Les Cahiers du GERAD ISSN: 0711–2440

The conditional p-dispersion problem

M. Cherkesly,
C. Contardo

G–2019–61

August 2019

La publication de ces rapports de recherche est rendue possible grâce au soutien de HEC Montréal, Polytechnique Montréal, Université McGill, Université du Québec à Montréal, ainsi que du Fonds de recherche du Québec – Nature et technologies.

Dépôt légal – Bibliothèque et Archives nationales du Québec, 2019
– Bibliothèque et Archives Canada, 2019

The publication of these research reports is made possible thanks to the support of HEC Montréal, Polytechnique Montréal, McGill University, Université du Québec à Montréal, as well as the Fonds de recherche du Québec – Nature et technologies.

Legal deposit – Bibliothèque et Archives nationales du Québec, 2019
– Library and Archives Canada, 2019

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

The conditional p-dispersion problem

Marilène Cherkesly [a, b]

Claudio Contardo [a, b, c]

[a] *GERAD, Montréal (Québec), Canada, H3T 2A7*

[b] *Department of management and technology, ESG UQAM, Montréal (Québec) Canada, H3C 3P8*

[c] *CIRRELT, Montréal (Québec), Canada, H3T 1J4*

cherkesly.marilene@uqam.ca
contardo.claudio@uqam.ca

August 2019
Les Cahiers du GERAD
G–2019–61

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., $z(P \cup Q) \coloneqq \min\{D(i,j), i,j \in P \cup Q\}$. 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 ad-hoc acceleration mechanisms designed to take advantage of the information provided by the set Q. We perform exhaustive computational experiments and show that the proposed acceleration mechanisms help reduce the total computational time by a large margin. We also assess the scalability of the algorithm and derive managerial insights regarding the value of building a network from scratch compared to an incremental design.

Acknowledgments: The authors thank the Canadian Natural Sciences and Engineering Research Council (NSERC) under Discovery Grant 2017–06106 for its financial support. We also thank the GERAD for providing access to its computing infrastructure.

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 $D(i, i) = 0$ 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 $z(P \cup Q) \coloneqq \min\{D(i, j), i, j \in P \cup Q\}$ 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 when p is an input parameter, as an instance of the pDP —$\mathcal{NP}$-hard as noticed by Erkut (1990)— 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($\mathtt{N}', \mathtt{Q}', \mathtt{D}', \mathtt{p}$), 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, for a fixed value of p, the c-pDP can be solved in polynomial 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 (Kuby 1987). Belotti et al. (2006) 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 (Saboonchi et al. 2014). In portfolio optimization, selecting a diversified portfolio is important (Statman 1987) 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 (Saboonchi et al. 2014). Finally, in the maximin split clustering problem, which can be solved in polynomial time (Delattre and Hansen 1980), 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 (Contardo 2019) 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 (2019) 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. Finally, we derive managerial insights regarding the value of optimally building a network from scratch as compared to an incremental setting.

The remainder of this article is organized as follows. In Section 2, we present a review of the relevant scientific literature. In Section 3, we propose an exact solution algorithm for the c-pDP including

new acceleration mechanisms tailored for this problem. In Section 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 managerial insights. Finally, Section 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 pDP and conditional location problems (the conditional p-median and the conditional p-center problems) have already been subjects of scientific analysis (Calik et al. 2015, Daskin and Maass 2015). Our proposed decremental clustering algorithm is similar to relaxation mechanisms that have been proposed to solve minimax and maximin optimization problems to optimality (Callaghan et al. 2019, Chen and Handler 1987, 1993, Chen and Chen 2009, 2010, Pisinger 2006, Aloise and Contardo 2018) as well as routing problems (Boland et al. 2006, Righini and Salani 2008, Martinelli et al. 2014). 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 (1987) 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 (2006) 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 (2017) 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 1,000 vertices. The state-of-the-art algorithm for the pDP was proposed by Contardo (2019) 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 (1989) 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 (1990) 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 (2008) 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 (1990) for the the conditional p-center and the conditional p-median problems. Drezner (1995) propose solution methods for the conditional 1-median problem on an instance with 100 vertices. Chen and Chen (2010) 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. (2014) 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 (2016) 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 that 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 (Contardo 2019). 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 (2009) and Contardo et al. (2019) 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. (2019) solves instances with to 1,000,000 vertices to optimality within reasonable computation time. Aloise and Contardo (2018) 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 (2019) introduced a progressive approximation scheme for the solution of sparse large-scale binary interdiction games that is closely related to our algorithm.

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 (Contardo 2019). 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
$\underline{z} \leftarrow$heuristicCPDP(N, Q, D, p)
$G', D' \leftarrow$ graphReduction(N, Q, D, $\underline{z}$)
$S \leftarrow$ 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 $i \in \arg\max\{d_{i^+} : i \in N \setminus X\}$. 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

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

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 equation (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 equation (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 equation (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 Section 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 (2019) 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 Section 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 (2019) 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., $C_i \cap C_j = \emptyset, \forall\ 1 \leq i < j \leq m$ 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 $\underline{z} \leq z^*$, 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 (2019). 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., $S = \{w \in W : |C_w| \geq 2\}$. 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 (Aloise and Contardo 2018, Contardo et al. 2019, Contardo 2019). 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 (2017) within the double binary search algorithm proposed by Contardo (2019) 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 (2017). 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 (2017) 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 and managerial insights

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 Section 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 seconds, 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 containing between 1,621 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. In addition, we detail in the remainder how we have generated the set Q. 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: a) the choice of the constructive strategy used to build set Q; and b) the different values of p and q. In the following subsection, we compare the performance of our algorithm against a naive adaptation of the state-of-the-art algorithm for the pDP to the conditional case. We show that our proposed algorithm reduces by a large margin the total computational time compared to a more naive approach. For conciseness reasons, we only report summarized results in the manuscript. Detailed results can be found in Appendices A–B.

4.1 Managerial insights

In this section, we present sensitivity analyses regarding the quality of the solutions and the computing times associated with a) the construction strategy for the set Q; and b) the values of p and q. Detailed results are reported in Appendix A.

4.1.1 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 (2019). 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 1 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* Δ 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, Figure 1, 2, and 3 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.

Table 1: Impact of the construction strategy for the set Q

	Greedy	**Optimal**	**Random**
Average Δ z^* (%)	**–1.4**	–1.9	–74.4
Worst (%)	**–12.8**	–24.0	–99.5
# Best	**368**	280	0
Average CPU time (s)	**1,095.2**	1,929.5	9,067.1
# Solved	**638**	543	543
Average elim. (%)	**74.2**	68.5	21.4

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.

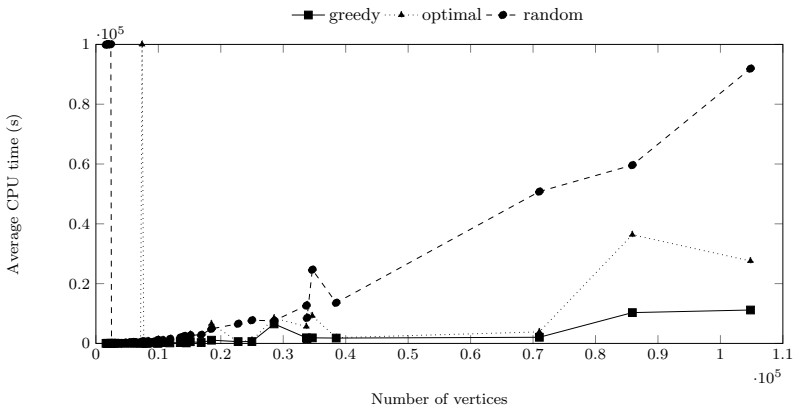

Figure 1: Impact of the construction algorithm on the average time

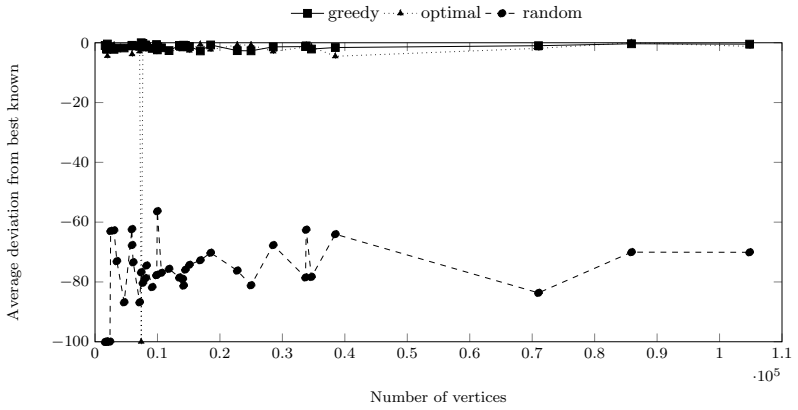

Figure 2: Impact of the construction algorithm on the average deviation from the best known solution

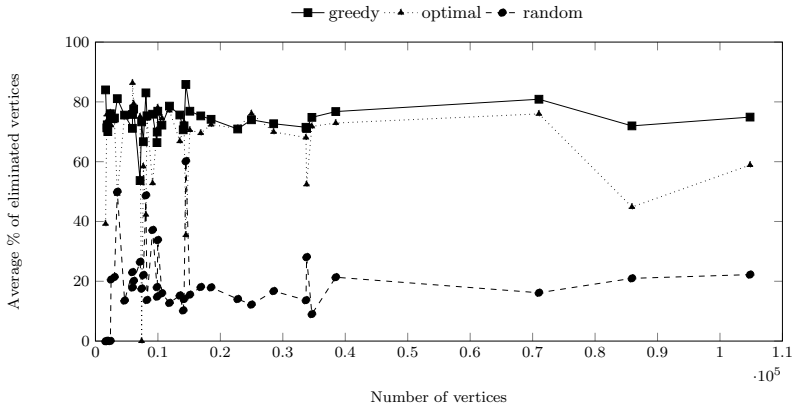

Figure 3: Impact of the construction algorithm on the percentage of eliminated vertices

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. For the remainder of our computational analysis, we will derive managerial insights for the instances where the set Q is greedily constructed. Note that the derived managerial insights also hold when the set Q is optimally and randomly constructed.

4.1.2 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. Tables 2–4 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.

Table 2: 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 3: 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 4: 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

4.1.3 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, Figures 4, 5, 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 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.

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.

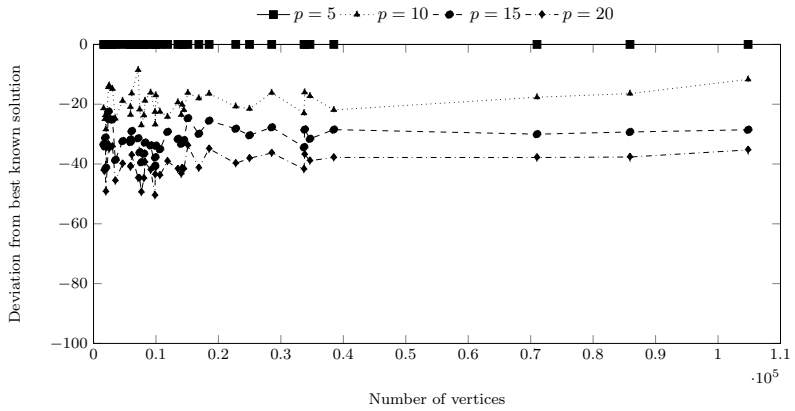

Figure 4: Impact of increasing the value of p on the deviation from the best known solution ($q = 5$)

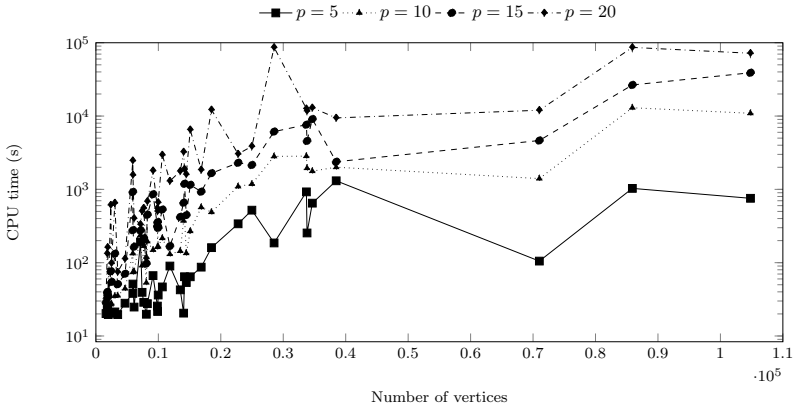

Figure 5: Impact of increasing the value of p on the computational time ($q = 5$)

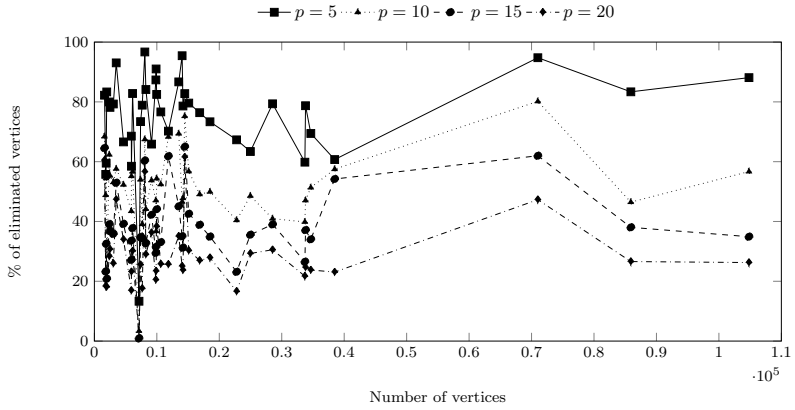

Figure 6: Impact of increasing the value of p on the percentage of eliminated vertices ($q = 5$)

4.1.4 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, Figures 7, 8, 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 q and fixing $p = 5$. Note that the best 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.

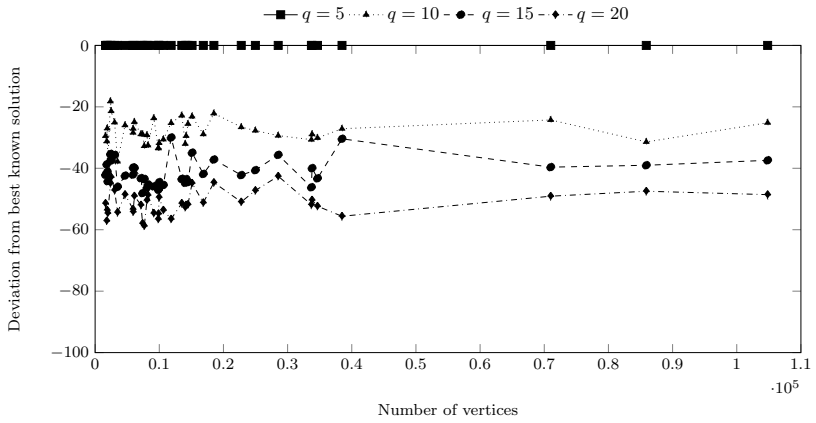

Figure 7: Impact of increasing the value of q on the deviation from the best known solution ($p = 5$)

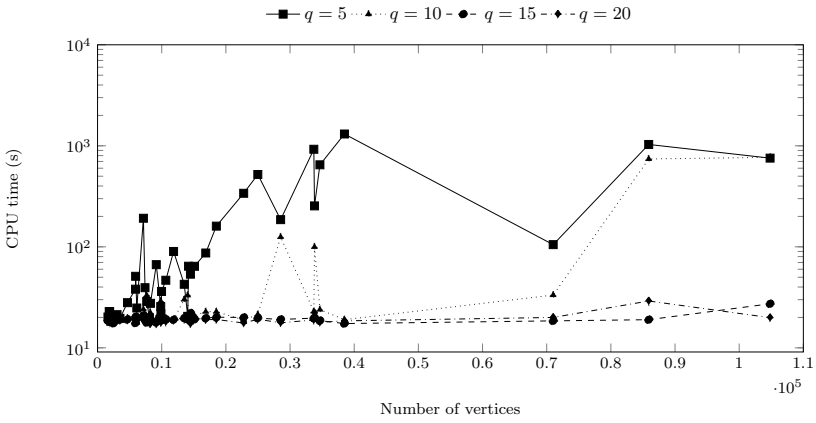

Figure 8: Impact of increasing the value of q on the total computational time ($p = 5$)

4.1.5 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 10, 11, 12 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

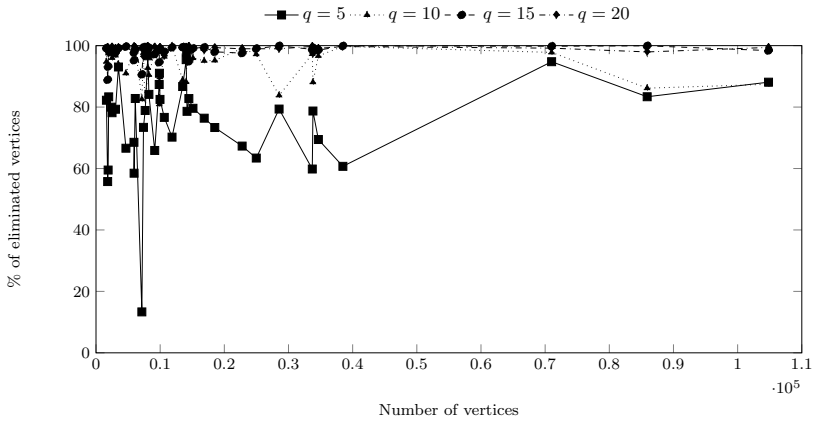

Figure 9: Impact of increasing the value of q on the percentage of eliminated vertices ($p = 5$)

increases as the size of q decreases and the size p increases. In fact, when q decreases and p increases, the percentage of eliminated 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.

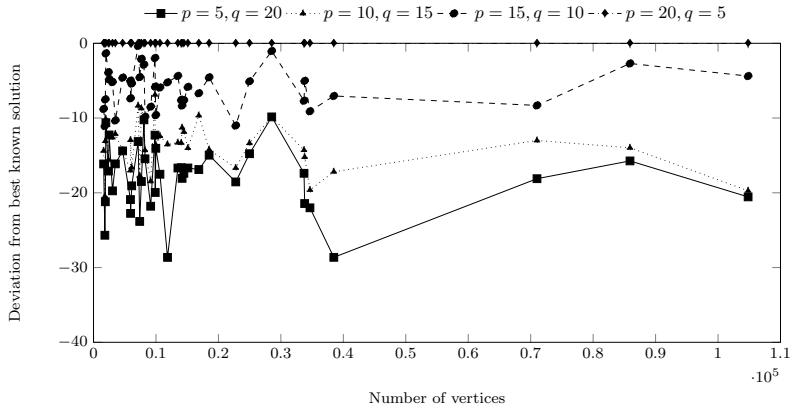

Figure 10: Impact of modifying the values of p and q on the deviation from the best known solution ($q + p = 25$)

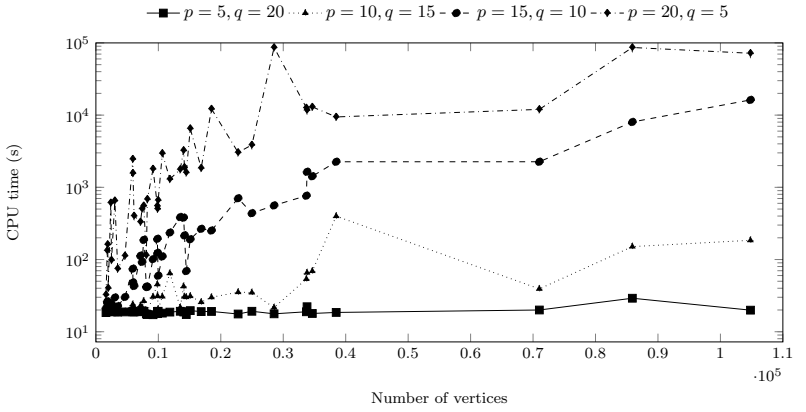

Figure 11: Impact of modifying the values of p and q on the computational times ($q + p = 25$)

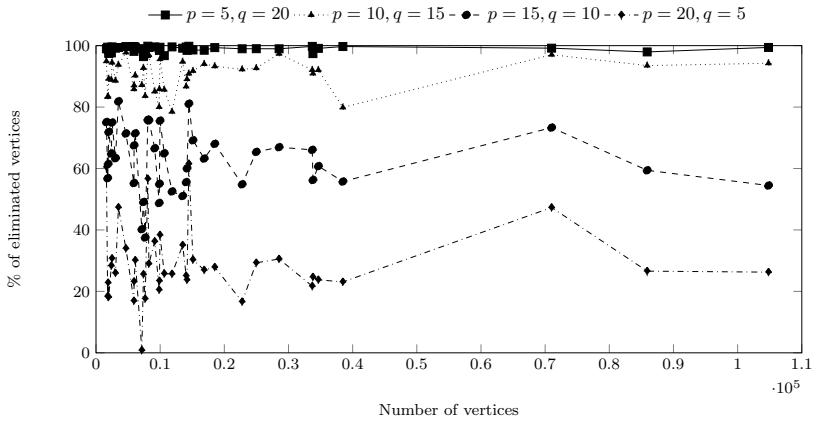

Figure 12: Impact of modifying the values of p and q on the percentage of eliminated vertices ($q + p = 25$)

4.2 Performance of the 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 (2019) 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. 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 B.

Tables 5–6 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 $\text{Sec}_{\text{naive}}/\text{Sec}_{ad\ hoc}$, where $\text{Sec}_{\text{naive}}$ and $\text{Sec}_{ad\ hoc}$ 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 Figures 13–16 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. 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 Figures 13–16. In addition, the differences become more apparent as the sizes of the instances grow. Our algorithm is most of the time orders of magnitude faster than the naive setting.

Table 5: 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 6: 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

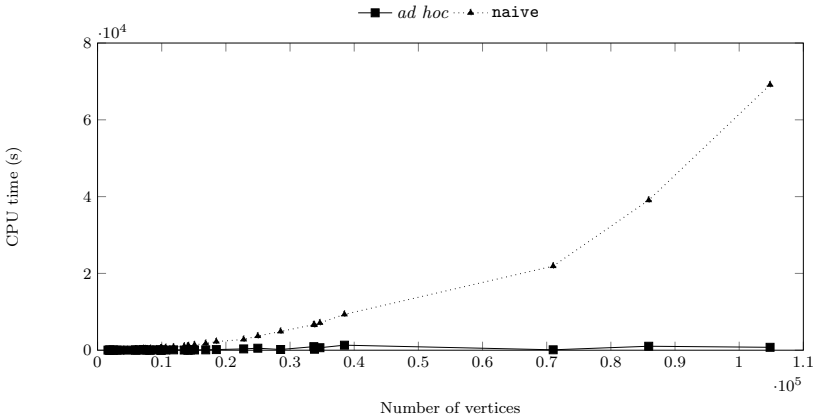

Figure 13: Time profiles of the naive and the ad hoc algorithms ($p = 5, q = 5$)

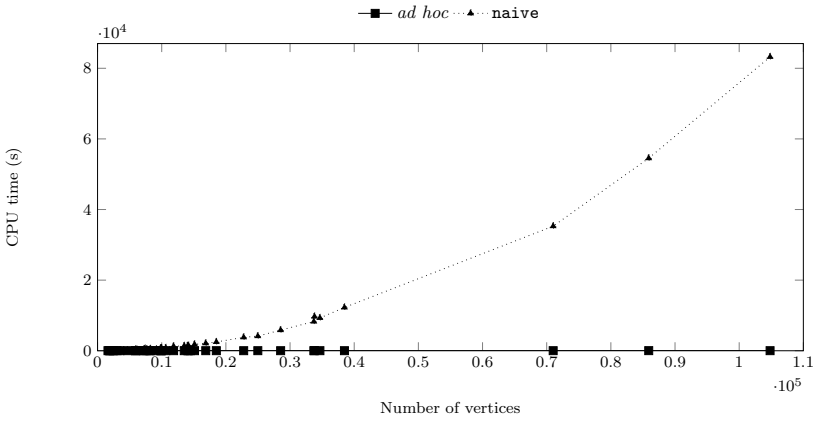

Figure 14: Time profiles of the naive and ad hoc algorithms ($p = 5, q = 20$)

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 (2019). 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. Our numerical experiments

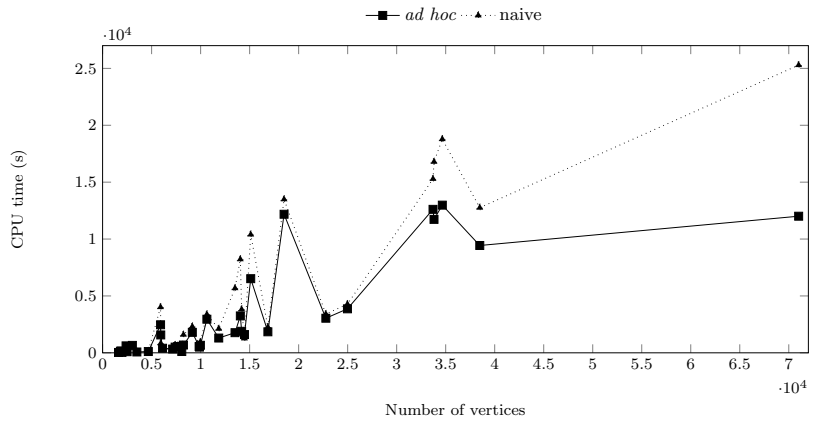

Figure 15: Time profiles of the naive and ad hoc algorithms ($p = 20, q = 5$)

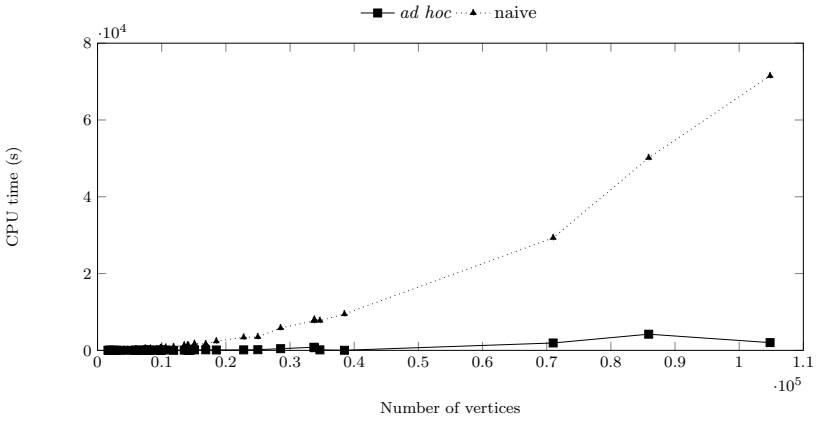

Figure 16: Time profiles of the naive and the ad hoc algorithms ($p = 20, q = 20$)

first show that greedily constructing the set Q, instead of optimally or randomly constructing it, provides the best results. We also 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 reduced the computational time compared to a naive adaptation of Contardo (2019)'s algorithm for the `pDP` to the conditional case.

A Detailed results

In this section, we presented detailed results. Tables 7–22 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 7: Detailed results with $p = 5$ and $q = 5$

	Greedy			Optimal			Random		
Instance	**Sec**	z^*	**Elim**	**Sec**	z^*	**Elim**	**Sec**	z^*	**Elim**
ar9152	66.6	7,329	6,029	3.4	7,142	5,759	537.1	985	2,979
bby34656	648.5	342	24,064	288.4	330	26,815	25,691.6	12	117
bm33708	924.6	3,482	20,169	623.3	3,649	21,340	4,793.3	1,275	11,133
brd14051	20.5	2,287	13,410	151.7	2,268	8,926	2,738.3	241	784
ca4663	28.0	17,154	3,106	12.6	19,090	2,972	149.4	3,022	1,202
ch71009	105.1	12,483	67,295	20.0	12,549	69,084	11,501.4	4,131	34,320
d15112	64.2	6,239	12,030	58.4	6,676	11,563	2,081.7	1,689	2,109
d15112-2500	20.5	6,077	1,954	2.4	6,556	1,858			
d18512	160.3	2,258	13,577	61.4	2,347	15,031	6,196.3	177	258
eg7146	191.9	2,599	952	0.6	2,821	6,696	505.3	218	1,409
ei8246	27.6	1,291	6,938	44.5	1,265	5,726	94.1	753	4,492
fi10639	46.7	3,451	8,154	0.3	3,649	10,459	785.3	860	2,501
fyg28534	185.8	303	22,644	116.6	296	23,295			
gr9882	21.6	2,448	8,994	8.4	2,382	8,245	913.6	490	1,171
ho14473	53.6	1,300	11,979	20.1	1,254	5,222	16.1	768	13,410
it16862	86.9	2,803	12,880	95.4	2,873	12,502	2,058.8	703	3,392
ja9847	25.6	5,045	8,601	29.4	4,524	7,508	963.7	738	1,142
kz9976	36.2	7,516	8,230	1.4	7,910	9,376	177.1	3,635	4,952
mo14185	64.3	2,538	11,152	4.0	2,639	13,362	54.0	1,590	11,300
mu1979	19.6	2,057	1,649	0.3	2,066	1,716			
nu3496	19.6	1,424	3,253	0.1	1,366	2,316	102.3	85	1,226
pba38478	1,309.1	365	23,360	0.2	399	38,392	14,720.5	79	4,337
pcb3038	21.3	1,169	2,409	1.2	1,297	2,568	52.1	518	910
pla33810	254.2	232,371	26,614	403.4	247,329	23,938	1,177.2	132,847	20,304
pla7397	39.4	223,567	5,429				37.2	132,687	5,050
pla85900	1,031.3	312,216	71,608	819.2	313,348	73,068	14,737.2	135,329	42,531
pm8079	19.8	1,202	7,812	0.7	1,207	4,473	332.2	118	3,467
pr2392	20.6	3,991	1,913	0.6	4,577	2,114			
rl11849	90.0	5,559	8,320	0.2	6,122	11,725	640.9	2,023	3,774
rl1889	22.8	5,208	1,124	0.1	5,768	1,859			
rl5915	38.1	5,527	4,054	4.7	5,137	4,943	20.4	3,963	4,593
rl5934	51.1	5,378	3,470	0.2	5,915	5,790	450.3	800	345
rw1621	20.2	492	1,333	0.7	474	630			
sra104815	755.4	564	92,352	4,128.1	602	80,723	96,546.1	134	15,017
sw24978	520.3	3,553	15,837	8.6	3,931	23,729	9,656.0	337	794
tz6117	24.8	3,239	5,065	5.8	3,300	5,184	263.1	896	1,391
u1817	22.9	830	1,013	0.1	867	1,790			
usa13509	42.6	117,178	11,711	148.1	121,128	8,126	316.6	54,638	7,090
vm22775	339.0	2,489	15,329	465.4	2,567	13,714	3,023.8	854	5,367
ym7663	28.8	2,565	6,046				32.8	1,143	5,884

Table 8: Detailed results with $p = 5$ and $q = 10$

	Greedy			Optimal			Random		
Instance	**Sec**	z^*	**Elim**	**Sec**	z^*	**Elim**	**Sec**	z^*	**Elim**
ar9152	19.4	5,594	8,987	2.5	5,048	6,035	619.7	574	3,134
bby34656	23.9	239	33,495	0.1	225	34,643	13,743.5	51	3,349
bm33708	22.9	2,414	32,791	3.3	2,354	32,759	14,490.3	399	3,338
brd14051	33.1	1,554	12,387	0.9	1,593	13,644	2,463.9	243	1,276
ca4663	19.7	12,698	4,243	0.8	12,207	4,209	402.3	555	189
ch71009	33.3	9,454	69,446	513.2	8,828	59,987	99,173.0	421	1,641
d15112	20.8	4,797	14,501	0.4	4,392	14,861	5,126.5	251	239
d15112-2500	18.8	4,778	2,400	0.1	4,424	2,424			
d18512	22.8	1,758	17,620	0.4	1,639	18,292	8,477.5	73	125
eg7146	24.1	1,848	5,901	0.2	1,883	6,994	312.9	238	2,460
ei8246	22.2	871	7,460	0.2	890	8,149	1,187.0	92	284
fi10639	19.6	2,395	10,257	0.7	2,227	10,301	1,564.8	331	801
fyg28534	125.2	214	23,915	33.8	222	25,634	5,524.3	72	6,525
gr9882	34.2	1,628	7,998	0.1	1,784	9,820	1,080.9	307	957
ho14473	18.8	968	14,452	0.8	854	6,731	412.1	224	8,963
it16862	22.8	1,993	16,029	4.2	2,166	15,743	2,579.0	437	3,141
ja9847	29.0	3,360	8,651	0.1	3,275	9,822	1,266.9	364	924
kz9976	24.9	5,128	8,951	1.3	5,217	9,548	172.5	2,485	5,749
mo14185	19.2	1,790	13,764	10.5	1,766	12,684	2,389.8	278	1,692
mu1979	19.2	1,280	1,929	0.1	1,267	1,939			
nu3496	19.7	886	3,284	0.8	880	2,029	79.7	118	1,402
pba38478	19.0	266	38,381	14.8	238	36,605	12,232.1	76	7,589
pcb3038	19.0	876	2,941	0.2	894	2,892	77.7	289	678
pla33810	99.8	165,217	29,788	40.6	156,765	30,726	3,834.2	69,657	15,291
pla7397	19.4	158,899	7,223				709.4	20,881	444
pla85900	741.1	214,133	74,012				79,080.2	46,681	12,171
pm8079	21.0	851	7,493	0.4	786	4,670	554.3	33	3,176
pr2392	19.1	3,265	2,372	0.4	2,909	2,197			
rl11849	19.1	4,155	11,837	1.0	3,639	11,428	1,544.6	861	1,555
rl1889	17.3	3,802	1,867	0.3	3,532	1,733			
rl5915	19.0	3,957	5,825	0.1	3,575	5,840	241.4	1,371	1,528
rl5934	18.9	3,920	5,910	0.4	3,538	5,716	29.7	2,788	4,238
rw1621	18.7	347	1,536	0.1	324	803			
sra104815	769.6	422	91,625	225.7	414	98,566	101,152.4	98	17,681
sw24978	21.4	2,568	24,284	1.9	2,655	24,384	6,600.8	485	3,447
tz6117	17.7	2,432	6,076	0.3	2,327	5,953	461.3	353	617
u1817	18.9	571	1,797	0.2	540	1,711			
usa13509	30.1	90,431	11,999	1.8	82,279	12,908	2,577.6	10,925	853
vm22775	18.9	1,827	22,572	0.5	1,684	22,472	6,267.3	297	2,516
ym7663	33.2	1,725	6,011				242.7	480	3,266

Table 9: Detailed results with $p = 5$ and $q = 15$

	Greedy			Optimal			Random		
Instance	**Sec**	z^*	**Elim**	**Sec**	z^*	**Elim**	**Sec**	z^*	**Elim**
ar9152	17.9	3,967	8,896	0.1	3,996	6,709	414.5	1,019	3,866
bby34656	18.6	194	34,141	2.0	183	33,958	17,560.6	32	2,164
bm33708	19.7	1,874	33,300				21,227.2	158	887
brd14051	19.3	1,264	13,668	6.6	1,217	13,011	2,708.4	193	1,237
ca4663	19.1	9,879	4,648	0.1	10,107	4,616	400.8	472	221
ch71009	18.5	7,536	70,876	1.1	7,074	70,596	68,138.4	889	7,115
d15112	19.1	4,060	14,978	0.2	3,751	15,009	3,156.6	613	1,462
d15112-2500	17.5	3,797	2,462	0.1	3,780	2,464			
d18512	20.1	1,419	18,139	0.5	1,386	18,266	20,800.6	11	15
eg7146	21.0	1,476	6,477	1.1	1,438	6,552	246.5	243	2,792
ei8246	18.7	706	8,184	0.4	693	7,999	598.1	235	1,617
fi10639	19.1	1,881	10,422	0.3	1,903	10,461	497.3	648	4,510
fyg28534	19.1	195	28,517	0.7	167	28,146	5,347.4	66	7,515
gr9882	19.2	1,298	9,757	0.2	1,213	9,770	1,982.5	83	259
ho14473	22.1	721	13,726	0.4	734	6,866	758.1	94	7,909
it16862	19.7	1,630	16,772	3.6	1,621	15,873	5,895.3	101	306
ja9847	21.1	2,696	9,312	2.0	2,431	9,160	597.4	635	3,031
kz9976	18.7	4,171	9,869	0.2	4,206	9,864	1,322.0	647	1,087
mo14185	19.2	1,433	13,881	0.9	1,375	13,727	2,808.2	186	1,362
mu1979	18.2	1,148	1,952	0.1	1,047	1,959			
nu3496	19.0	768	3,464	0.1	729	2,336			
pba38478	17.4	254	38,445	0.1	193	38,447	14,106.9	61	6,134
pcb3038	19.2	752	2,969	0.1	694	2,976	160.4	92	126
pla33810	19.8	139,533	33,553				19,324.5	16,049	1,276
pla7397	19.0	116,087	7,361				684.6	15,713	894
pla85900	19.0	190,403	85,854				81,063.8	36,501	12,439
pm8079	19.3	645	7,979	0.2	655	4,849	215.9	137	4,214
pr2392	19.2	2,581	2,345	0.1	2,486	2,354			
rl11849	19.0	3,892	11,779	0.2	3,301	11,741	2,993.7	227	191
rl1889	19.9	3,066	1,760	0.1	3,185	1,868			
rl5915	17.6	3,202	5,782	0.1	3,044	5,812	533.2	448	328
rl5934	20.4	3,242	5,652	0.1	3,067	5,907	415.5	692	732
rw1621	18.7	285	1,609	0.1	275	855			
sra104815	27.3	353	103,136				16,240.0	171	63,251
sw24978	19.7	2,110	24,739	2.0	2,040	24,214	3,107.6	719	8,841
tz6117	19.5	1,944	6,036	0.3	1,917	5,917	539.1	248	712
u1817	19.2	509	1,615	0.1	472	1,808			
usa13509	19.8	66,284	13,450	0.2	62,636	13,414	1,931.8	12,417	2,377
vm22775	20.0	1,438	22,220	0.1	1,425	22,723	6,085.5	284	3,179
ym7663	17.8	1,449	7,627				1,215.2	58	249

Table 10: Detailed results with $p = 5$ and $q = 20$

	Greedy			Optimal			Random		
Instance	**Sec**	z^*	**Elim**	**Sec**	z^*	**Elim**	**Sec**	z^*	**Elim**
ar9152	17.2	3,326	9,114	0.2	3,244	6,616	287.4	1,022	4,693
bby34656	17.9	163	34,350				12,031.0	47	6,074
bm33708	18.9	1,677	33,627	1.4	1,535	33,023	16,240.5	264	3,137
brd14051	18.8	1,082	13,989	0.5	1,030	13,826	4,370.8	79	356
ca4663	18.8	8,826	4,649	0.1	8,133	4,621	483.5	271	163
ch71009	20.0	6,356	70,428	0.9	6,447	70,522	34,246.4	1,363	22,313
d15112	19.6	3,442	14,898	0.4	3,064	14,843	4,849.1	265	369
d15112-2500	18.9	3,471	2,490	0.2	3,064	2,423			
d18512	19.1	1,250	18,393	1.5	1,094	17,924	7,811.0	87	336
eg7146	18.9	1,249	7,075	0.1	1,148	7,053	670.5	97	1,175
ei8246	17.3	662	8,197	0.1	614	8,186	875.1	134	961
fi10639	18.1	1,601	10,292	0.2	1,525	10,503	1,056.6	429	2,079
fyg28534	17.7	174	28,241				12,749.5	25	1,955
gr9882	17.8	1,107	9,728	0.4	1,068	9,645	2,075.5	69	222
ho14473	17.3	627	14,445	0.1	627	7,062	2,050.2	17	7,390
it16862	19.0	1,368	16,610	0.3	1,434	16,674	2,907.1	297	3,078
ja9847	19.1	2,192	9,759	0.2	2,108	9,731	1,541.7	194	801
kz9976	19.2	3,801	9,924	0.1	3,560	9,883	1,231.8	638	1,266
mo14185	18.9	1,220	13,967	0.1	1,163	14,105	3,883.2	98	447
mu1979	19.1	934	1,929	0.1	837	1,938			
nu3496	18.7	650	3,475	0.1	591	2,340			
pba38478	18.5	162	38,364				21,603.0	33	2,740
pcb3038	18.9	618	3,013	0.1	619	3,019	94.2	193	595
pla33810	22.2	115,413	32,928				5,191.4	46,174	13,526
pla7397	19.4	94,022	7,133						
pla85900	29.1	164,010	84,142				125,028.7	17,749	3,188
pm8079	17.6	596	8,065	0.2	537	4,808	518.7	37	3,278
pr2392	19.4	2,202	2,335	0.1	2,057	2,354			
rl11849	18.7	2,417	11,800	0.3	2,501	11,638	1,670.1	657	1,536
rl1889	19.2	2,416	1,865	0.1	2,340	1,837			
rl5915	19.1	2,580	5,805	0.1	2,554	5,874	310.3	878	1,238
rl5934	18.8	2,465	5,918	0.1	2,366	5,908	630.5	310	234
rw1621	18.4	239	1,605	0.1	240	860			
sra104815	19.9	290	104,204				126,650.3	60	14,945
sw24978	19.2	1,875	24,728	1.4	1,752	24,437	8,426.5	285	2,782
tz6117	18.8	1,651	6,085	0.1	1,627	6,076	580.4	217	568
u1817	19.2	356	1,808	0.1	345	1,766			
usa13509	19.2	56,954	13,396						
vm22775	17.6	1,221	22,542	1.4	1,249	22,225	10,244.1	105	719
ym7663	19.2	1,058	7,520				760.2	148	1,052

Table 11: Detailed results with $p = 10$ and $q = 5$

	Greedy			Optimal			Random		
Instance	**Sec**	z^*	**Elim**	**Sec**	z^*	**Elim**	**Sec**	z^*	**Elim**
ar9152	148.9	6,148	4,919	67.2	5,764	3,378	76.2	2,965	5,855
bby34656	1,772.5	283	17,792	5,175.8	279	11,417	9,868.9	89	3,416
bm33708	2,839.3	2,681	13,433	4,361.2	2,672	11,722	15,189.3	308	1,501
brd14051	373.1	1,748	6,546	712.1	1,745	4,900	1,652.2	420	1,755
ca4663	44.3	13,916	2,438	44.5	14,950	1,715	144.9	4,779	374
ch71009	1,403.3	10,278	56,932	5,411.9	10,107	35,984	35,563.4	1,929	14,305
d15112	268.9	5,230	8,578	722.1	5,592	5,452	293.8	3,379	7,736
d15112-2500	27.6	5,243	1,383	19.6	5,468	1,008			
d18512	488.8	1,886	9,242	976.2	1,976	7,465	405.8	1,305	9,457
eg7146	237.3	2,377	241	15.2	2,070	5,318	556.8	151	734
ei8246	195.2	1,048	3,642	177.9	1,030	3,531	376.6	398	1,685
fi10639	215.9	2,674	5,579	288.7	2,644	3,837	1,352.0	369	579
fyg28534	2,818.1	254	11,683	2,510.1	257	11,284	1,895.5	170	9,123
gr9882	293.4	1,793	3,630	155.5	1,851	5,006	1,646.8	100	178
ho14473	134.7	1,015	10,887	137.7	1,026	3,186	526.0	180	7,986
it16862	570.3	2,298	8,279	645.6	2,203	7,037	322.4	2,253	9,964
ja9847	163.0	3,907	4,633	76.4	3,604	5,510	576.2	999	2,018
kz9976	165.9	6,239	5,425	190.9	5,933	4,849	92.0	5,304	6,803
mo14185	431.5	2,027	6,779	272.9	2,005	7,705	4,805.2	52	24
mu1979	34.7	1,473	466	13.7	1,419	476			
nu3496	35.5	1,071	2,014	13.6	998	1,027	17.7	529	2,525
pba38478	2,005.4	285	22,124	4,990.7	305	14,757	532.8	235	25,722
pcb3038	35.1	995	1,614	40.7	1,045	972	28.5	921	2,134
pla33810	1,947.4	195,151	15,908	4,292.8	200,849	8,614	492.4	181,080	25,089
pla7397	91.7	174,952	3,994				318.2	42,427	1,688
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	5,452	42.2	917	2,318	346.7	85	3,281
pr2392	26.9	3,422	1,491	21.1	3,662	918			
rl11849	130.8	4,213	8,094	358.8	4,702	4,583	748.9	1,680	2,796
rl1889	27.7	3,974	1,028	12.0	4,385	705			
rl5915	133.8	4,372	2,571	79.3	4,378	2,493	382.5	907	430
rl5934	74.1	4,112	3,273	103.8	4,428	2,292	231.6	1,516	944
rw1621	23.6	387	1,109	2.8	393	359			
sra104815	10,912.5	498	59,428	23,832.2	488	45,701	124,421.9	73	5,901
sw24978	1,182.5	2,788	12,128	1,673.7	2,932	9,040	1,767.6	1,473	8,391
tz6117	75.5	2,709	3,467	47.6	2,791	3,612	254.7	802	1,114
u1817	27.3	623	887	12.4	656	689			
usa13509	144.0	94,515	9,372	483.9	92,853	5,138	1,260.5	29,797	1,829
vm22775	1,095.3	1,973	9,207	1,178.0	1,973	9,261	5,275.3	384	1,709
ym7663	168.7	1,872	2,996				126.6	703	3,647

Table 12: Detailed results with $p = 10$ and $q = 10$

	Greedy			Optimal			Random		
Instance	**Sec**	z^*	**Elim**	**Sec**	z^*	**Elim**	**Sec**	z^*	**Elim**
ar9152	33.4	4,516	7,511	16.2	4,396	5,218	1,224.4	83	2,449
bby34656	337.8	213	27,443	1.2	216	34,214	15,175.6	41	2,303
bm33708	268.7	2,051	26,583	147.8	2,087	28,279	8,634.0	554	6,231
brd14051	113.9	1,384	10,363	31.6	1,363	11,767	1,765.9	318	2,152
ca4663	23.7	11,488	3,620	4.8	10,563	3,675	74.2	2,900	2,049
ch71009	581.5	8,009	61,090	1,776.9	7,901	52,639	82,183.2	492	2,268
d15112	46.5	4,164	13,155	5.9	4,275	13,968	2,000.6	1,032	2,468
d15112-2500	20.5	4,033	2,111	1.1	3,932	2,164			
d18512	97.1	1,498	14,824	5.6	1,501	17,526	2,014.3	569	4,517
eg7146	72.3	1,615	3,373	23.2	1,618	4,693	279.9	244	2,143
ei8246	34.1	736	6,660	6.2	833	7,227	349.6	355	2,079
fi10639	34.5	2,066	8,718	12.6	2,027	9,070	1,670.5	227	506
fyg28534	177.4	199	23,233	165.2	191	23,187	11,618.9	26	729
gr9882	63.1	1,409	7,262	2.9	1,445	9,139	1,626.6	97	245
ho14473	37.7	777	12,775	7.3	762	5,903	416.1	209	8,559
it16862	57.6	1,680	14,043	254.1	1,850	10,345	1,086.5	684	6,071
ja9847	72.4	2,760	6,647	24.2	2,745	7,590	1,276.2	232	618
kz9976	36.2	4,428	8,224	5.2	4,886	9,041			
mo14185	71.6	1,551	10,901	95.8	1,489	10,312	1,776.1	337	2,191
mu1979	19.8	1,188	1,828	0.4	1,117	1,805			
nu3496	20.5	784	3,150	2.5	814	1,747	23.5	291	2,331
pba38478	20.1	254	37,994	34.2	232	35,592	17,285.0	46	2,664
pcb3038	21.4	793	2,449	2.1	783	2,546	130.7	148	204
pla33810	340.1	146,697	26,119	39.7	152,675	30,488	2,628.4	74,216	15,840
pla7397	48.9	128,577	5,050				407.1	32,063	1,420
pla85900	2,112.6	197,511	66,261				47,012.1	59,572	19,705
pm8079	28.5	720	6,685	10.8	694	3,499	198.0	142	4,146
pr2392	20.1	2,695	1,975	1.8	2,713	1,923			
rl11849	19.5	3,892	11,581	2.0	3,543	11,204	1,223.8	1,019	1,289
rl1889	20.6	3,295	1,429	0.6	3,293	1,615			
rl5915	28.0	3,465	4,852	0.2	3,475	5,773	110.8	1,819	2,453
rl5934	26.1	3,412	4,854				384.7	700	560
rw1621	20.6	300	1,425	0.3	296	703			
sra104815	7,696.7	380	69,898	1,123.6	375	89,193	191,920.6	26	1,560
sw24978	155.0	2,256	19,681	104.6	2,249	20,215	6,632.3	395	2,648
tz6117	23.8	2,169	5,235	3.5	2,004	5,397	232.1	680	1,441
u1817	21.4	509	1,224	0.4	500	1,602			
usa13509	178.8	71,317	8,553	19.1	72,235	11,713	1,956.1	12,623	1,622
vm22775	204.8	1,526	16,762	59.3	1,560	19,409	3,583.9	448	4,661
ym7663	56.6	1,528	5,316				450.5	288	1,770

Table 13: Detailed results with $p = 10$ and $q = 15$

	Greedy			Optimal			Random		
Instance	**Sec**	z^*	**Elim**	**Sec**	z^*	**Elim**	**Sec**	z^*	**Elim**
ar9152	30.5	3,468	7,787	0.4	3,638	6,511	649.3	494	2,768
bby34656	69.1	168	31,905	5.2	177	33,550	15,993.8	31	1,921
bm33708	53.7	1,740	31,045	54.7	1,715	30,142	24,292.7	83	283
brd14051	42.5	1,125	12,185	19.6	1,081	12,183	3,287.3	113	524
ca4663	19.0	8,835	4,560	0.4	9,091	4,513	384.6	389	206
ch71009	39.2	6,753	68,920	44.5	6,273	67,828	38,730.1	1,363	14,875
d15112	31.1	3,552	13,869	2.6	3,474	14,276	374.5	1,978	8,372
d15112-2500	19.6	3,471	2,362	0.4	3,551	2,325			
d18512	30.0	1,261	17,272	6.0	1,244	17,422	3,746.4	289	2,117
eg7146	23.0	1,318	6,230	6.8	1,335	5,699	286.1	177	2,445
ei8246	19.2	671	7,993	0.9	638	7,834	450.0	230	1,969
fi10639	30.7	1,700	9,114	1.7	1,687	10,107	1,302.5	300	1,015
fyg28534	21.6	174	27,780	5.8	161	27,558	9,838.4	33	2,424
gr9882	31.2	1,188	8,485	2.3	1,156	9,198	1,027.4	194	1,343
ho14473	30.1	669	13,156	2.3	667	6,412	975.3	50	7,519
it16862	25.7	1,487	15,856	4.7	1,485	15,846	6,468.9	69	173
ja9847	45.4	2,327	7,881	16.6	2,226	8,197	3,100.6	37	34
kz9976	20.4	3,821	9,538	2.0	3,754	9,483	1,322.5	485	771
mo14185	30.6	1,321	12,639	28.0	1,217	11,746	3,921.3	75	183
mu1979	19.4	939	1,766	1.4	865	1,466			
nu3496	19.2	681	3,278	0.7	650	2,045	52.8	149	1,687
pba38478	398.8	188	30,736	75.1	173	34,484	18,457.8	36	2,779
pcb3038	19.7	673	2,691	0.2	654	2,918	153.3	84	113
pla33810	65.7	124,552	30,745				6,829.8	42,802	9,217
pla7397	21.6	101,560	6,861				934.1	7,212	200
pla85900	151.0	167,431	80,319				41,341.4	56,929	24,923
pm8079	19.5	596	7,806	0.7	580	4,605	249.8	105	3,888
pr2392	20.8	2,242	2,126	0.1	2,221	2,282			
rl11849	64.5	2,929	9,301	25.6	2,685	9,812	1,911.4	407	561
rl1889	20.9	2,665	1,577	1.1	2,660	1,520			
rl5915	23.9	2,707	5,078	2.3	2,728	5,321	487.2	440	266
rl5934	22.7	2,778	5,167				252.2	989	1,450
rw1621	19.3	244	1,539	0.2	245	794			
sra104815	184.6	293	98,810				29,404.7	138	47,204
sw24978	34.9	1,906	23,152	9.5	1,874	23,273	10,006.9	186	874
tz6117	21.2	1,701	5,525	0.7	1,671	5,783	710.1	85	251
u1817	20.7	381	1,514	0.6	395	1,536			
usa13509	21.7	59,273	12,803	1.6	57,000	12,990	1,864.5	12,758	1,824
vm22775	35.3	1,249	21,018	1.1	1,319	22,335	4,633.8	300	3,791
ym7663	26.6	1,185	6,408				618.4	170	1,100

Table 14: Detailed results with $p = 10$ and $q = 20$

	Greedy			Optimal			Random		
Instance	**Sec**	z^*	**Elim**	**Sec**	z^*	**Elim**	**Sec**	z^*	**Elim**
ar9152	19.3	3,110	9,026	0.9	3,034	6,323	641.4	418	2,983
bby34656	32.9	150	33,106				70,925.1	2	34
bm33708	25.4	1,623	32,709				14,987.7	261	2,748
brd14051	22.2	969	13,313	1.7	962	13,448	2,135.5	210	1,821
ca4663	20.9	7,938	4,325	0.2	7,397	4,509	361.2	414	279
ch71009	301.7	5,732	63,007	38.9	5,639	68,186			
d15112	20.4	3,254	14,652				4,390.9	281	408
d15112-2500	19.4	3,045	2,371	0.2	2,889	2,380			
d18512	24.2	1,155	17,568						
eg7146	20.6	1,120	6,633	0.2	1,081	6,929	253.9	169	2,722
ei8246	20.8	600	7,720	1.0	566	7,838	1,248.6	59	205
fi10639	22.0	1,441	9,796	0.9	1,406	10,225	866.7	403	2,356
fyg28534	79.1	140	25,000				9,509.7	31	2,752
gr9882	19.8	1,050	9,464	3.5	970	9,023	4,160.6	17	20
ho14473	19.2	567	14,261	0.9	555	6,709	570.2	112	8,248
it16862	22.7	1,281	16,057	5.9	1,251	15,692	4,786.0	123	674
ja9847	21.8	1,817	9,204	5.6	1,940	8,854	1,508.6	149	569
kz9976	21.4	3,265	9,327	1.3	3,356	9,526	612.9	1,077	2,803
mo14185	25.1	1,088	13,193	1.3	1,048	13,655	3,051.6	122	862
mu1979	20.3	817	1,815	0.1	725	1,911			
nu3496	19.2	560	3,260	0.2	544	2,260	57.2	130	1,628
pba38478	19.5	156	38,081				5,708.1	80	14,234
pcb3038	19.2	563	2,912	0.1	595	2,983	72.6	215	743
pla33810	72.1	110,345	30,517				15,527.6	18,111	2,359
pla7397	19.6	92,032	7,097				1,988.4	2,000	21
pla85900	392.9	149,610	77,213				100,495.3	22,100	5,819
pm8079	19.8	516	7,772	2.6	501	4,299	160.9	137	4,535
pr2392	19.2	2,007	2,257	0.2	1,908	2,289			
rl11849	19.3	2,398	11,779	1.0	2,365	11,540	2,359.4	289	357
rl1889	19.5	2,278	1,813	0.1	2,264	1,815			
rl5915	20.0	2,328	5,586	0.5	2,386	5,668	456.0	472	519
rl5934	19.1	2,382	5,863				222.2	914	1,567
rw1621	20.3	206	1,558	0.1	207	820			
sra104815	56.3	281	101,773				87,538.8	73	20,448
sw24978	26.1	1,658	23,732	2.3	1,650	24,191	3,225.2	584	8,093
tz6117	19.4	1,460	5,910	0.8	1,489	5,773	473.3	190	704
u1817	19.2	345	1,798	0.1	330	1,752			
usa13509	19.9	53,369	13,134				1,825.8	9,935	2,477
vm22775	21.9	1,101	21,944	26.4	1,081	20,374	8,527.3	121	1,004
ym7663	20.0	990	7,251				1,139.1	50	243

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

	Greedy			Optimal			Random		
Instance	**Sec**	z^*	**Elim**	**Sec**	z^*	**Elim**	**Sec**	z^*	**Elim**
ar9152	858.0	4,853	3,867	337.3	4,765	2,219	206.7	2,017	3,907
bby34656	9,091.4	234	11,800	15,220.3	237	11,005	2,960.1	143	11,698
bm33708	7,630.1	2,286	8,949	6,055.4	2,288	7,841	8,589.0	655	3,684
brd14051	657.9	1,527	4,925	2,219.5	1,499	4,288	2,095.8	283	787
ca4663	70.5	11,607	1,827	40.7	11,607	2,261	305.5	765	93
ch71009	4,638.1	8,736	44,013	16,570.2	8,464	20,816	71,455.5	695	2,520
d15112	1,151.8	4,699	6,431	4,045.6	4,610	4,072	1,733.4	1,340	2,118
d15112-2500	54.5	4,557	921	190.6	4,502	701	20.0	3,243	1,192
d18512	1,659.8	1,682	6,468	2,599.7	1,631	5,977	3,491.6	346	1,280
eg7146	275.3	1,783	67	25.2	1,791	4,940	256.2	266	1,995
ei8246	450.3	867	2,696	568.0	863	2,749	521.3	281	907
fi10639	529.5	2,241	3,518	514.2	2,274	2,713	949.1	523	1,221
fyg28534	6,134.2	219	11,138	7,938.8	220	10,247	1,412.2	199	15,323
gr9882	358.5	1,524	3,119	214.7	1,541	4,163	436.9	568	2,372
ho14473	444.9	884	9,407	213.0	891	2,280	137.1	434	9,947
it16862	930.1	1,965	6,556	1,061.1	1,929	5,969	2,379.2	490	1,330
ja9847	317.5	2,997	2,906	204.4	2,975	3,565	402.6	1,129	2,314
kz9976	298.5	4,971	4,399	327.7	5,024	4,276	233.9	4,777	4,901
mo14185	1,194.8	1,703	4,430	480.1	1,655	6,909	2,072.6	298	820
mu1979	34.3	1,208	412	16.9	1,202	390			
nu3496	50.7	874	1,851	22.8	920	671	16.0	567	2,534
pba38478	2,379.1	261	20,857	6,415.1	254	14,067	1,962.3	233	23,557
pcb3038	132.6	875	1,089	144.4	877	906	43.5	788	1,690
pla33810	4,579.0	166,242	12,556	8,568.4	165,217	5,726	976.7	162,113	22,247
pla7397	194.9	142,916	2,576				54.0	132,509	5,438
pla85900	26,555.8	220,778	32,678	37,165.1	221,516	25,079	4,603.2	160,639	50,140
pm8079	96.9	762	4,876	86.5	771	1,300	125.5	260	4,353
pr2392	76.2	3,094	936	130.2	3,035	807			
rl11849	168.3	3,933	7,329	712.1	3,866	4,431	803.3	1,567	2,217
rl1889	36.3	3,587	616	16.6	3,590	636			
rl5915	923.0	3,770	1,608	283.7	3,644	1,856	142.7	3,239	2,641
rl5934	278.1	3,622	1,994				230.8	1,386	796
rw1621	29.0	328	1,047	4.4	320	395			
sra104815	38,966.7	403	36,589	77,413.5	393	30,552	6,026.1	346	74,643
sw24978	2,150.4	2,473	8,884	2,392.6	2,625	8,077	7,757.3	324	586
tz6117	164.7	2,301	2,314	126.8	2,246	2,942	71.2	2,098	3,980
u1817	39.9	547	424	18.4	544	628			
usa13509	422.2	80,181	6,095	975.7	81,319	4,013	3,169.3	4,663	290
vm22775	2,304.6	1,787	5,275	2,013.7	1,777	5,341	543.6	1,584	14,105
ym7663	215.6	1,554	2,631				536.8	276	961

Table 16: Detailed results with $p = 15$ and $q = 10$

	Greedy			Optimal			Random		
Instance	**Sec**	z^*	**Elim**	**Sec**	z^*	**Elim**	**Sec**	z^*	**Elim**
ar9152	101.5	3,892	6,094	54.7	4,122	3,780	197.6	1,285	4,693
bby34656	1,422.9	190	21,063	66.4	206	31,216	15,775.4	27	1,188
bm33708	765.6	1,874	22,271	1,126.6	1,957	20,637	13,589.9	302	1,861
brd14051	379.6	1,200	7,810	215.3	1,238	8,585	1,672.2	346	1,600
ca4663	30.3	9,837	3,329	15.7	9,927	2,873	124.4	2,335	1,196
ch71009	2,260.1	7,117	52,065	3,223.8	7,259	48,966	46,376.2	1,173	9,768
d15112	190.9	3,890	10,457	77.7	3,841	11,611	1,955.6	1,064	1,822
d15112-2500	24.0	3,766	1,874	11.6	3,605	1,331	49.2	1,443	431
d18512	252.3	1,403	12,593	314.9	1,348	11,649	554.1	799	9,510
eg7146	113.3	1,433	2,876	61.0	1,318	3,836	420.5	156	1,282
ei8246	41.5	706	6,236	68.2	756	4,995	1,289.6	53	91
fi10639	110.2	1,826	6,909	117.8	1,909	6,158	983.1	406	1,455
fyg28534	561.2	191	19,101	450.5	178	20,306	12,159.4	18	372
gr9882	123.6	1,303	5,436	46.1	1,323	7,366	644.4	321	2,078
ho14473	69.2	701	11,738	39.6	724	4,892	802.5	75	7,541
it16862	266.0	1,536	10,662	575.4	1,641	7,709	1,457.6	506	4,776
ja9847	193.7	2,450	4,803	73.7	2,460	6,228	213.0	1,062	4,244
kz9976	59.7	3,997	7,540	99.9	4,237	6,308	390.4	1,618	3,210
mo14185	216.3	1,365	8,528	329.2	1,431	6,857	1,953.8	237	1,528
mu1979	21.6	1,031	1,424	1.3	1,009	1,618			
nu3496	22.6	695	2,863	6.7	755	1,427	64.4	134	1,396
pba38478	2,258.8	211	21,456	699.5	210	28,623	21,986.7	23	760
pcb3038	29.8	730	1,924	4.6	724	2,302	53.4	332	822
pla33810	1,642.0	139,533	19,023	1,003.6	137,252	21,929	9,500.8	35,384	4,090
pla7397	92.8	117,843	3,632				579.4	17,205	562
pla85900	8,033.0	189,381	51,042				44,002.0	58,320	17,463
pm8079	42.3	645	6,133	23.8	663	2,937	285.5	85	3,570
pr2392	23.9	2,553	1,552	4.8	2,558	1,750			
rl11849	236.2	3,210	6,226	76.1	3,304	8,174	356.1	1,936	4,628
rl1889	25.0	2,836	1,161	4.4	2,956	1,293			
rl5915	74.7	3,099	3,269	36.2	3,088	3,545	204.1	1,181	1,352
rl5934	48.1	2,956	4,008				101.8	1,659	2,267
rw1621	21.3	260	1,218	1.4	270	503			
sra104815	16,236.2	349	57,088	2,289.4	342	80,550	56,516.1	120	24,788
sw24978	438.2	2,088	16,338	408.2	2,134	16,481	21,010.3	37	42
tz6117	43.0	1,930	4,371	32.0	1,930	3,989	75.6	1,070	3,047
u1817	26.3	426	1,033	2.4	450	1,299			
usa13509	387.7	65,374	6,897	183.9	67,189	7,958	2,724.0	5,819	613
vm22775	705.0	1,334	12,496	327.6	1,431	15,348	9,392.5	94	309
ym7663	187.8	1,271	2,877				318.6	333	1,918

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

	Greedy			Optimal			Random		
Instance	**Sec**	z^*	**Elim**	**Sec**	z^*	**Elim**	**Sec**	z^*	**Elim**
ar9152	43.0	3,221	7,359	3.6	3,425	5,941	791.5	219	2,626
bby34656	210.5	160	29,052	29.2	169	32,296	26,030.3	8	154
bm33708	263.4	1,639	27,780	114.1	1,636	28,933	14,656.8	240	2,035
brd14051	143.2	1,040	9,774	45.3	1,051	11,436	3,531.1	104	338
ca4663	20.7	8,133	4,030	0.6	8,463	4,389	140.3	1,604	1,200
ch71009	911.9	5,976	59,375	212.7	6,097	64,589	43,293.6	1,013	11,996
d15112	42.2	3,347	13,337	18.0	3,349	13,281	4,308.4	256	328
d15112-2500	20.9	3,224	2,077	1.1	3,312	2,098	76.2	695	242
d18512	113.1	1,194	14,960	23.3	1,196	16,276	4,871.7	172	894
eg7146	34.9	1,152	5,608	21.7	1,189	4,646	822.5	56	320
ei8246	29.3	640	7,008	1.7	627	7,664	934.2	90	360
fi10639	75.9	1,518	8,058	15.7	1,607	9,041	754.0	490	2,556
fyg28534	220.8	148	22,247	40.2	156	25,876	3,928.9	69	8,298
gr9882	41.0	1,104	7,830	10.9	1,064	8,554	684.0	273	1,918
ho14473	46.8	617	12,579	10.9	611	5,646	369.7	157	8,954
it16862	37.3	1,325	15,013	34.0	1,424	14,190	1,090.9	538	5,892
ja9847	135.9	2,101	5,890	26.2	2,103	7,626	1,589.9	142	333
kz9976	46.1	3,516	8,070	14.1	3,489	8,299	769.1	903	1,846
mo14185	116.0	1,148	10,631	39.1	1,137	11,251	3,264.9	109	365
mu1979	19.9	897	1,561	1.8	817	1,417			
nu3496	22.5	584	2,856	0.9	613	2,002	101.5	50	1,203
pba38478	549.3	169	29,776	93.0	172	34,264	28,248.4	11	289
pcb3038	21.9	598	2,451	1.7	612	2,592	106.4	152	317
pla33810	386.2	112,324	26,367				11,345.9	28,636	3,982
pla7397	22.9	93,146	6,801				471.6	24,167	1,153
pla85900	1,923.9	151,644	67,592				71,477.0	33,745	10,204
pm8079	29.3	537	6,967	10.7	521	3,523	328.0	67	3,436
pr2392	22.0	2,051	1,956	0.5	2,089	2,147			
rl11849	86.7	2,479	9,243	34.9	2,634	9,404	770.9	1,010	3,010
rl1889	21.4	2,432	1,456	2.1	2,442	1,353			
rl5915	31.7	2,534	4,629	7.5	2,564	4,830	336.7	700	721
rl5934	25.2	2,465	5,012				470.8	379	280
rw1621	19.6	222	1,449	0.5	227	710			
sra104815	791.7	288	92,886				79,848.1	84	19,489
sw24978	120.0	1,784	20,497	26.1	1,790	22,492	9,207.5	170	957
tz6117	29.1	1,540	4,920	1.9	1,615	5,539	408.8	240	653
u1817	20.4	356	1,489	1.0	364	1,485			
usa13509	30.7	55,755	12,110	17.6	55,034	11,840	3,050.2	4,476	513
vm22775	94.5	1,168	19,504	20.2	1,250	20,788	6,883.8	163	1,396
ym7663	42.0	1,055	5,769				1,104.8	49	194

Table 18: Detailed results with $p = 15$ and $q = 20$

	Greedy			Optimal			Random		
Instance	**Sec**	z^*	**Elim**	**Sec**	z^*	**Elim**	**Sec**	z^*	**Elim**
ar9152	22.0	2,928	8,455	2.7	2,936	5,992	686.5	300	2,840
bby34656	51.7	144	32,349				21,304.7	13	481
bm33708	91.0	1,452	30,276	50.3	1,384	30,398	6,321.0	463	9,753
brd14051	39.3	880	12,171	3.0	937	13,142	2,818.7	138	642
ca4663	21.9	7,147	4,003	0.3	7,096	4,401	243.0	886	605
ch71009	1,077.1	5,316	57,019	82.0	5,493	66,604	66,248.8	589	6,065
d15112	67.7	2,908	12,597				6,027.6	141	140
d15112-2500	21.7	2,797	1,987	0.7	2,739	2,252	56.0	890	513
d18512	61.5	1,041	16,182				5,258.7	144	766
eg7146	37.0	997	5,377	7.8	968	5,898	556.0	90	959
ei8246	27.2	560	7,156	3.4	541	7,497	1,567.1	28	64
fi10639	40.0	1,374	8,956	2.0	1,366	10,006	2,582.5	69	104
fyg28534	121.0	137	24,277				11,581.6	19	1,131
gr9882	36.0	919	8,239	30.9	935	7,682	668.0	279	2,218
ho14473	20.6	543	14,003	2.6	541	6,419	490.6	112	8,435
it16862	48.3	1,209	14,862	10.6	1,213	15,247	4,528.2	120	496
ja9847	30.9	1,659	8,366	18.7	1,662	8,285	991.8	283	1,268
kz9976	30.1	3,005	8,564	2.7	3,114	9,263	419.8	1,216	3,442
mo14185	41.1	1,018	12,428	7.3	1,008	13,050	3,442.2	89	409
mu1979	21.5	697	1,444	0.3	707	1,861			
nu3496	20.0	521	3,088	0.8	517	2,119	69.6	90	1,422
pba38478	23.5	152	37,588				21,924.6	23	1,580
pcb3038	20.0	534	2,714	2.2	535	2,423	170.2	61	72
pla33810	201.2	102,026	28,569				22,995.6	6,325	418
pla7397	23.2	89,405	6,500				579.4	14,000	851
pla85900	1,113.3	138,402	73,153				64,295.7	37,857	12,397
pm8079	22.8	461	7,321	10.5	468	3,596	675.9	17	3,154
pr2392	19.7	1,814	2,096	0.4	1,811	2,214			
rl11849	19.0	2,365	11,754	0.2	2,354	11,750	2,296.5	253	297
rl1889	19.3	2,236	1,763	0.4	2,158	1,741			
rl5915	21.9	2,244	5,330	0.9	2,249	5,555	621.5	169	85
rl5934	19.9	2,294	5,702				505.0	307	285
rw1621	19.4	200	1,487	0.2	195	763			
sra104815	239.5	255	98,047				101,425.5	60	14,489
sw24978	56.6	1,547	22,518	10.9	1,576	23,556	9,148.7	189	1,046
tz6117	20.8	1,402	5,696	1.6	1,419	5,623	633.4	101	382
u1817	19.4	330	1,738	0.2	325	1,719			
usa13509	32.9	49,907	11,921				3,989.0	2,379	280
vm22775	60.0	1,008	20,227	66.6	1,036	19,285	12,847.2	47	129
ym7663	24.6	929	6,574				601.7	138	1,102

Table 19: Detailed results with $p = 20$ and $q = 5$

	Greedy			Optimal			Random		
Instance	**Sec**	z^*	**Elim**	**Sec**	z^*	**Elim**	**Sec**	z^*	**Elim**
ar9152	1,799.5	4,253	3,319	2,043.7	4,124	984	467.9	717	2,900
bby34656	12,980.8	209	8,239	86,403.8	212[3]	8,047	5,076.7	198	15,137
bm33708	12,611.4	2,030	7,318	62,650.5	1,983	5,466	8,459.7	601	3,840
brd14051	3,244.3	1,298	3,524	4,242.1	1,312	2,607	544.1	776	4,126
ca4663	112.8	10,310	1,584	75.6	10,310	1,179	173.3	1,826	542
ch71009	12,007.7	7,762	33,585	22,482.8	7,765	15,284	22,965.1	2,167	18,742
d15112	6,517.1	4,131	4,579	33,350.9	4,105	3,259	669.0	2,382	4,586
d15112-2500	98.2	3,956	768	708.7	4,018	475	60.8	1,174	255
d18512	12,180.1	1,470	5,164	86,402.5	1,459[4]	4,428	920.6	1,397	8,829
eg7146	332.2	1,438	59	59.6	1,396	4,278	87.5	447	3,464
ei8246	684.7	783	2,391	1,858.9	777	2,111	971.7	84	105
fi10639	2,954.4	1,941	2,742	2,014.4	1,970	2,034	180.9	1,213	4,649
fyg28534	86,410.4	193[1]	8,716	86,403.6	195[5]	8,613	2,318.3	113	8,316
gr9882	503.8	1,383	2,316	358.3	1,400	3,487	119.5	926	4,243
ho14473	1,603.9	759	8,908	2,033.1	768	1,495	128.1	417	9,944
it16862	1,846.7	1,646	4,546	2,100.6	1,708	4,295	611.1	1,036	5,876
ja9847	552.7	2,499	2,020	392.7	2,443	2,938	517.0	954	1,493
kz9976	663.0	4,422	3,826	595.6	4,497	3,423	269.7	4,213	5,521
mo14185	1,892.4	1,489	3,369	1,617.7	1,479	4,091	615.7	651	4,335
mu1979	40.2	1,045	359	24.7	1,031	358			
nu3496	74.6	775	1,655	110.8	771	403	35.4	264	1,665
pba38478	9,433.9	227	8,881	6,522.6	230	12,800	4,528.3	207	18,552
pcb3038	653.7	770	789	2,210.0	772	611	42.0	479	666
pla33810	11,729.3	146,906	8,342	86,402.3	147,649[6]	4,683	10,930.1	31,501	1,810
pla7397	503.9	123,435	1,890				306.2	45,123	1,125
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	4,576	151.0	666	869	51.8	591	6,059
pr2392	611.2	2,656	678	1,290.4	2,670	616			
rl11849	1,299.1	3,387	3,043	3,633.4	3,394	3,608	1,062.6	1,137	1,025
rl1889	161.7	3,066	431	292.0	3,083	465			
rl5915	2,464.7	3,262	1,001	1,620.7	3,225	1,421	129.4	1,884	1,470
rl5934	1,571.1	3,191	1,376				1,183.4	2,974	2,142
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	3,866.4	2,200	7,319	10,324.9	2,229	5,854	4,116.8	709	3,074
tz6117	402.7	2,040	1,843	942.0	2,016	2,535	588.6	142	209
u1817	133.0	479	334	90.4	479	562			
usa13509	1,769.8	68,360	4,738	5,759.7	70,119	2,175	361.9	46,139	5,159
vm22775	3,041.0	1,499	3,788	2,950.0	1,535	3,088	1,695.2	777	5,939
ym7663	556.2	1,298	1,349				209.4	488	2,339

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 20: Detailed results with $p = 20$ and $q = 10$

	Greedy			Optimal			Random		
Instance	**Sec**	z^*	**Elim**	**Sec**	z^*	**Elim**	**Sec**	z^*	**Elim**
ar9152	107.4	3,532	5,677	198.8	3,696	2,532	1,034.9	105	2,527
bby34656	2,638.8	176	17,529	3,031.9	187	14,042	106,159.2	1	10
bm33708	2,795.1	1,686	18,311	3,055.4	1,746	15,344	1,070.8	1,081	20,015
brd14051	1,355.6	1,127	5,589	743.7	1,104	6,313	785.5	519	3,958
ca4663	38.0	8,943	2,696	31.0	8,754	2,646	298.4	639	207
ch71009	5,911.5	6,605	40,813	9,992.6	6,638	32,464	39,182.1	1,216	10,120
d15112	308.9	3,498	9,762	659.9	3,655	6,513	1,516.7	1,427	2,198
d15112-2500	29.9	3,386	1,514	13.6	3,475	1,178	39.5	1,472	631
d18512	1,746.0	1,264	9,976	1,219.8	1,296	8,602	3,647.3	289	1,130
eg7146	160.4	1,200	2,319	66.6	1,138	3,158	185.0	231	2,866
ei8246	96.6	657	5,085	194.8	663	3,211	328.9	310	2,009
fi10639	228.1	1,650	5,774	269.0	1,693	4,920	1,019.7	359	1,174
fyg28534	6,690.4	161	12,036	3,674.6	170	12,212	10,136.5	27	940
gr9882	246.3	1,184	4,766	212.1	1,180	4,561	413.0	499	2,383
ho14473	134.8	635	10,995	270.3	667	3,099	482.0	141	8,304
it16862	424.8	1,443	9,148	956.0	1,464	6,678	2,240.5	380	2,000
ja9847	287.4	2,241	3,403	197.5	2,104	4,255	49.1	1,910	7,513
kz9976	201.3	3,689	5,820	199.1	3,789	5,109	648.1	1,102	1,968
mo14185	531.3	1,244	6,489	599.6	1,251	5,817	1,558.5	283	2,314
mu1979	23.0	918	1,207	4.3	868	1,162			
nu3496	32.7	630	2,418	19.9	655	952	31.0	237	1,908
pba38478	9,253.6	184	20,278	3,877.9	195	15,449	4,812.8	102	12,392
pcb3038	47.8	652	1,631	61.8	678	1,263	80.7	218	396
pla33810	3,363.2	127,201	14,715	2,677.9	131,606	15,637	8,764.7	36,361	4,308
pla7397	132.1	106,902	3,514				627.7	12,650	515
pla85900	15,816.8	173,042	37,704				17,189.7	84,403	34,853
pm8079	98.8	579	5,026	77.3	555	1,944	82.6	232	4,985
pr2392	28.9	2,223	1,315	18.6	2,301	1,259			
rl11849	391.5	2,848	6,145	355.9	3,008	4,744	1,853.7	403	331
rl1889	25.7	2,630	1,040	10.1	2,753	822			
rl5915	95.6	2,790	2,705	98.5	2,821	2,708	126.9	1,511	1,812
rl5934	56.2	2,605	3,804				313.5	733	629
rw1621	23.8	243	1,108	2.6	237	383			
sra104815	26,138.9	313	49,418	25,140.9	315	44,098	158,029.1	33	2,318
sw24978	823.6	1,888	13,425	1,013.5	1,959	12,385	11,728.4	97	122
tz6117	72.1	1,761	3,592	90.7	1,769	3,219	122.9	847	2,206
u1817	27.2	381	992	10.1	423	720			
usa13509	967.2	59,273	6,046	678.6	61,276	6,243	1,037.1	18,880	2,904
vm22775	1,289.1	1,247	10,220	1,312.2	1,236	9,392	2,297.0	546	5,596
ym7663	250.5	1,185	2,399	158.1	1,124	2,765	1,179.3	37	51

Table 21: Detailed results with $p = 20$ and $q = 15$

	Greedy			Optimal			Random		
Instance	**Sec**	z^*	**Elim**	**Sec**	z^*	**Elim**	**Sec**	z^*	**Elim**
ar9152	92.8	3,134	6,058	68.3	3,253	3,590	469.7	507	3,310
bby34656	341.8	153	27,686	389.2	162	27,391	16,555.1	26	1,386
bm33708	799.7	1,483	23,628	348.6	1,576	26,419	16,196.9	180	1,107
brd14051	323.3	967	7,883	133.9	987	9,346	3,521.7	88	212
ca4663	25.6	7,639	3,632	7.2	7,713	3,658	142.0	1,508	1,147
ch71009	1,637.1	5,732	52,662	819.2	5,602	58,724	40,719.4	1,203	9,756
d15112	220.5	3,067	10,108	172.4	3,028	9,938	1,933.2	841	2,213
d15112-2500	28.5	2,984	1,535	6.3	2,969	1,708	73.2	622	230
d18512	271.2	1,107	12,533	322.4	1,092	11,677	593.2	658	9,315
eg7146	57.4	1,069	4,255	79.1	1,069	3,179	249.5	170	2,297
ei8246	51.6	580	6,153	23.9	579	6,411	734.2	132	624
fi10639	133.3	1,441	6,980	66.6	1,461	7,615	1,180.1	297	924
fyg28534	340.8	142	21,211	182.4	150	23,127	8,027.7	38	2,369
gr9882	128.1	1,000	6,101	27.2	1,017	7,699	521.9	321	2,378
ho14473	80.8	566	12,184	43.8	574	4,656	787.9	60	7,590
it16862	87.8	1,269	13,521	346.8	1,276	9,811	3,850.8	157	651
ja9847	205.5	1,916	4,969	156.4	1,945	5,128	731.7	430	1,870
kz9976	95.7	3,277	6,909	35.2	3,313	7,674	519.2	1,163	2,726
mo14185	184.8	1,066	9,264	169.4	1,104	8,866	1,805.8	232	1,925
mu1979	24.0	706	1,187	4.6	742	1,150			
nu3496	25.2	556	2,640	2.7	555	1,785			
pba38478	895.9	159	28,818	173.6	162	33,241	15,079.7	41	3,610
pcb3038	27.9	544	2,148	2.4	580	2,504	149.4	79	86
pla33810	633.3	108,904	23,793				6,556.3	38,053	7,854
pla7397	29.7	92,000	6,093				1,055.6	4,473	106
pla85900	3,300.1	144,430	62,260				83,129.9	26,005	5,716
pm8079	37.6	490	6,380	15.8	485	3,417	320.6	60	3,390
pr2392	23.2	1,937	1,707	2.4	1,953	1,920			
rl11849	103.8	2,408	9,145	57.3	2,529	8,857	2,528.4	192	127
rl1889	21.8	2,322	1,384	2.6	2,382	1,352			
rl5915	43.3	2,328	4,231	14.3	2,418	4,492	374.2	560	496
rl5934	32.7	2,370	4,769				374.7	518	488
rw1621	19.9	201	1,362	0.8	210	629			
sra104815	2,048.9	264	86,483				188,607.9	18	1,103
sw24978	272.8	1,582	18,342	127.6	1,714	19,881	4,264.8	434	5,164
tz6117	35.8	1,471	4,601	8.5	1,519	5,130	218.4	560	1,385
u1817	22.2	339	1,413	2.5	345	1,338			
usa13509	78.3	53,107	10,355	17.4	51,562	11,854	1,456.0	12,658	2,265
vm22775	263.8	1,051	16,745	391.0	1,120	15,240	7,737.0	121	736
ym7663	74.3	983	4,797	81.4	1,017	3,805	278.2	287	2,386

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

	Greedy			Optimal			Random		
Instance	**Sec**	z^*	**Elim**	**Sec**	z^*	**Elim**	**Sec**	z^*	**Elim**
ar9152	30.4	2,618	8,010	6.3	2,753	5,746	630.7	300	2,890
bby34656	127.7	139	30,872				19,810.7	14	589
bm33708	815.1	1,335	23,444	142.2	1,342	28,906	14,399.9	217	2,147
brd14051	81.7	838	11,302	28.0	895	11,836	2,005.9	206	1,488
ca4663	25.0	6,692	3,702	1.6	6,668	4,130	256.7	709	434
ch71009	1,922.8	4,943	51,440	554.6	5,269	61,000	61,940.5	587	6,293
d15112	119.0	2,737	11,953				2,989.4	460	902
d15112-2500	23.7	2,621	1,874	1.7	2,678	2,070	44.7	1,119	631
d18512	99.4	1,000	15,173	12.9	1,006	17,086	4,209.5	177	1,310
eg7146	82.8	897	3,935	5.9	938	5,968	449.8	105	1,270
ei8246	76.3	529	5,876	10.8	513	7,000	768.7	117	666
fi10639	89.7	1,268	7,666	9.2	1,315	9,429	1,386.3	224	743
fyg28534	438.8	131	22,115				7,381.8	35	3,817
gr9882	49.4	841	7,419	27.1	850	7,533	862.8	180	1,500
ho14473	29.9	521	13,245	13.1	511	5,548	406.7	134	8,684
it16862	140.6	1,161	12,338	49.9	1,163	14,044	3,819.8	152	951
ja9847	57.5	1,567	7,527	69.5	1,586	6,622	1,711.3	93	291
kz9976	40.5	2,896	8,094	8.4	2,983	8,681	378.1	1,301	3,528
mo14185	56.7	947	11,972	29.4	952	11,914	1,826.3	217	2,071
mu1979	23.0	677	1,253	2.7	653	1,434			
nu3496	22.6	488	2,800	0.8	502	2,081	45.1	130	1,774
pba38478	26.8	151	37,438				14,194.5	42	4,491
pcb3038	21.2	508	2,517	4.4	501	2,276	57.8	230	848
pla33810	692.8	96,933	25,154				10,561.4	23,410	4,351
pla7397	39.2	81,104	5,604				1,876.9	2,000	21
pla85900	4,213.0	133,047	62,549				100,595.7	18,310	3,665
pm8079	28.9	428	6,964	11.4	445	3,529	207.5	97	4,063
pr2392	20.4	1,765	2,048	0.8	1,779	2,078			
rl11849	19.6	2,343	11,609	0.7	2,297	11,557	1,943.9	334	491
rl1889	20.1	2,124	1,547	0.4	2,100	1,706			
rl5915	25.5	2,110	5,013	3.5	2,194	5,164	159.1	1,011	1,893
rl5934	23.6	2,193	5,194				808.0	80	47
rw1621	19.7	182	1,423	0.4	185	711			
sra104815	2,018.0	242	86,999				57,130.8	87	27,605
sw24978	167.7	1,474	20,194	56.7	1,514	21,879	7,313.8	239	1,884
tz6117	24.5	1,344	5,195	8.0	1,335	5,054	336.7	261	1,042
u1817	19.4	321	1,653	0.5	310	1,628			
usa13509	61.0	47,096	10,855				2,704.2	4,649	822
vm22775	128.5	955	18,609	137.3	1,005	18,196	16,332.7	33	78
ym7663	33.9	860	6,173	5.3	851	6,857	592.9	140	934

B 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 23–26 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 23: 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	7,108.8	23.9	8,221.0	18.6	8,304.9	17.9	9,290.0
bm33708	924.6	6,659.0	22.9	7,068.8	19.7	8,214.4	18.9	8,268.1
brd14051	20.5	1,028.6	33.1	1,464.2	19.3	1,306.5	18.8	1,429.8
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	1,406.2	20.8	1,458.0	19.1	1,592.0	19.6	1,811.1
d15112-2500	20.5	66.7	18.8	61.9	17.5	70.3	18.9	70.7
d18512	160.3	2,242.4	22.8	2,337.4	20.1	2,497.6	19.1	2,486.1
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	4,891.5	125.2	5,949.3	19.1	5,360.3	17.7	5,842.3
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	1,751.9	22.8	1,881.2	19.7	1,873.2	19.0	2,114.2
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	1,155.6	19.2	1,197.6	19.2	1,314.8	18.9	1,425.1
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	1,309.1	9,329.2	19.0	9,611.5	17.4	8,158.0	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	6,553.3	99.8	7,884.3	19.8	8,058.4	22.2	9,689.0
pla7397	39.4	351.1	19.4	346.9	19.0	452.9	19.4	599.2
pla85900	1,031.3	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	1,272.2
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	3,686.5	21.4	3,913.2	19.7	3,852.7	19.2	4,139.1
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	1,031.4	30.1	1,135.7	19.8	1,311.4	19.2	1,391.8
vm22775	339.0	2,801.4	18.9	3,057.9	20.0	3,000.5	17.6	3,803.0
ym7663	28.8	295.1	33.2	400.8	17.8	363.3	19.2	443.0

Table 24: 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	1,772.5	7,558.7	337.8	7,517.3	69.1	8,049.8	32.9	8,992.0
bm33708	2,839.3	7,328.2	268.7	6,957.4	53.7	7,483.8	25.4	6,955.4
brd14051	373.1	1,183.2	113.9	1,352.2	42.5	1,379.0	22.2	1,329.3
ca4663	44.3	145.7	23.7	118.3	19.0	133.4	20.9	148.5
ch71009	1,403.3	20,580.6	581.5	24,159.0	39.2	24,813.4	301.7	31,975.0
d15112	268.9	1,484.9	46.5	1,454.5	31.1	1,613.6	20.4	1,522.2
d15112-2500	27.6	64.1	20.5	62.1	19.6	67.3	19.4	71.5
d18512	488.8	2,267.9	97.1	2,232.1	30.0	2,341.6	24.2	2,315.1
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	2,818.1	6,074.0	177.4	5,516.5	21.6	4,522.7	79.1	6,151.0
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	1,613.3	57.6	1,775.8	25.7	1,673.4	22.7	1,986.4
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	1,272.2	71.6	1,228.8	30.6	1,166.6	25.1	1,270.4
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	2,005.4	9,604.1	20.1	7,330.5	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	1,947.4	7,280.2	340.1	7,769.5	65.7	7,511.5	72.1	8,473.5
pla7397	91.7	398.4	48.9	435.9	21.6	468.6	19.6	477.8
pla85900	12,980.6	41,836.4	2,112.6	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	1,061.7
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	7,696.7	69,347.3	184.6	72,591.8	56.3	70,570.2
sw24978	1,182.5	3,778.1	155.0	3,676.5	34.9	3,776.4	26.1	3,802.9
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	1,055.6	178.8	1,451.7	21.7	1,155.9	19.9	1,364.8
vm22775	1,095.3	2,976.6	204.8	3,095.8	35.3	3,143.4	21.9	3,159.9
ym7663	168.7	314.8	56.6	342.0	26.6	353.4	20.0	424.4

Table 25: 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	9,091.4	16,154.8	1,422.9	8,341.4	210.5	7,748.4	51.7	8,267.3
bm33708	7,630.1	10,562.7	765.6	6,965.5	263.4	7,092.4	91.0	7,358.6
brd14051	657.9	1,293.2	379.6	1,671.9	143.2	1,428.4	39.3	1,365.0
ca4663	70.5	133.5	30.3	137.0	20.7	128.1	21.9	149.4
ch71009	4,638.1	24,469.5	2,260.1	26,286.9	911.9	28,453.1	1,077.1	29,831.6
d15112	1,151.8	1,823.9	190.9	1,489.5	42.2	1,514.4	67.7	1,672.6
d15112-2500	54.5	86.7	24.0	65.1	20.9	70.1	21.7	77.2
d18512	1,659.8	2,651.2	252.3	2,123.7	113.1	2,290.1	61.5	2,512.3
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	6,134.2	8,915.5	561.2	5,179.8	220.8	5,670.0	121.0	5,703.1
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	1,831.7	266.0	1,827.3	37.3	1,673.7	48.3	1,781.9
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	1,194.8	1,551.8	216.3	1,303.0	116.0	1,264.3	41.1	1,289.2
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	2,379.1	8,965.4	2,258.8	9,458.6	549.3	9,389.7	23.5	9,998.1
pcb3038	132.6	129.4	29.8	92.5	21.9	101.5	20.0	95.1
pla33810	4,579.0	8,775.7	1,642.0	7,478.3	386.2	8,139.6	201.2	8,256.6
pla7397	194.9	520.3	92.8	524.3	22.9	524.4	23.2	410.8
pla85900	26,555.8	66,566.2	8,033.0	46,521.9	1,923.9	47,950.9	1,113.3	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	1,110.5	19.0	971.0
rl1889	36.3	50.4	25.0	52.3	21.4	49.6	19.3	47.8
rl5915	923.0	1,151.9	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	2,150.4	3,733.7	438.2	3,388.6	120.0	3,648.4	56.6	3,750.4
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	1,449.6	387.7	1,377.7	30.7	1,243.4	32.9	1,294.1
vm22775	2,304.6	3,065.4	705.0	3,543.6	94.5	3,023.7	60.0	3,372.0
ym7663	215.6	417.9	187.8	415.4	42.0	384.5	24.6	393.7

Table 26: 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	1,799.5	2,314.1	107.4	301.2	92.8	347.5	30.4	356.8
bby34656	12,980.8	18,784.6	2,638.8	8,067.5	341.8	7,768.0	127.7	7,740.9
bm33708	12,611.4	15,287.7	2,795.1	8,236.2	799.7	7,670.6	815.1	7,677.7
brd14051	3,244.3	8,223.2	1,355.6	1,534.6	323.3	1,434.5	81.7	1,350.2
ca4663	112.8	161.7	38.0	123.6	25.6	139.4	25.0	156.3
ch71009	12,007.7	25,290.6	5,911.5	29,321.6	1,637.1	26,838.4	1,922.8	29,306.5
d15112	6,517.1	10,397.4	308.9	1,805.6	220.5	1,656.7	119.0	1,732.3
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	1,746.0	2,598.7	271.2	2,358.0	99.4	2,375.0
eg7146	332.2	325.7	160.4	284.6	57.4	243.0	82.8	277.3
ei8246	684.7	1,593.4	96.6	486.5	51.6	483.2	76.3	495.4
fi10639	2,954.4	3,380.1	228.1	697.7	133.3	764.9	89.7	758.0
fyg28534			6,690.4	6,790.2	340.8	5,358.8	438.8	5,830.3
gr9882	503.8	748.9	246.3	704.3	128.1	627.4	49.4	700.7
ho14473	1,603.9	1,264.5	134.8	400.8	80.8	419.4	29.9	304.8
it16862	1,846.7	2,256.4	424.8	1,717.9	87.8	1,601.0	140.6	1,685.0
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	1,892.4	3,821.6	531.3	1,298.0	184.8	1,342.7	56.7	1,232.8
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	9,433.9	12,756.2	9,253.6	12,363.4	895.9	11,006.8	26.8	9,451.3
pcb3038	653.7	598.2	47.8	110.2	27.9	105.1	21.2	94.9
pla33810	11,729.3	16,791.0	3,363.2	8,575.8	633.3	7,655.9	692.8	8,036.5
pla7397	503.9	686.9	132.1	484.6	29.7	471.6	39.2	525.7
pla85900			15,816.8	49,604.0	3,300.1	47,023.4	4,213.0	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	1,299.1	2,112.3	391.5	1,050.3	103.8	1,067.2	19.6	911.8
rl1889	161.7	88.5	25.7	49.2	21.8	51.4	20.1	47.6
rl5915	2,464.7	4,018.6	95.6	311.2	43.3	287.4	25.5	278.4
rl5934	1,571.1	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	2,048.9	66,830.4	2,018.0	71,468.4
sw24978	3,866.4	4,281.3	823.6	3,659.6	272.8	3,782.4	167.7	3,500.8
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	1,769.8	5,681.3	967.2	1,644.6	78.3	1,243.2	61.0	1,323.2
vm22775	3,041.0	3,410.7	1,289.1	3,486.9	263.8	3,225.2	128.5	3,399.1
ym7663	556.2	628.7	250.5	441.3	74.3	407.0	33.9	360.5

References

D. Aloise and C. Contardo. A sampling-based exact algorithm for the solution of the minimax diameter clustering problem. Journal of Global Optimization, pages 1–18, 2018.

P. Belotti, M. Labbé, F. Maffioli, and M. M. Ndiaye. A branch-and-cut method for the obnoxious p-median problem. 4OR, 5(4):299–314, 2006.

O. Berman and Z. Drezner. A new formulation for the conditional p-median and p-center problems. Operations Research Letters, 36(4):481–483, 2008.

O. Berman and D. Simchi-Levi. Conditional location problems on networks. Transportation Science, 24(1), 1990.

N. Boland, J. Dethridge, and I. Dumitrescu. Accelerated label setting algorithms for the elementary resource constrained shortest path problem. Operations Research Letters, 34(1):58–68, 2006.

H. Calik, M. Labbé, and H. Yaman. p-center problems. In Location Science, pages 79–92. Springer, 2015.

B. Callaghan, S. Salhi, and J. Brimberg. Optimal solutions for the continuous p-centre problem and related-neighbour and conditional problems: A relaxation-based algorithm. Journal of the Operational Research Society, 70(2):192–211, 2019.

D. Chen and R. Chen. New relaxation-based algorithms for the optimal solution of the continuous and discrete p-center problems. Computers & Operations Research, 36(5):1646–1655, 2009.

D. Chen and R. Chen. A relaxation-based algorithm for solving the conditional p-center problem. Operations Research Letters, 38(3):215–217, 2010.

R. Chen and G. Y. Handler. Relaxation method for the solution of the minimax location-allocation problem in euclidean space. Naval Research Logistics (NRL), 34(6):775–788, 1987.

R. Chen and Y. Handler. The conditional p-center problem in the plane. Naval Research Logistics (NRL), 40(1):117–127, 1993.

C. Contardo. Decremental clustering for the solution of p-dispersion problems to proven optimality. INFORMS Journal on Optimization, 2019. Forthcoming.

C. Contardo and J. A. Sefair. A progressive approximation approach for the exact solution of sparse large-scale binary interdiction games. techreport G–2019–49, Cahiers du GERAD, 2019.

C. Contardo, M. Iori, and R. Kramer. A scalable exact algorithm for the vertex p-center problem. Computers & Operations Research, 103:211–220, 2019.

M. S. Daskin and K. L. Maass. The p-median problem. In Location science, pages 21–45. Springer, 2015.

M. Delattre and P. Hansen. Bicriterion cluster analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2(4):277–291, 1980.

T. Drezner and Z. Drezner. Sequential location of two facilities: comparing random to optimal location of the first facility. Annals of Operations Research, 246(1–2):5–18, 2016.

Z. Drezner. Conditional p-center problems. Transportation Science, 23(1):51–53, 1989.

Z. Drezner. On the conditional p-median problem. Computers & operations research, 22(5):525–530, 1995.

E. Erkut. The discrete p-dispersion problem. European Journal of Operational Research, 46(1):48–60, 1990.

C. A. Irawan, S. Salhi, and M. P. Scaparra. An adaptive multiphase approach for large unconditional and conditional p-median problems. European Journal of Operational Research, 237(2):590–605, 2014.

M. J. Kuby. Programming models for facility dispersion: The p-dispersion and maxisum dispersion problems. Geographical Analysis, 19(4):315–329, 1987.

R. Martinelli, D. Pecin, and M. Poggi. Efficient elementary and restricted non-elementary route pricing. European Journal of Operational Research, 239(1):102–111, 2014.

D. Pisinger. Upper bounds and exact algorithms for p-dispersion problems. Computers & Operations Research, 33(5):1380–1398, 2006.

G. Righini and M. Salani. New dynamic programming algorithms for the resource constrained elementary shortest path problem. Networks: An International Journal, 51(3):155–170, 2008.

B. Saboonchi, P. Hansen, and S. Perron. Maxminmin p-dispersion problem: A variable neighborhood search approach. Computers & Operations Research, 52:251–259, 2014.

D. Sayah and S. Irnich. A new compact formulation for the discrete p-dispersion problem. European Journal of Operational Research, 256(1):62–67, 2017.

M. Statman. How many stocks make a diversified portfolio? Journal of Financial and Quantitative Analysis, 353–363, 1987. doi: 10.2307/2330969.