyg28534			6690.4	6790.2	340.8	5358.8	438.8	5830.3
gr9882	503.8	748.9	246.3	704.3	128.1	627.4	49.4	700.7
ho14473	1603.9	1264.5	134.8	400.8	80.8	419.4	29.9	304.8
it16862	1846.7	2256.4	424.8	1717.9	87.8	1601.0	140.6	1685.0
ja9847	552.7	715.7	287.4	591.1	205.5	670.8	57.5	623.6
kz9976	663.0	945.2	201.3	708.4	95.7	672.3	40.5	699.8
mo14185	1892.4	3821.6	531.3	1298.0	184.8	1342.7	56.7	1232.8
mu1979	40.2	44.7	23.0	37.0	24.0	47.6	23.0	45.6
nu3496	74.6	81.9	32.7	62.3	25.2	58.0	22.6	62.2
pba38478	9433.9	12,756.2	9253.6	12,363.4	895.9	11,006.8	26.8	9451.3
pcb3038	653.7	598.2	47.8	110.2	27.9	105.1	21.2	94.9
pla33810	11,729.3	16,791.0	3363.2	8575.8	633.3	7655.9	692.8	8036.5
pla7397	503.9	686.9	132.1	484.6	29.7	471.6	39.2	525.7
pla85900			15,816.8	49,604.0	3300.1	47,023.4	4213.0	50,145.4
pm8079	114.6	189.7	98.8	189.6	37.6	174.3	28.9	183.5
pr2392	611.2	294.0	28.9	73.9	23.2	74.2	20.4	66.1
rl11849	1299.1	2112.3	391.5	1050.3	103.8	1067.2	19.6	911.8
rl1889	161.7	88.5	25.7	49.2	21.8	51.4	20.1	47.6
rl5915	2464.7	4018.6	95.6	311.2	43.3	287.4	25.5	278.4
rl5934	1571.1	835.6	56.2	266.3	32.7	260.4	23.6	262.6
rw1621	32.5	48.4	23.8	29.7	19.9	29.8	19.7	31.0
sra104815	71,884.2		26,138.9	72,847.4	2048.9	66,830.4	2018.0	71,468.4
sw24978	3866.4	4281.3	823.6	3659.6	272.8	3782.4	167.7	3500.8
tz6117	402.7	353.2	72.1	259.1	35.8	277.0	24.5	273.4
u1817	133.0	92.8	27.2	52.7	22.2	50.6	19.4	49.6
usa13509	1769.8	5681.3	967.2	1644.6	78.3	1243.2	61.0	1323.2
vm22775	3041.0	3410.7	1289.1	3486.9	263.8	3225.2	128.5	3399.1
ym7663	556.2	628.7	250.5	441.3	74.3	407.0	33.9	360.5

D Detailed computational time with increasing values of *q* and *p*

In this section, we present the total computational time for the results obtained with increasing values of q and p, that is $q, p \in \{10, 20, 40, 80\}$, and for the instances with $|N| \leq 10{,}000$. Tables 29, 30, 31 and 32 present the detailed results for the values of $p = \{10, 20, 40, 80\}$, respectively. In each table, the first column contains the name of the instance (*Instance*). Then, for each value of q, i.e., $q = \{10, 20, 40, 80\}$, we report the total computational time in seconds. If no optimal solution was found within the prescribed time limit, the cells are left blank.

Table 29 Time profiles of the algorithm ($p = 10$)

Instance	$q = 10$	$q = 20$	$q = 40$	$q = 80$
ar9152	33.4	19.3	34.9	35.4
ca4663	23.7	20.9	34.9	34.2
d15112-2500	20.5	19.4	35.0	34.7
eg7146	72.3	20.6	35.3	34.7
ei8246	34.1	20.8	34.9	34.7
gr9882	63.1	19.8	35.4	35.0
ja9847	72.4	21.8	35.9	34.8
kz9976	36.2	21.4	35.9	35.6
mu1979	19.8	20.3	34.9	35.0
nu3496	20.5	19.2	34.1	34.2
pcb3038	21.4	19.2	35.2	34.6
pla7397	48.9	19.6	34.8	34.6
pm8079	28.5	19.8	35.5	35.4
pr2392	20.1	19.2	34.2	34.9
rl1889	20.6	19.5	34.6	34.2
rl5915	28.0	20.0	34.2	34.2
rl5934	26.1	19.1	35.0	34.0
rw1621	20.6	20.3	34.5	34.1
tz6117	23.8	19.4	35.8	35.8
u1817	21.4	19.2	34.7	34.4
ym7663	56.6	20.0	34.3	34.1

Table 30 Time profiles of the algorithm ($p = 20$)

Instance	$q = 10$	$q = 20$	$q = 40$	$q = 80$
ar9152	107.4	30.4	36.3	35.3
ca4663	38.0	25.0	39.2	34.9
d15112-2500	29.9	23.7	34.6	35.0
eg7146	160.4	82.8	42.6	35.4
ei8246	96.6	76.3	36.3	35.9
gr9882	246.3	49.4	40.8	36.1
ja9847	287.4	57.5	45.5	35.8
kz9976	201.3	40.5	37.8	36.5
mu1979	23.0	23.0	36.2	34.7
nu3496	32.7	22.6	34.8	34.9
pcb3038	47.8	21.2	36.1	34.9
pla7397	132.1	39.2	35.8	35.7
pm8079	98.8	28.9	36.3	36.6
pr2392	28.9	20.4	34.9	34.7
rl1889	25.7	20.1	35.5	34.8
rl5915	95.6	25.5	36.5	34.5
rl5934	56.2	23.6	36.6	35.7
rw1621	23.8	19.7	35.0	35.2
tz6117	72.1	24.5	38.2	35.3
u1817	27.2	19.4	34.5	34.3
ym7663	250.5	33.9	37.0	35.7

Table 31 Time profiles of the algorithm ($p = 40$)

Instance	$q = 10$	$q = 20$	$q = 40$	$q = 80$
ar9152	13,135.5	175.7	64.0	37.5
ca4663	120.4	119.5	63.1	37.1
d15112-2500		113.5	39.0	37.0
eg7146	439.8	279.0	86.6	61.4
ei8246	8283.6	390.7	50.4	40.9
gr9882	13,410.8	1006.1	82.3	41.0
ja9847	3479.6	569.0	99.5	41.5
kz9976	6499.9	279.9	60.7	43.6
mu1979	78.7	55.7	40.8	37.9
nu3496	157.7	73.4	42.1	35.8
pcb3038	6816.5	106.2	42.5	36.3
pla7397	17,682.1	158.2	49.3	42.8
pm8079		143.6	51.5	36.6
pr2392	2487.3	70.9	39.5	36.3
rl1889	183.1	51.3	40.1	36.0
rl5915	23,679.3	302.8	56.0	39.0
rl5934	2158.0	152.6	43.7	41.3
rw1621	60.4	39.4	35.8	35.4
tz6117	2754.3	132.0	59.0	37.2

Table 31 continued

Instance	$q = 10$	$q = 20$	$q = 40$	$q = 80$
u1817	349.3	50.6	37.0	35.6
ym7663	10,423.1	276.4	58.1	40.9

Table 32 Time profiles of the algorithm ($p = 80$)

Instance	$q = 10$	$q = 20$	$q = 40$	$q = 80$
ar9152			1625.5	57.3
ca4663	18,850.2	2024.5	693.8	62.1
d15112-2500		11,723.7	100.0	50.8
eg7146	14,491.5	13,702.3		827.7
ei8246			961.0	82.2
gr9882			3867.9	106.4
ja9847			5782.2	178.4
kz9976			821.7	145.6
mu1979	337.4	163.4	82.1	57.1
nu3496	6202.2	2218.8	184.8	49.0
pcb3038		6504.3	253.5	51.1
pla7397		12,771.0	900.0	92.9
pm8079			171.0	66.5
pr2392		2391.0	101.4	49.6
rl1889	3578.1	167.5	97.8	44.4
rl5915			292.5	119.9
rl5934			331.3	105.6
rw1621	110.3	65.4	43.1	38.5
tz6117		1629.0	390.4	72.1
u1817	4918.6	199.7	61.8	48.3
ym7663			566.8	124.3

References

1. Aloise, D., Contardo, C.: A sampling-based exact algorithm for the solution of the minimax diameter clustering problem. J. Glob. Optim. **71**, 613–630 (2018)
2. Belotti, P., Labbé, M., Maffioli, F., Ndiaye, M.M.: A branch-and-cut method for the obnoxious p-median problem. 4OR **5**(4), 299–314 (2006)
3. Berman, O., Drezner, Z.: A new formulation for the conditional p-median and p-center problems. Oper. Res. Lett. **36**(4), 481–483 (2008)
4. Berman, O., Simchi-Levi, D.: Conditional location problems on networks. Transp. Sci. **24**(1), 77–78 (1990)
5. Boland, N., Dethridge, J., Dumitrescu, I.: Accelerated label setting algorithms for the elementary resource constrained shortest path problem. Oper. Res. Lett. **34**(1), 58–68 (2006)
6. Calik, H., Labbé, M., Yaman, H.: p-center problems. In: Location Science, pp. 79–92. Springer, Berlin (2015)
7. Callaghan, B., Salhi, S., Brimberg, J.: Optimal solutions for the continuous p-centre problem and related-neighbour and conditional problems: a relaxation-based algorithm. J. Oper. Res. Soc. **70**(2), 192–211 (2019)
8. Chen, D., Chen, R.: New relaxation-based algorithms for the optimal solution of the continuous and discrete p-center problems. Comput. Oper. Res. **36**(5), 1646–1655 (2009)

9. Chen, D., Chen, R.: A relaxation-based algorithm for solving the conditional p-center problem. Oper. Res. Lett. **38**(3), 215–217 (2010)
10. Chen, R., Handler, G.Y.: Relaxation method for the solution of the minimax location-allocation problem in euclidean space. Nav. Res. Logist. (NRL) **34**(6), 775–788 (1987)
11. Chen, R., Handler, Y.: The conditional p-center problem in the plane. Nav. Res. Logist. (NRL) **40**(1), 117–127 (1993)
12. Chvatal, V.: Linear Programming. Series of Books in the Mathematical Sciences. W. H. Freeman, New York (1983)
13. Contardo, C.: Decremental clustering for the solution of p-dispersion problems to proven optimality. techreport G-2019-22, Cahiers du GERAD (2019)
14. Contardo, C., Sefair, J.A.: A progressive approximation approach for the exact solution of sparse large-scale binary interdiction games. techreport G-2019-49, Cahiers du GERAD (2019)
15. Contardo, C., Iori, M., Kramer, R.: A scalable exact algorithm for the vertex p-center problem. Comput. Oper. Res. **103**, 211–220 (2019)
16. Daskin, M.S., Maass, K.L.: The p-median problem. In: Location Science, pp. 21–45. Springer, Berlin (2015)
17. Delattre, M., Hansen, P.: Bicriterion cluster analysis. IEEE Trans. Pattern Anal. Mach. Intell. **2**(4), 277–291 (1980)
18. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Springer, Berlin (2012)
19. Drezner, T., Drezner, Z.: Sequential location of two facilities: comparing random to optimal location of the first facility. Ann. Oper. Res. **246**(1–2), 5–18 (2016)
20. Drezner, Z.: Conditional p-center problems. Transp. Sci. **23**(1), 51–53 (1989)
21. Drezner, Z.: On the conditional p-median problem. Comput. Oper. Res. **22**(5), 525–530 (1995)
22. Drezner, Z., Erkut, E.: Solving the continuous p-dispersion problem using non-linear programming. J. Oper. Res. Soc. **46**(4), 516–520 (1995)
23. Drezner, Z., Kalczynski, P., Salhi, S.: The planar multiple obnoxious facilities location problem: a voronoi based heuristic. Omega **87**, 105–116 (2019)
24. Erkut, E.: The discrete p-dispersion problem. Eur. J. Oper. Res. **46**(1), 48–60 (1990)
25. Gauthier, J.B., Desrosiers, J., Lübbecke, M.E.: About the minimum mean cycle-canceling algorithm. Discret. Appl. Math. **196**, 115–134 (2015)
26. Hooker, J.N., Garfinkel, R.S., Chen, C.: Finite dominating sets for network location problems. Oper. Res. **39**(1), 100–118 (1991)
27. Irawan, C.A., Salhi, S., Scaparra, M.P.: An adaptive multiphase approach for large unconditional and conditional p-median problems. Eur. J. Oper. Res. **237**(2), 590–605 (2014)
28. Irnich, S., Desaulniers, G., Desrosiers, J., Hadjar, A.: Path-reduced costs for eliminating arcs in routing and scheduling. INFORMS J. Comput. **22**(2), 297–313 (2010)
29. Kuby, M.J.: Programming models for facility dispersion: The p-dispersion and maxisum dispersion problems. Geogr. Anal. **19**(4), 315–329 (1987)
30. Laporte, G., Nickel, S., da Gama, F.S.: Location Science. Springer, Berlin (2015)
31. Laporte, G., Nickel, S., da Gama, F.S.: Location science. Springer, Berlin (2019)
32. Martinelli, R., Pecin, D., Poggi, M.: Efficient elementary and restricted non-elementary route pricing. Eur. J. Oper. Res. **239**(1), 102–111 (2014)
33. Pisinger, D.: Upper bounds and exact algorithms for p-dispersion problems. Comput. Oper. Res. **33**(5), 1380–1398 (2006)
34. Reinelt, G.: TSPLIB-A traveling salesman problem library. INFORMS J. Comput. **3**(4), 376–384 (1991). https://doi.org/10.1287/ijoc.3.4.376
35. Righini, G., Salani, M.: New dynamic programming algorithms for the resource constrained elementary shortest path problem. Netw. Int. J. **51**(3), 155–170 (2008)
36. Saboonchi, B., Hansen, P., Perron, S.: Maxminmin p-dispersion problem: a variable neighborhood search approach. Comput. Oper. Res. **52**, 251–259 (2014)
37. Sayah, D., Irnich, S.: A new compact formulation for the discrete p-dispersion problem. Eur. J. Oper. Res. **256**(1), 62–67 (2017)
38. Statman, M.: How many stocks make a diversified portfolio? J. Financ. Quant. Anal. **22**(3), 353–363 (1987). https://doi.org/10.2307/2330969