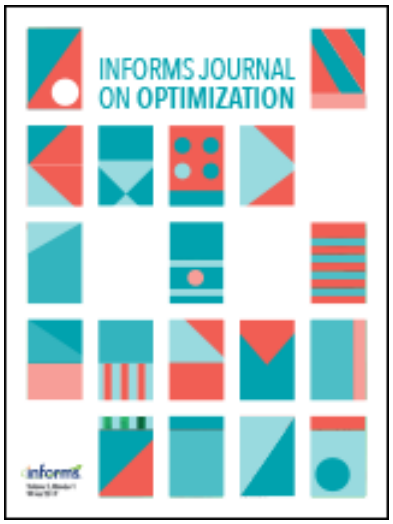


INFORMS Journal on Optimization

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Decremental Clustering for the Solution of p-Dispersion Problems to Proven Optimality

Claudio Contardo

To cite this article:
Claudio Contardo (2020) Decremental Clustering for the Solution of p-Dispersion Problems to Proven Optimality. INFORMS Journal on Optimization

Published online in Articles in Advance 21 Apr 2020

. https://doi.org/10.1287/ijoo.2019.0027

http://pubsonline.informs.org/journal/ijoo

INFORMS JOURNAL ON OPTIMIZATION
Articles in Advance, pp. 1–11
ISSN 2575-1484 (print), ISSN 2575-1492 (online)

Decremental Clustering for the Solution of p-Dispersion Problems to Proven Optimality

Claudio Contardo[a]

[a] Department of Management and Technology, ESG UQAM, GERAD and CIRRELT, Montreal, Quebec H2X 1L7, Canada
Contact: claudio.contardo@gerad.ca, https://orcid.org/0000-0001-7595-3904 (CC)

Received: March 27, 2019
Revised: July 2, 2019; September 8, 2019
Accepted: September 12, 2019
Published Online in Articles in Advance: April 21, 2020

https://doi.org/10.1287/ijoo.2019.0027

Abstract. Given n points, a symmetric dissimilarity matrix D of dimensions $n \times n$, and an integer $p \geq 2$, the p-dispersion problem (pDP) consists of selecting a subset of exactly p points in such a way that the minimum dissimilarity between any pair of selected points is maximum. The pDP is $\mathcal{NP}$ hard when p is an input of the problem. We propose a decremental clustering method to reduce the problem to the solution of a series of smaller pDPs until reaching proven optimality. A k-means algorithm is used to construct and refine the clusterings along the algorithm's execution. The proposed method can handle problems orders of magnitude larger than the limits of the state-of-the-art solver for the pDP for small values of p.

Keywords: decremental clustering • p-dispersion problem • exact algorithm • k-means

1. Introduction

In the p-dispersion problem (pDP), we are given a set of n points, a symmetric dissimilarity matrix $D = \{D(i,j) : 1 \leq i,j \leq n\}$ satisfying $D(i,j) \geq 0$ for every $1 \leq i,j \leq n$ and $D(i,i) = 0$ for every $1 \leq i \leq n$, and an integer $p \geq 2$. The objective is to select p points from the set of n so as to maximize the minimum pairwise dissimilarity within the selected points. The pDP, as noticed by Erkut (1990), is $\mathcal{NP}$ hard when p makes part of the input parameters (otherwise, it can be solved in $O(n^p)$ time by exhaustive enumeration). We denote this problem for given input parameters D and p (n is implicitly given in the dimensions of D) as $\texttt{pDP}(\texttt{D},\texttt{p})$.

The pDP arises in a number of practical contexts. In location analysis, a pDP can help decide the placement of installations in which proximity may be hazardous—such as is the case of power plants, oil storage tanks, or ammunition—or the location of retail stores to prevent cannibalization (Kuby 1987). In multiobjective optimization, in the presence of multiple solutions for a given optimization problem, one may solve a pDP to select a subset of those solutions as complementary as possible with respect to the values for each of the objectives (Saboonchi et al. 2014). In finance, a pDP can be used as a proxy to build diversified portfolios, which are known to provide low risk (Statman 1987).

The state-of-the-art solver for the pDP (Sayah and Irnich 2017) relies on the solution of an integer program containing $O(n + \Delta)$ variables and constraints, where Δ is the number of distinct entries in the dissimilarity matrix D. The model remains tractable for medium-sized problems, but memory/time limits may prevent the solution of problems containing more than a few hundred nodes. The problem size and the large amount of symmetries impact the model's performance.

Our article contributes to narrowing this gap by allowing the solution of potentially much larger problems (in terms of the number of nodes n) under the assumption that parameter p remains low (typically ≤ 10 when going large scale). To this end, we introduce a decremental clustering scheme that, in a dynamic fashion, forms clusters of points and constructs instances of the pDP that are smaller in size and with much better numerical properties (most notably a much smaller amount of symmetries). These smaller instances are shown to provide upper bounds of the original problem, and they are much more tractable than the original pDP. The proposed iterative mechanism can scale and solve problems containing up to 100,000 nodes to proven optimality within reasonable time limits; this is orders of magnitude larger than the scope of previous methods. Although clustering techniques are of common use in the development of metaheuristics, this is to the best of our knowledge the first time that they are embedded within an exact solver for combinatorial optimization problems arising in location analysis.

The remainder of this article is organized as follows. In Section 2, we present a review of the relevant scientific literature related to this article. In Section 3, we present the decremental clustering framework. In Section 4, we present the results of our computational campaign to assess the effectiveness of our method. Finally, Section 5 concludes the paper.

2. Literature Review

Applications of the pDP can be found in multiple fields, including location analysis, multiobjective optimization, and portfolio optimization. Kuby (1987) mentions the importance of locating facilities as far as possible from each other when they represent a potential hazard for the surrounding communities. The same author also mentions applications in store location. If two stores of the same chain are located too close, cannibalism may prevent them from selling at full potential. Saboonchi et al. (2014) discuss an application of the pDP in multiobjective optimization. If the Pareto frontier of a problem contains multiple solutions, one shall solve a pDP to find p such solutions with distinct features. The same authors also describe an application in portfolio optimization to—given a set of potential investment opportunities—choose a subset that reduces the closeness in terms of features between the different investment options so as to reduce the risk associated with the portfolio. The problem of selecting diversified portfolios has been recognized as most important in finance (Statman 1987).

The pDP is tightly related to facility location problems (FLPs) (Laporte et al. 2015). In its simplest version, an FLP corresponds to the problem of, given a set of potential facility locations and a set of customers, selecting a subset of potential facility locations and allocating the customers to those facilities at minimum total cost. Facility location problems and applications have been widely studied in the scientific literature, and several comprehensive surveys have been recently published that take into account several of the latest advances in the field (Melo et al. 2009, Laporte et al. 2015). The pDP differs from a typical FLP model in the importance of the notion of customer. Although they are of key importance for the right choice of the facilities in the FLP, in the pDP they are irrelevant. Only the facility locations are of importance, and their choice must reflect the objective function to be optimized: to maximize the minimum distance between any two chosen facilities. One particular variant of FLP, namely the obnoxious p-median problem (OpMP) (Belotti et al. 2006), is closely related to the pDP. In the OpMP, we are given a set of potential facilities and customers. A planner must select the location of p facilities in such a way that the sum of the distances from each customer to its closest facility is maximized. This problem arises in the location of hazardous or obnoxious installations.

The pDP is also related to clustering problems and more specifically, the maximin split clustering problem (MMSCP). In the MMSCP, we are given a set N of observations, a dissimilarity matrix D, and a target number of clusters p. One has to group the observations into p groups such that the minimum dissimilarity between any two observations belonging to different groups is maximized. The MMSCP, unlike the pDP, is polynomially solvable (Delattre and Hansen 1980).

Regarding the methodological contributions to the solution of the pDP, a handful of articles have dealt with the problem of solving the pDP to proven optimality. Pisinger (2006) introduces a quadratic formulation for the pDP, which is then partially solved by a series of relaxations, including semidefinite programming, and reformulation-linearization. The bounds are embedded within a branch-and-bound framework, and the author reports the solution of problems containing a few hundred nodes. Kuby (1987) introduces a mixed integer linear formulation of the problem with a series of Big M coefficients. The model can be seen as a linearization of that of Pisinger (2006), even though it was introduced almost 20 years earlier. The model is more compact than that of Pisinger (2006) but provides much weaker upper bounds. Sayah and Irnich (2017) introduce a novel pure binary compact formulation of the problem that the authors solve by branch and cut. Clique-like inequalities are used to strengthen the model. Problems with up to 1,000 nodes are solved to proven optimality as reported by the authors. The same authors also mention that linear and binary search methods may be used with the different formulations to speed up the solution process. Such techniques have already been studied by Chandrasekaran and Daughety (1981) and Pisinger (2006) for the pDP. For this to be beneficial, the models need to exploit the availability of lower and upper bounds to fathom nonpromising branches of the implicit enumeration tree.

The decremental clustering method introduced in this article is tightly related to other decremental relaxation mechanisms recently introduced in the literature for the solution of other minimax (or equivalently, maximin) combinatorial optimization problems to proven optimality. In the vertex p-center problem (VPCP), for the same input parameters n, D, and p, one has to select p points and allocate the remaining points to their closest centers in such a way that the maximum dissimilarity between a node and its assigned center is minimized. Chen and Chen (2009) and Contardo et al. (2019) propose decremental relaxation mechanisms to ignore some node allocation constraints, which are only added as needed. The relaxed problems can thus be modeled as smaller VPCPs in an iterative manner. Contardo et al. (2019) report the solution of problems containing up to 1 million observations to proven optimality. The minimax diameter clustering problem (MMDCP) is another problem for which the decremental relaxation mechanism has proven useful. In the

MMDCP, given n points, a dissimilarity matrix D, and an integer $k \geq 2$, the objective is to group the observations into k clusters such as to minimize the maximum intracluster dissimilarity. Aloise and Contardo (2018) introduced a sampling mechanism to solve the MMDCP as a series of smaller MMDCPs in a dynamic fashion, allowing the solution to proven optimality of problems containing up to 600,000 observations.

Using clustering mechanisms for finding feasible solutions for hard combinatorial optimization problems is not something totally new in the operations research literature. Embedding a clustering scheme within a heuristic solver has been common practice for many years and for multiple classes of problems. In vehicle routing and scheduling, the so-called cluster-first, route-second (Solomon 1987, Bräysy and Gendreau 2005) and route-first, cluster-second (Beasley 1983, Prins et al. 2014) paradigms are both based on combining routing and clustering techniques so as to reduce the computational burden associated with the routing or scheduling substructures. None of those techniques, however, provide any guarantee of optimality.

3. Decremental Clustering

In this section, we describe the decremental clustering method for the pDP. This section is subdivided in five subsections. In the first subsection, we provide the theoretical foundations and a high-level description of the method. The next four sections describe the different procedures of the method.

3.1. High-level Description and Theoretical Foundations

Let us introduce some notation and vocabulary first. A *clustering* of the n nodes, denoted by $\mathscr{C}$, is a family $\{C_i : i = 1 \dots m\}$ such that (i) $C_i \cap C_j = \emptyset$ for every $1 \leq i < j \leq m$ and (ii) $\cup\{C_i : i = 1 \dots m\} = \{1 \dots n\}$. A clustering $\mathscr{C}$ is said to be *sufficiently refined* if, for every set $C_i \in \mathscr{C}$, $D(C_i) := \max\{D(u,v) : u, v \in C_i, u < v\} < z^*$, where z^* is the optimal value of problem $\texttt{pDP}(\texttt{D},\texttt{p})$. For practical purposes, it is sufficient to test the refinement of a clustering with respect to a lower bound $l \leq z^*$. The correctness of the decremental clustering method is supported on the following result.

Lemma 1. *Let $\mathscr{C}$ be a sufficiently refined clustering of the nodes of size m. Let $D^{\mathscr{C}}$ be a $m \times m$ dissimilarity matrix where $D^{\mathscr{C}}(i,j) = \max\{D(u,v) : u \in C_i, v \in C_j\}$. The optimal value ζ^* of the problem $\texttt{pDP}(\texttt{D}^{\mathscr{C}},\texttt{p})$ provides an upper bound of problem $\texttt{pDP}(\texttt{D},\texttt{p})$.*

Proof of Lemma 1. Let $S = \{s_1 \dots s_p\}$ be an optimal solution of problem $\texttt{pDP}(\texttt{D},\texttt{p})$ of value z^*. Because the clustering $\mathscr{C}$ is sufficiently refined, it follows that no two nodes in S can be found in the same cluster $C \in \mathscr{C}$. For every $s \in S$, let $k(s)$ denote the cluster index in $\mathscr{C}$ where node s lies. By construction of $D^{\mathscr{C}}$, we have that $D(s,t) \leq D^{\mathscr{C}}(k(s),k(t))$ for every two nodes $s, t \in S, s < t$, and therefore, $z^* \leq \zeta^*$. □

Our method works as follows. A lower bound $L \leq z^*$ is computed using a simple heuristic (using procedure $\texttt{heuristicPDP}(\texttt{D},\texttt{p})$; see Section 3.2). An initial upper bound U is also computed as simply the largest dissimilarity between any two points in the data set. Using the lower bound L, we build an initial sufficiently refined clustering $\mathscr{C}$ and a reduced dissimilarity matrix $D^{\mathscr{C}}$ (using procedure $\texttt{initialClustering}(\texttt{D},\texttt{p},\texttt{L})$; see Section 3.3). We initially let $S, W \leftarrow \emptyset$, where S represents the set of optimal nonsingleton clusters, and W represents the complete optimal solution to the restricted pDP. In an iterative fashion, we use the sets S, W to refine the current clustering, yielding a refined clustering $\mathscr{C}$ and dissimilarity matrix $D^{\mathscr{C}}$ (using procedure $\texttt{splitAndAdd}(\texttt{S},\texttt{W},\mathscr{C},\texttt{D}^{\mathscr{C}})$; see Section 3.4). The resulting reduced pDP is then solved, yielding an upper bound U, and its optimal solution is used to update the sets S, W (using procedure $\texttt{solvePDP}(\texttt{D}^{\mathscr{C}},\texttt{p})$; see Section 3.5), after which the algorithm iterates. The pseudocode provided in Algorithm 1 formalizes the main steps of our algorithm.

Algorithm 1 (Decremental clustering for $\texttt{pDP}(\texttt{D},\texttt{p})$)

Require: D, p
Ensure: Set $X = \{x_1 \dots x_p\}$ of optimal locations
  $L \leftarrow \texttt{heuristicPDP}(\texttt{D},\texttt{p})$, $U \leftarrow \max\{D(i,j) : 1 \leq i < j \leq n\}$
  $\mathscr{C}, D^{\mathscr{C}} \leftarrow \texttt{initialClustering}(\texttt{D},\texttt{p},\texttt{L})$
  $S \leftarrow \emptyset, W \leftarrow \emptyset$
  repeat
    $\mathscr{C}, D^{\mathscr{C}} \leftarrow \texttt{splitAndAdd}(\texttt{S},\texttt{W},\mathscr{C},\texttt{D}^{\mathscr{C}})$
    $U, W \leftarrow \texttt{solvePDP}(\texttt{D}^{\mathscr{C}},\texttt{p})$
    $S \leftarrow \{w \in W : |C_w| \geq 2\}$
  until $S = \emptyset$
  return $X \leftarrow \{C_w : w \in W\}$

The following proposition formalizes the exactness of the decremental clustering procedure.

Proposition 1. *The decremental clustering method ends in at most n iterations and produces an optimal solution to problem* pDP(D, p).

Proof of Proposition 1. Let $X = \{x_1 \dots x_p\}$ be the optimal solution of problem pDP($\mathtt{D}^{\mathscr{C}}$, p). If the clusters corresponding to the solution X are all singletons, then this is also a feasible solution to problem pDP(D, p) and therefore, produces a lower bound that matches with the upper bound provided by problem pDP($\mathtt{D}^{\mathscr{C}}$, p). Otherwise, the method identifies at least one cluster i such that $|C_i| \geq 2$ and splits it into two separate groups. This can be done at most n times when the clusters in $\mathscr{C}$ become all singletons. □

In Figure 1, we illustrate by means of an example the result of applying the decremental clustering mechanism on instance mu1979.tsp from the TSP Library (TSPLIB) for $p = 5$. On the left side of Figure 1, we plot all of the 1,979 data points of the data set. On the right side of Figure 1, we plot circles representing the different clusters at the last iteration of the method, which are only 37 (note that the circles are only for illustrative purposes, because the clusters themselves are discrete and do not necessarily form circles). This means that the largest reduced pDP solved by our method contained 37 points, and the associated dissimilarity matrix was of dimensions 37×37; this is orders of magnitude smaller than the sizes of the original data structures. The extreme points of the edges appearing on the right in Figure 1 represent the optimal solution of the problem, with the solid red line representing the optimal dissimilarity of 3,845. We would like to highlight the following key observation. Note the right-most point in the optimal solution to the problem surrounded by other points that may be even farther from the other four points in the optimal solution. Therefore, many of those points could be used to replace the one chosen by the algorithm, yielding an equally good value. This is also true for the point in the bottom left corner chosen by the algorithm. It is only reasonable to believe that the large amount of symmetries that the pDP presents is at the core of its intractability. Our algorithm is successful not only at reducing the problem size but also, at reducing the symmetries by a large amount.

Remark 1. When unable to prove optimality, the decremental clustering mechanism can be used to find good-quality lower bounds by solving (exactly or heuristically) a pDP restricted to the points inside the clusters appearing in the last solution found by procedure solvePDP($\mathtt{D}^{\mathscr{C}}$, p). The size of this problem will typically be orders of magnitude smaller than that of the original pDP.

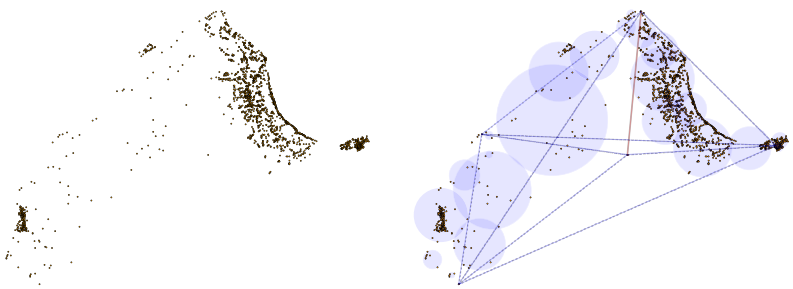
3.2. Procedure heuristicPDP(D, p)

In this section, we describe a simple procedure to compute a nontrivial lower bound L of problem pDP(D, p). This procedure is far from producing a near-optimal solution to the problem but is sufficient to feed the procedure initialClustering(D, p, L) to be described later in Section 3.3. We execute a k-means algorithm using the dissimilarity matrix D to construct p clusters. For each of the p centers in the cluster, we find the node in each cluster that is closest to its center. Let us call this set of points $X = \{x_1 \dots x_p\}$. We compute $d \leftarrow \min\{D(x_i, x_j) : 1 \leq i < j \leq p\}$. This procedure is performed not once but multiple times for as long as the value d keeps increasing. Indeed, we stop after 10 iterations without being able to improve this value. The highest possible such value d is returned as lower bound L.

3.3. Procedure initialClustering(D, p, L)

In this section, we describe a two-step procedure used to build an initial sufficiently refined clustering of the n points using the lower bound L as stopping point. In the first step, a p clustering of the nodes is found using a k-means algorithm with $k = p$ (similar to procedure heuristicPDP(D, p)). This clustering may not be sufficiently

Figure 1. (Color online) Decremental Clustering on Instance mu1979.tsp for $p = 5$



refined, and thus, the second step is executed. This second step is iterative and goes as follows. At any given iteration—say when the number of clusters has reached a value of $m \geq p$—we check if, for every cluster, the maximum dissimilarity between any two nodes is strictly lower than L. If yes, the current clustering $\mathscr{C}$ and dissimilarity matrix $D^{\mathscr{C}}$ are returned. Otherwise, we compute $i^* \leftarrow \arg\max\{D^{\mathscr{C}}(i,i) : i = 1 \dots m\}$ and execute a k-means algorithm with $k = 2$ to further divide cluster C_{i^*} into two clusters. The dissimilarity matrix $D^{\mathscr{C}}$ is then extended to dimensions $(m+1) \times (m+1)$. At this point, only the new rows and columns need to be recomputed to alleviate the computational effort.

3.4. Procedure `splitAndAdd(S,W,`$\mathscr{C}$`,D`$^{\mathscr{C}}$`)`

In this section, we describe a procedure that, given a clustering $\mathscr{C}$, a dissimilarity matrix $D^{\mathscr{C}}$, a family S of cluster indices with $|C_i| \geq 2$ for every $i \in S$, and a set of optimal cluster locations W (with $S \subseteq W$), selects one cluster from those indexed in S and splits it into two separate clusters.. The extended clustering and dissimilarity matrix are returned. By convention, if $S = \emptyset$, the procedure returns $\mathscr{C}$ and $D^{\mathscr{C}}$. This can only happen at the first iteration of the proposed mechanism and assures a correct initialization of the method. We first compute $(s^*, w^*) \leftarrow \arg\min\{D^{\mathscr{C}}(s,w), s \in S, w \in W\}$, which is the pair of indices in $S \times W$ with minimum dissimilarity. This computation excludes on purpose the pairs with both indices in $W \setminus S$, because both associated nodes are—by construction of set S—singletons. If $w^* \in S$, then for the following, the index with highest value of $D^{\mathscr{C}}(u,u)$ is kept, with $u \in \{s^*, w^*\}$. For the retained index, we execute a k-means algorithm with $k = 2$ similar to the one described in the previous section to split the associated cluster into two separate clusters. We update and return the clustering $\mathscr{C}$ and the dissimilarity matrix $D^{\mathscr{C}}$ accordingly.

3.5. Procedure `solvePDP(D`$^{\mathscr{C}}$`,p)`

In this section, we introduce a heuristic and an exact solver for problem `pDP(D`$^{\mathscr{C}}$`,p)`. Without loss of generality and to alleviate the reading, we will drop the superindex $\mathscr{C}$ from the dissimilarity matrix. Therefore, we will simply denote D to refer to it. It goes without saying that we always execute the heuristic solver before any attempt at executing the exact one.

3.5.1. Exact Solver. Our exact solver uses the pure integer formulation introduced by Sayah and Irnich (2017) and solves it by branch and cut embedded within a double-binary search method. This formulation uses m binary variables—one per row/column of the matrix D—to represent the location decisions and Δ binary variables z, where Δ is the number of different values appearing in the matrix D. We refer to Sayah and Irnich (2017) for details of the model and the associated valid inequalities.

Within the decremental clustering scheme, we exploit the existence of a monotonically decreasing upper bound U and exploit this further within a double-binary search scheme as follows. Let us denote by `exactPDP(D,p,L,U)` the solver of problem `pDP(D,p)` when fed with the additional lower and upper bounds L and U. These bounds can be exploited in two aspects: first, to reduce the number of binary variables z and second, to derive cutting planes to strengthen the model. The details of these two accelerating features can be found in full extent in Sayah and Irnich (2017). Our double-binary search method starts with making $l, u \leftarrow U$. It iterates by executing `exactPDP(D,p,l,u)` at every iteration. If no feasible solution exists, the quantities are updated according to the formulas $u \leftarrow l - 1, l \leftarrow l - 2^t$, where t is the iteration number. The problem `exactPDP(D,p,l,u)` is likely to be infeasible for a few iterations. We abort this procedure as soon as one feasible solution is identified, and its objective value is used to update the lower bound. At this point, the final quantities l, u are used to feed another binary search method with the aim of closing the gap between l and u. For as long as $u > l$, we make $r \leftarrow \lceil (l+u)/2 \rceil$ and execute `exactPDP(D,p,r,u)`. If the problem is feasible, we make $l \leftarrow r$; otherwise, we make $u \leftarrow r - 1$ and repeat.

3.5.2. Heuristic Solver. We have observed that, in a large number of iterations, the optimal value of problem `pDP(D,p)` does not decrease from one iteration to the next. This type of dual degeneracy is often observed in decremental relaxation schemes (Aloise and Contardo 2018, Contardo et al. 2019). Therefore, before resorting to executing the exact solver described in the previous section, our heuristic scheme checks if it is possible to select p points out of the $p+1$ points identified from the previous iteration—which includes $p-1$ optimal clusters that remain untouched plus the one that has been split into two—as described in Section 3.4. If the value of this solution equals the upper bound U from the last iteration, the associated solution is then optimal, and there is no need to execute the exact solver.

Table 1. Method Performance on Some Hard OR-Library Instances

Instance	OPT	CPU
pmed29	22	123
pmed30	15	27
pmed33	27	386
pmed34	19	55
pmed37	27	545
pmed40	23	732

Note. OPT, optimal value.

4. Computational Experience

In this section, we provide computational evidence of the effectiveness of the proposed method. Our method has been coded in Julia v1.1 using the JuMP interface v18.5 with Gurobi v8.1 as multipurpose optimization solver. It runs on an Intel Xeon E5-2637 v2 @ 3.50 GHz with 128 GB of RAM. Although this machine is capable of executing code in parallel, for reproducibility purposes, we limit the number of threads to one. We consider data sets coming from two different sources to assess the effectiveness of our method: (i) instances extracted from the OR-Library introduced by Beasley (1990) for the p-median problem and containing small-sized problems with up to 1,000 points and (ii) instances extracted from the TSPLIB data set containing between 1,621 and 104,815 points in the Euclidean plane. In both cases, only integral distances are considered. The OR-Library data set serves for comparison purposes with other methods from the literature, whereas the TSPLIB data set has never been used to assess the performance of p-dispersion algorithms owing to the problems' sizes. We noticed that, in some instances, there exist points with identical coordinates. We proceed to remove all of the redundant entries from an instance before beginning the optimization.

For each instance in the TSPLIB data set, we consider four values of p, namely $p \in \{5, 10, 15, 20\}$. In addition to the algorithm described in this paper, we have also implemented a variant of Sayah and Irnich (2017)'s algorithm embedded within the same double-binary search method described in Section 3.5. Using the notation described in our paper, this method resorts to executing procedure `solvePDP(D,p)` at once. We have executed both algorithms and given them a maximum CPU time of 86,400 seconds (one day). Our implementation of the method of Sayah and Irnich (2017) could not handle problems containing 3,000 nodes or more (it rapidly ran out of memory), and therefore, the comparison between both methods is restricted to the smaller ones.

In Table 1, we report the performance of our algorithm on some difficult instances from the OR-Library data set. Namely, we consider the only six instances that the method of Sayah and Irnich (2017) could not solve within a maximum CPU time of 30 minutes. Five of those instances remain open, because they are reportedly not solved by earlier methods either. In Table 1, we report the optimal value (in the column labeled OPT) and the total CPU time elapsed in seconds (in the column labeled CPU). Our method was able to prove optimality in all six of them.

In Table 2, we report a comparison between our method and our implementation of the method of Sayah and Irnich (2017) restricted to the problems of the TSPLIB data set containing strictly less than 3,000 nodes. We

Table 2. Method Comparison on Small Instances

	Sayah and Irnich (2017)								This paper							
	$p = 5$		$p = 10$		$p = 15$		$p = 20$		$p = 5$		$p = 10$		$p = 15$		$p = 20$	
Instance	UB	CPU	UB	CPU	UB	CPU	UB	CPU	UB	CPU	UB	CPU	UB	CPU	UB	CPU
rw1621.tsp	**971**	206.8	**558**	163.2	**407**	474.9	**339**	582.8	**971**	16.9	**558**	18.6	**407**	21.5	**339**	33.1
u1817.tsp	**1,535**	690.6	**881**	1,897.4	**665**	7,046.3	1,077	TL	**1,535**	31.8	**881**	56.2	**665**	441.8	**559**	1,608.8
rl1889.tsp	**10,166**	4,475.0	**5,846**	3,630.3	**4,478**	72,237.8	4,706	TL	**10,166**	36.2	**5,846**	69.9	**4,478**	258.9	**3,727**	296.9
mu1979.tsp	**3,845**	1,327.9	**2,159**	1,100.2	**1,562**	1,781.7	**1,229**	2,085.2	**3,845**	30.0	**2,159**	30.4	**1,562**	35.1	**1,229**	38.2
pr2392.tsp	**8,086**	9,830.1	**4,976**	18,112.0	**3,788**	73,647.6	3,173	TL	**8,086**	45.0	**4,976**	156.2	**3,788**	535.2	**3,150**	7,489.6
d15112-modif-2500.tsp	**12,217**	24,402.8	**7,132**	22,290.7	**5,771**	12,580.9	4,776	TL	**12,217**	49.1	**7,132**	134.4	**5,771**	191.4	**4,773**	733.2

Notes. Bold indicates the upper bounds (UBs) that match a proven optimal value. TL, time limit.

report, for each method and for each value of p, the final upper bounds (in the column labeled UB in Table 2) and the elapsed CPU times in seconds (in the column labeled CPU in Table 2). We highlight in bold characters in Table 2 the upper bounds that match a proven optimal value. As the results show, our method is more robust, and it is capable of solving to proven optimality all of the problems in this restricted testbed, something that our implementation of the method of Sayah and Irnich (2017) did not. For the problems solved by both methods, ours is always substantially faster.

In Tables 3–6, we report the results obtained by our method for the problems of the TSPLIB data set containing 3,000 or more nodes. We report, in a separate table for each value of p, the final lower and upper bounds (in the columns labeled LB and UB, respectively, in Tables 3–6), the CPU time in seconds (in the column labeled CPU in Tables 3–6), the total number of iterations of the method (in the column labeled ITS in Tables 3–6), the total number of calls to the binary search method (in the column labeled BS in Tables 3–6), the total number of executions of the mixed-integer program (MIP) solver required to solve problem `exactPDP(D,p,L,U)` (in the column labeled MIP in Tables 3–6), the final number of clusters at the final iteration (in the column

Table 3. Decremental Clustering on Large Instances for $p = 5$

Instance	LB	UB	CPU	ITS	BS	MIP	C	X
pcb3038.tsp	**2,390**	**2,390**	57.6	43	14	43	54	1.0
nu3496.tsp	**2,462**	**2,462**	37.1	43	5	43	50	1.0
ca4663.tsp	**34,256**	**34,256**	87.3	36	7	36	47	1.0
rl5915.tsp	**9,793**	**9,793**	199.9	94	25	94	102	1.0
rl5934.tsp	**10,396**	**10,396**	178.9	78	27	78	89	1.0
tz6117.tsp	**6,116**	**6,116**	237.6	81	29	81	94	1.0
eg7146.tsp	**5,247**	**5,247**	241.9	50	13	50	61	1.0
pla7397.tsp	**374,026**	**374,026**	347.3	89	27	89	97	1.0
ym7663.tsp	**4,974**	**4,974**	242.4	70	11	70	78	1.0
pm8079.tsp	**2,078**	**2,078**	84.6	34	11	34	41	1.0
ei8246.tsp	**2,426**	**2,426**	372.5	135	42	135	141	1.0
ar9152.tsp	**13,820**	**13,820**	190.5	47	10	47	54	1.0
ja9847.tsp	**10,651**	**10,651**	439.8	66	23	66	73	1.0
gr9882.tsp	**4,295**	**4,295**	536.3	112	30	112	118	1.0
kz9976.tsp	**13,969**	**13,969**	528.6	100	21	100	106	1.0
fi10639.tsp	**6,284**	**6,284**	485.8	73	27	73	82	1.0
rl11849.tsp	**10,736**	**10,736**	671.3	126	26	126	136	1.0
usa13509.tsp	**229,767**	**229,767**	845.0	54	7	54	63	1.0
brd14051.tsp	**4,379**	**4,379**	820.6	68	12	68	76	1.0
mo14185.tsp	**4,748**	**4,748**	890.4	50	11	50	58	1.0
ho14473.tsp	**2,357**	**2,357**	243.1	56	7	56	64	1.0
d15112.tsp	**12,348**	**12,348**	1,428.1	154	53	154	163	1.0
it16862.tsp	**5,855**	**5,855**	1,289.9	75	19	75	81	1.0
d18512.tsp	**4,396**	**4,396**	2,067.9	197	47	197	209	1.0
vm22775.tsp	**5,348**	**5,348**	2,075.3	63	9	63	69	1.0
sw24978.tsp	**7,128**	**7,128**	2,370.4	59	9	59	71	1.0
fyg28534.tsp	**565**	**565**	3,865.9	90	21	90	96	1.0
bm33708.tsp	**7,094**	**7,094**	4,732.9	87	32	87	91	1.0
pla33810.tsp	**417,437**	**417,437**	7,118.5	154	40	154	162	1.0
bby34656.tsp	**623**	**623**	5,888.5	97	23	97	104	1.0
pba38478.tsp	**698**	**698**	7,100.7	75	16	75	87	1.0
ch71009.tsp	**22,263**	**22,263**	15,256.4	84	24	84	91	1.0
pla85900.tsp	**553,829**	**553,829**	41,421.4	142	44	142	151	1.0
sra104815.tsp	**1,066**	**1,066**	71,588.8	232	56	232	241	1.0
Optimal				34/34				
Average (solved)			5,116	89	23	178	97	
Average (unsolved)								—

Notes. The final lower and upper bounds are in the columns labeled LB and UB, respectively. The CPU times in seconds are in the column labeled CPU. The total numbers of iterations of the method are in the column labeled ITS. The total numbers of calls to the binary search method are in the column labeled BS. The total numbers of executions of the MIP solver required to solve problem `exactPDP(D,p,L,U)` are in the column labeled MIP. The final numbers of clusters at the final iteration are in the column labeled C, and the average numbers of nodes in each cluster appearing in the optimal solution of the last successful call to procedure `solvePDP(`$\mathtt{D}^{\mathscr{C}}$`,p)` are in the column labeled X. We mark in bold whenever a problem is solved to proven optimality. In the last three rows, we report the total number of problems solved to proven optimality as well as averages for the reported data. The averages are computed restricted to the problems solved to proven optimality, except for the average of column X that makes sense only for the unsolved problems.

Table 4. Decremental Clustering on Large Instances for $p = 10$

Instance	LB	UB	CPU	ITS	BS	MIP	C	X
pcb3038.tsp	**1,414**	**1,414**	403.2	448	208	448	471	1.0
nu3496.tsp	**1,524**	**1,524**	46.2	128	43	128	147	1.0
ca4663.tsp	**20,267**	**20,267**	112.7	110	49	110	127	1.0
rl5915.tsp	**6,160**	**6,160**	307.2	285	125	285	310	1.0
rl5934.tsp	**5,951**	**5,951**	614.7	397	181	397	431	1.0
tz6117.tsp	**3,818**	**3,818**	313.3	292	122	292	316	1.0
eg7146.tsp	**3,187**	**3,187**	202.2	89	23	89	112	1.0
pla7397.tsp	**238,412**	**238,412**	460.9	245	70	245	269	1.0
ym7663.tsp	**2,743**	**2,743**	276.9	129	36	129	146	1.0
pm8079.tsp	**1,347**	**1,347**	125.7	118	34	118	135	1.0
ei8246.tsp	**1,500**	**1,500**	444.2	303	108	303	328	1.0
ar9152.tsp	**8,117**	**8,117**	260.2	196	62	196	223	1.0
ja9847.tsp	**5,405**	**5,405**	439.6	126	31	126	137	1.0
gr9882.tsp	**2,633**	**2,633**	715.5	328	103	328	346	1.0
kz9976.tsp	**8,607**	**8,607**	699.9	284	115	284	313	1.0
fi10639.tsp	**3,767**	**3,767**	714.9	310	108	310	332	1.0
rl11849.tsp	**6,243**	**6,243**	1,344.3	436	172	436	470	1.0
usa13509.tsp	**133,500**	**133,500**	1,479.6	363	143	363	390	1.0
brd14051.tsp	**2,465**	**2,465**	1,433.7	432	175	432	446	1.0
mo14185.tsp	**2,803**	**2,803**	1,182.0	247	104	247	267	1.0
ho14473.tsp	**1,427**	**1,427**	388.2	267	123	267	292	1.0
d15112.tsp	**7,319**	**7,319**	4,423.0	682	315	682	707	1.0
it16862.tsp	**3,407**	**3,407**	1,379.0	170	45	170	188	1.0
d18512.tsp	**2,599**	**2,599**	6,114.3	798	380	798	825	1.0
vm22775.tsp	**2,789**	**2,789**	2,404.1	162	47	162	180	1.0
sw24978.tsp	**4,196**	**4,196**	4,175.5	371	149	371	401	1.0
fyg28534.tsp	**340**	**340**	72,442.0	1,292	605	1,292	1,323	1.0
bm33708.tsp	**3,867**	**3,867**	6,339.8	261	83	261	281	1.0
pla33810.tsp	**262,557**	**262,557**	48,718.0	831	409	831	863	1.0
bby34656.tsp	**377**	**377**	15,946.9	1,049	470	1,049	1,086	1.0
pba38478.tsp	**407**	**407**	12,191.9	694	300	694	727	1.0
ch71009.tsp	**14,353**	**14,353**	32,936.8	580	227	580	608	1.0
pla85900.tsp	268,347	349,188	TL	768	391	768	801	139.8
sra104815.tsp	**669**	**669**	84,254.1	656	290	656	690	1.0
Optimal				33/34				
Average (solved)			9,191	396	165	509	421	
Average (unsolved)								139.8

Notes. The final lower and upper bounds are in the columns labeled LB and UB, respectively. The CPU times in seconds are in the column labeled CPU. The total numbers of iterations of the method are in the column labeled ITS. The total numbers of calls to the binary search method are in the column labeled BS. The total numbers of executions of the MIP solver required to solve problem `exactPDP(D,p,L,U)` are in the column labeled MIP. The final numbers of clusters at the final iteration are in the column labeled C, and the average numbers of nodes in each cluster appearing in the optimal solution of the last successful call to procedure `solvePDP(D`$^{\mathscr{C}}$`,p)` are in the column labeled X. We mark in bold whenever a problem is solved to proven optimality. In the last three rows, we report the total number of problems solved to proven optimality as well as averages for the reported data. The averages are computed restricted to the problems solved to proven optimality, except for the average of column X that makes sense only for the unsolved problems. TL, time limit.

labeled C in Tables 3–6), and the average number of nodes in each cluster appearing in the optimal solution of the last successful call to procedure `solvePDP(D`$^{\mathscr{C}}$`,p)` (in the column labeled X in Tables 3–6). The lower bound reported in the column labeled LB in Tables 3–6 is computed following the observations raised in Remark 1. When a problem is not solved to proven optimality, the optimal clusters associated with the last successful execution of procedure `solvePDP(D`$^{\mathscr{C}}$`,p)` are considered, and one point in each such cluster is selected randomly. We perform several random picks while the associated dispersion keeps improving. Once again, we mark in bold characters in Tables 3–6 whenever a problem is solved to proven optimality. In the last three rows in Tables 3–6, we report the total number of problems solved to proven optimality as well as averages for the reported data. The averages are computed restricted to the problems solved to proven optimality, except for the average of column X in Tables 3–6 that makes sense only for the unsolved problems.

As the results show, our method is robust for solving pDPs for small values of p. Only 1 in 68 problems could not be solved within the time limit of one day for $p \leq 10$. For larger values of p, the method is less robust but still capable of handling some very large problems. This behavior is not new, and it has already been

Table 5. Decremental Clustering on Large Instances for $p = 15$

Instance	LB	UB	CPU	ITS	BS	MIP	C	X
pcb3038.tsp	**1,075**	**1,075**	4,751.6	783	383	783	823	1.0
nu3496.tsp	**1,092**	**1,092**	72.7	274	121	274	311	1.0
ca4663.tsp	**15,467**	**15,467**	124.5	191	52	191	220	1.0
rl5915.tsp	**4,544**	**4,544**	16,074.0	994	539	994	1,031	1.0
rl5934.tsp	**4,576**	**4,576**	3,379.5	697	339	697	731	1.0
tz6117.tsp	**2,887**	**2,887**	2,129.6	677	304	677	712	1.0
eg7146.tsp	**2,377**	**2,377**	189.4	120	27	120	151	1.0
pla7397.tsp	**183,522**	**183,522**	736.2	441	148	441	473	1.0
ym7663.tsp	**1,987**	**1,987**	375.1	273	90	273	303	1.0
pm8079.tsp	**941**	**941**	175.0	253	98	253	289	1.0
ei8246.tsp	**1,113**	**1,113**	4,202.0	856	410	856	904	1.0
ar9152.tsp	**6,371**	**6,371**	387.5	346	140	346	382	1.0
ja9847.tsp	**3,907**	**3,907**	520.3	210	50	210	233	1.0
gr9882.tsp	**1,969**	**1,969**	765.5	483	181	483	523	1.0
kz9976.tsp	**6,360**	**6,360**	764.8	381	155	381	426	1.0
fi10639.tsp	**2,806**	**2,806**	3,049.2	690	357	690	722	1.0
rl11849.tsp	**4,719**	**4,719**	9,661.4	932	489	932	965	1.0
usa13509.tsp	**99,689**	**99,689**	9,177.5	700	362	700	735	1.0
brd14051.tsp	**1,862**	**1,862**	1,979.7	558	254	558	603	1.0
mo14185.tsp	**2,125**	**2,125**	1,596.1	502	222	502	540	1.0
ho14473.tsp	**1,104**	**1,104**	466.5	420	188	420	457	1.0
d15112.tsp	**5,907**	**5,907**	9,964.9	888	464	888	931	1.0
it16862.tsp	**2,468**	**2,468**	1,786.9	405	98	405	442	1.0
d18512.tsp	**2,109**	**2,109**	13,913.0	1,088	554	1,088	1,130	1.0
vm22775.tsp	**2,237**	**2,237**	2,626.8	331	72	331	365	1.0
sw24978.tsp	**3,149**	**3,149**	8,416.1	884	386	884	928	1.0
fyg28534.tsp	**276**	**276**	13,926.6	1,044	500	1,044	1,103	1.0
bm33708.tsp	**2,876**	**2,876**	16,545.6	1,015	452	1,015	1,053	1.0
pla33810.tsp	164,269	209,038	TL	1,007	530	1,007	1,069	42.87
bby34656.tsp	**299**	**299**	24,939.4	1,221	615	1,221	1,268	1.0
pba38478.tsp	**311**	**311**	30,705.1	1,324	653	1,324	1,368	1.0
ch71009.tsp	**10,845**	**10,845**	53,997.2	1,094	518	1,094	1,134	1.0
pla85900.tsp	233,189	280,536	TL	678	383	678	736	82.67
sra104815.tsp	242	522	TL	891	446	891	961	194.8
Optimal				31/34				
Average (solved)			7,658	648	297	675	686	
Average (unsolved)								106.8

Notes. The final lower and upper bounds are in the columns labeled LB and UB, respectively. The CPU times in seconds are in the column labeled CPU. The total numbers of iterations of the method are in the column labeled ITS. The total numbers of calls to the binary search method are in the column labeled BS. The total numbers of executions of the MIP solver required to solve problem `exactPDP(D,p,L,U)` are in the column labeled MIP. The final numbers of clusters at the final iteration are in the column labeled C, and the average numbers of nodes in each cluster appearing in the optimal solution of the last successful call to procedure `solvePDP(`$\mathtt{D}^{\mathscr{C}}$`,p)` are in the column labeled X. We mark in bold whenever a problem is solved to proven optimality. In the last three rows, we report the total number of problems solved to proven optimality as well as averages for the reported data. The averages are computed restricted to the problems solved to proven optimality, except for the average of column X that makes sense only for the unsolved problems. TL, time limit.

observed and reported for other relaxation-based methods for minimax and maximin combinatorial optimization problems (Aloise and Contardo 2018, Contardo et al. 2019); it seems to be related to the dual degeneracy occurring when a larger number of clusters can be rearranged from one iteration to the next to find solutions of the same cost. This type of degeneracy occurs at a much smaller scale when the target number of points p is small. It remains an open question how to mitigate the effect of this dual degeneracy when p goes large scale. The averages reported across Tables 3–6 also show that, as the value of p increases, the number of iterations and the number of MIPs required to solve a problem to proven optimality increase, which impacts negatively on the performance of our method and makes it less robust to scale with p. Once again, this behavior is not totally new and has already been observed for other relaxation-based methods for related problems (see, for instance, Aloise and Contardo 2018 and Contardo et al. 2019). In addition, the quality of the lower bound when optimality is not proven is extremely sensitive to the sizes of the final optimal clusters (as reported under column X in Tables 3–6). We would like to remark that the largest instance considered in this study, namely problem sra104815.tsp, would require more than 40 GB of RAM if the full dissimilarity matrix

Table 6. Decremental Clustering on Large Instances for $p = 20$

Instance	LB	UB	CPU	ITS	BS	MIP	C	X
pcb3038.tsp	**898**	**898**	20,582.2	1,056	573	1,056	1,123	1.0
nu3496.tsp	**926**	**926**	71.4	280	133	280	325	1.0
ca4663.tsp	**12,376**	**12,376**	159.6	297	98	297	343	1.0
rl5915.tsp	**3,887**	**3,887**	20,738.6	1,002	572	1,002	1,051	1.0
rl5934.tsp	**3,817**	**3,817**	26,819.2	1,092	581	1,092	1,151	1.0
tz6117.tsp	**2,401**	**2,401**	6,850.7	998	440	998	1,056	1.0
eg7146.tsp	**1,833**	**1,833**	279.6	195	58	195	240	1.0
pla7397.tsp	**148,000**	**148,000**	1,294.7	633	217	633	685	1.0
ym7663.tsp	**1,578**	**1,578**	1,486.7	629	265	629	667	1.0
pm8079.tsp	**805**	**805**	250.0	491	181	491	532	1.0
ei8246.tsp	**939**	**939**	13,253.2	1,071	583	1,071	1,138	1.0
ar9152.tsp	**5,019**	**5,019**	6,108.3	891	420	891	945	1.0
ja9847.tsp	**3,055**	**3,055**	596.9	440	101	440	478	1.0
gr9882.tsp	**1,625**	**1,625**	1,031.6	611	239	611	657	1.0
kz9976.tsp	**5,230**	**5,230**	7,431.4	961	434	961	1,015	1.0
fi10639.tsp	**2,322**	**2,322**	17,607.8	1,046	589	1,046	1,103	1.0
rl11849.tsp	3,538	4,005	TL	1,356	697	1,356	1,394	9.15
usa13509.tsp	79,495	83,894	TL	1,233	630	1,233	1,295	2.15
brd14051.tsp	**1,569**	**1,569**	6,674.7	975	448	975	1,031	1.0
mo14185.tsp	**1,746**	**1,746**	9,122.8	1,029	489	1,029	1,086	1.0
ho14473.tsp	**914**	**914**	6,317.5	880	472	880	937	1.0
d15112.tsp	4,494	4,944	TL	1,278	709	1,278	1,338	6.1
it16862.tsp	**2,100**	**2,100**	1,921.2	504	192	504	555	1.0
d18512.tsp	1,624	1,769	TL	1,336	738	1,336	1,402	7.35
vm22775.tsp	**1,817**	**1,817**	3,166.9	613	199	613	656	1.0
sw24978.tsp	**2,681**	**2,681**	23,601.8	1,285	576	1,285	1,343	1.0
fyg28534.tsp	209	230	TL	1,508	869	1,508	1,564	13.75
bm33708.tsp	2,247	2,394	TL	1,366	654	1,366	1,428	10.65
pla33810.tsp	142,534	178,914	TL	796	513	796	868	40.45
bby34656.tsp	226	250	TL	1,486	829	1,486	1,564	16.6
pba38478.tsp	234	267	TL	1,567	811	1,567	1,605	19.2
ch71009.tsp	**9,311**	**9,311**	56,126.0	1,409	597	1,409	1,462	1.0
pla85900.tsp	177,323	241,559	TL	619	376	619	686	87.2
sra104815.tsp	347	437	TL	858	434	858	963	311.55
Optimal				23/34				
Average (solved)			10,065	799	368	739	851	
Average (unsolved)								47.6

Notes. The final lower and upper bounds are in the columns labeled LB and UB, respectively. The CPU times in seconds are in the column labeled CPU. The total numbers of iterations of the method are in the column labeled ITS. The total numbers of calls to the binary search method are in the column labeled BS. The total numbers of executions of the MIP solver required to solve problem `exactPDP(D,p,L,U)` are in the column labeled MIP. The final numbers of clusters at the final iteration are in the column labeled C, and the average numbers of nodes in each cluster appearing in the optimal solution of the last successful call to procedure `solvePDP(`$\mathtt{D}^{\mathscr{C}}$`,p)` are in the column labeled X. We mark in bold whenever a problem is solved to proven optimality. In the last three rows, we report the total number of problems solved to proven optimality as well as averages for the reported data. The averages are computed restricted to the problems solved to proven optimality, except for the average of column X that makes sense only for the unsolved problems. TL, time limit.

had to be stored in RAM, let alone to load and solve the associated integer program required to execute the method of Sayah and Irnich (2017). Our method avoids this storage and never required more than 2 GB to run even for the largest problems.

5. Concluding Remarks

We have introduced a decremental clustering method for the solution of the pDP. Our method works by building an initial clustering of the nodes and refining this clustering in an iterative fashion. In each iteration, a restricted pDP is solved to compute upper bounds of the problem. In practice, for small values of p, we are capable of proving optimality within a few iterations for problems containing up to 100,000 nodes; this is orders of magnitude larger than the limits of previous methods. To the best of our knowledge, this is the first time that a clustering technique is embedded within an exact solver for location analysis. As an avenue of further research, we believe that the algorithm could be adapted to solve variants of the pDP or other location problems with a potential to benefit from clustering techniques. Although this potential is well understood in

the scientific literature on nonsupervised learning and heuristics for combinatorial optimization, their use within exact methods is rather new, and its full potential has yet to be understood in more depth.

Acknowledgments

The author thanks the two anonymous reviewers whose helpful comments and suggestions greatly helped to improve the quality of this manuscript.

References

Aloise D, Contardo C (2018) A sampling-based exact algorithm for the solution of the minimax diameter clustering problem. *J. Global Optim.* 17(3):613–630.

Beasley JE (1983) Route first–cluster second methods for vehicle routing. *Omega* 11(4):403–408.

Beasley JE (1990) OR-library: Distributing test problems by electronic mail. *J. Oper. Res. Soc.* 41(11):1069–1072.

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

Bräysy O, Gendreau M (2005) Vehicle routing problem with time windows, Part I. Route construction and local search algorithms. *Transportation Sci.* 39(1):104–118.

Chandrasekaran R, Daughety A (1981) Location on tree networks: *P*-centre and *n*-dispersion problems. *Math. Oper. Res.* 6(1):50–57.

Chen D, Chen R (2009) New relaxation-based algorithms for the optimal solution of the continuous and discrete *p*-center problems. *Comput. Oper. Res.* 36(5):1646–1655.

Contardo C, Iori M, Kramer R (2019) A scalable exact algorithm for the vertex *p*-center problem. *Comput. Oper. Res.* 103(2019):211–220.

Delattre M, Hansen P (1980) Bicriterion cluster analysis. *IEEE Trans. Pattern Anal. Machine Intelligence* 2(4):277–291.

Erkut E (1990) The discrete *p*-dispersion problem. *Eur. J. Oper. Res.* 46(1):48–60.

Kuby MJ (1987) Programming models for facility dispersion: The *p*-dispersion and maxisum dispersion problems. *Geographical Anal.* 19(4): 315–329.

Laporte G, Nickel S, da Gama FS, eds. (2015) *Location Science*, vol. 528 (Springer, Cham, Switzerland).

Melo MT, Nickel S, Saldanha-Da-Gama F (2009) Facility location and supply chain management—a review. *Eur. J. Oper. Res.* 196(2):401–412.

Pisinger D (2006) Upper bounds and exact algorithms for *p*-dispersion problems. *Comput. Oper. Res.* 33(5):1380–1398.

Prins C, Lacomme P, Prodhon C (2014) Order-first split-second methods for vehicle routing problems: A review. *Transportation Res. Part C: Emerging Tech.* 40(2014):179–200.

Saboonchi B, Hansen P, Perron S (2014) Maxminmin *p*-dispersion problem: A variable neighborhood search approach. *Comput. Oper. Res.* 52(Part B): 251–259.

Sayah D, Irnich S (2017) A new compact formulation for the discrete *p*-dispersion problem. *Eur. J. Oper. Res.* 256(1):62–67.

Solomon MM (1987) Algorithms for the vehicle routing and scheduling problems with time window constraints. *Oper. Res.* 35(2):254–265.

Statman M (1987) How many stocks make a diversified portfolio? *J. Financial Quant. Anal.* 22(3):353–363.