Submitted to *INFORMS Journal on Optimization*
manuscript MS-0001-1922.65

Decremental clustering for the solution of p-dispersion problems to proven optimality

Claudio Contardo
Department of management and technology, ESG UQAM, GERAD and CIRRELT
E-mail address: claudio.contardo@gerad.ca

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 in 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.

Key words: decremental clustering, p-dispersion problem, exact algorithm, k-means.
History: Submitted on July 3, 2019.

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 pDP(D, p).

The pDP arises in a number of practical contexts. In location analysis, a pDP can help decide the placement of installations whose proximity may be hazardous —as is the case of power plants, oil storage tanks or ammunition—, or in 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 at 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 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. While 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) mention the importance of locating facilites are far as possible from each other when they represent a potential hazard for the surrounding communities. The same authors also mention 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 of most importance in Finance (Statman 1987).

The pDP is tightly related to facility location problems (FLP, Laporte et al. (2015)). In its simplest version, a FLP corresponds to the problem of, given a set of potential facility locations and a set of customers, select a subset of potential facility locations and allocate 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 (Laporte et al. 2015, Melo et al. 2009). The pDP differs from a typical FLP model in the importance of the notion of customer. While 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 fo the pDP. In the OpMP, we are given a set of potential facilities and of 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 to 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) introduces 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), Pisinger (2006) for the pDP. For this to be beneficial, the models need to exploit the availability of lower and upper bounds to fathom non-promising 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 1M 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 intra-cluster 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 600k 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 5 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, and denoted by $\mathcal{C}$, 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) $\bigcup\{C_i : i = 1 \dots m\} = \{1 \dots n\}$. A clustering $\mathcal{C}$ is said to be *sufficiently refined* if, for every set $C_i \in \mathcal{C}$, $D(C_i) := \max\{D(u,v) : u, v \in C_i, u < v\} < z^*$, 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 $\mathcal{C}$ be a sufficiently refined clustering of the nodes of size m. Let $D^{\mathcal{C}}$ be a $m \times m$ dissimilarity matrix where $D^{\mathcal{C}}(i,j) = \max\{D(u,v) : u \in C_i, v \in C_j\}$. The optimal value ζ^* of the problem $\texttt{pDP}(\texttt{D}^{\mathcal{C}},\texttt{p})$ 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 $\mathcal{C}$ is sufficiently refined, it follows that no two nodes in S can be found in the same cluster $C \in \mathcal{C}$. For every $s \in S$, let $k(s)$ denote the cluster index in $\mathcal{C}$ where node s lies. By construction of $D^{\mathcal{C}}$, we have that $D(s,t) \leq D^{\mathcal{C}}(k(s),k(t))$ for every two nodes $s,t \in S, s < t$ and therefore $z^* \leq \zeta^*$. □

Our method works as follows. First, 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 dataset. Using the lower bound L, we build an initial sufficiently refined clustering $\mathcal{C}$ and a reduced dissimilarity matrix $D^{\mathcal{C}}$ (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 non-singleton clusters, and W 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 $\mathcal{C}$ and dissimilarity matrix $D^{\mathcal{C}}$ (using procedure $\texttt{splitAndAdd}(\texttt{S},\texttt{W},\mathcal{C},\texttt{D}^{\mathcal{C}})$, 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 $\texttt{solvePDP}(\texttt{D}^{\mathcal{C}},\texttt{p})$, see Section 3.5), after which the algorithm iterates. The pseudo-code 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

$L \leftarrow \texttt{heuristicPDP}(\texttt{D},\texttt{p})$, $U \leftarrow \max\{D(i,j) : 1 \le i < j \le n\}$

$\mathcal{C}, D^{\mathcal{C}} \leftarrow \texttt{initialClustering}(\texttt{D},\texttt{p},\texttt{L})$

$S \leftarrow \emptyset, W \leftarrow \emptyset$

repeat

$\mathcal{C}, D^{\mathcal{C}} \leftarrow \texttt{splitAndAdd}(\texttt{S},\texttt{W},\mathcal{C},\texttt{D}^{\mathcal{C}})$

$U, W \leftarrow \texttt{solvePDP}(\texttt{D}^{\mathcal{C}},\texttt{p})$

$S \leftarrow \{w \in W : |C_w| \ge 2\}$

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* $\texttt{pDP}(\texttt{D},\texttt{p})$.

Proof of Proposition 1 Let $X = \{x_1 \dots x_p\}$ be the optimal solution of problem $\texttt{pDP}(\texttt{D}^{\mathcal{C}},\texttt{p})$. If the clusters corresponding to the solution X are all singletons, then this is also a feasible solution to problem $\texttt{pDP}(\texttt{D},\texttt{p})$ and therefore produces a lower bound that matches with the upper bound provided by problem $\texttt{pDP}(\texttt{D}^{\mathcal{C}},\texttt{p})$. Otherwise, the method identifies at least one cluster i such that $|C_i| \ge 2$ and splits it into two separate groups. This can be done at most n times when the clusters in $\mathcal{C}$ 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 TSPLIB for $p = 5$. In the left, we plot all the 1,979 data points of the dataset. In the right, we plot circles representing the different clusters at the last iteration of the method, which are only 47 (note that the circles are only for illustrative purposes, as the clusters themselves are discrete and do not necessarily form circles). This means that the largest reduced pDP solved by our method contained 47 points and the associated dissimilarity matrix was of dimensions 47×47, this is orders of magnitude smaller than the sizes of the original data structures. The extreme points of the edges appearing in the right 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 further from the other 4 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 not only successful at reducing the problem size, but also at reducing the symmetries by a large amount.

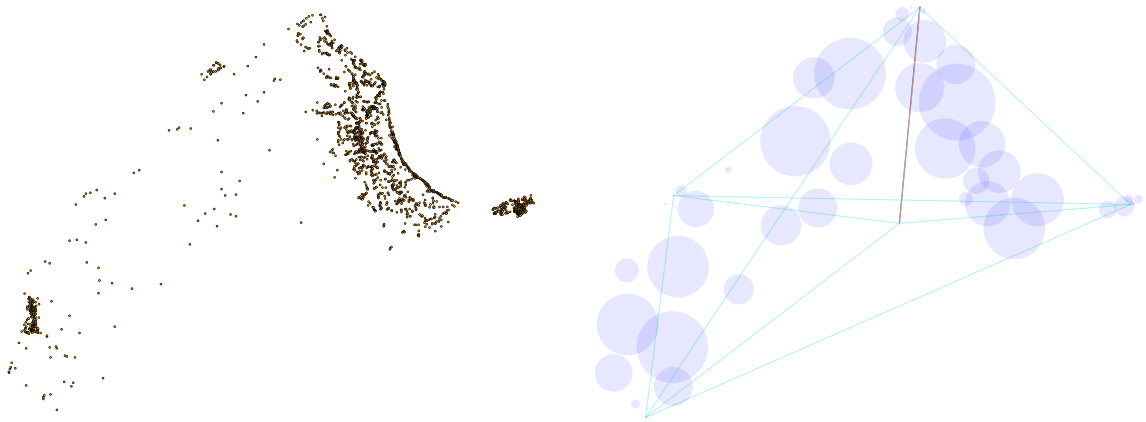

Figure 1 **Decremental clustering on instance** `mu1979.tsp` **for** $p = 5$

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`($\mathtt{D}^{\mathcal{C}}$, `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 non-trivial 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`$(\mathtt{D},\mathtt{p},\mathtt{L})$

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`$(\mathtt{D},\mathtt{p})$). This clustering may not be sufficiently 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 $\mathcal{C}$ and dissimilarity matrix $D^{\mathcal{C}}$ are returned. Otherwise, we compute $i^* \leftarrow \arg\max\{D^{\mathcal{C}}(i,i) : i = 1 \dots m\}$ and execute a k-means algorithm with $k=2$ to further divide cluster C_{i^*} into two clusters. The dissimilarity matrix $D^{\mathcal{C}}$ 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`$(\mathtt{S},\mathtt{W},\mathcal{C},\mathtt{D}^{\mathcal{C}})$

In this section we describe a procedure that, given a clustering $\mathcal{C}$, a dissimilarity matrix $D^{\mathcal{C}}$, 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 $\mathcal{C}$ and $D^{\mathcal{C}}$. This can only happen at the first iteration of the proposed mechanism and assures a correct initialization of the method. We first compute $(s^*,w^*) \leftarrow \arg\min\{D^{\mathcal{C}}(s,w), s\in S, w\in W\}$, which is the pair of indices in $S\times W$ with minimum dissimilarity. This computation excludes on purpose the pairs with both indices in $W\setminus S$ as both associated nodes are —by construction of set S— singletons. If $w^* \in S$, then for the following, the index with highest value of $D^{\mathcal{C}}(u,u)$ 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 $\mathcal{C}$ and the dissimilarity matrix $D^{\mathcal{C}}$ accordingly.

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

In this section we introduce a heuristic and an exact solver for problem `pDP`$(\mathtt{D}^{\mathcal{C}},\mathtt{p})$. Without loss of generality and to alleviate the reading, we will drop the superindex $\mathcal{C}$ 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 $\texttt{exactPDP}(\texttt{D},\texttt{p},\texttt{L},\texttt{U})$ the solver of problem $\texttt{pDP}(\texttt{D},\texttt{p})$ 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. 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 $\texttt{exactPDP}(\texttt{D},\texttt{p},\texttt{l},\texttt{u})$ 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 $\texttt{exactPDP}(\texttt{D},\texttt{p},\texttt{l},\texttt{u})$ 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 $\texttt{exactPDP}(\texttt{D},\texttt{p},\texttt{r},\texttt{u})$. 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 $\texttt{pDP}(\texttt{D},\texttt{p})$ 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.

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 datasets 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; ii) instances extracted from the TSPLIB dataset containing between 1,621 and 104,815

points in the euclidean plane. In both cases, only integral distances are considered. The OR-Library dataset serves for comparison purposes with other methods from the literature, while the TSPLIB dataset has never been used to assess the performance of p-dispersion algorithms due to the problems' sizes. We noticed that in some instances there exist points with identical coordinates. We proceed to remove all the redundant entries from an instance before beginning the optimization.

For each instance in the TSPLIB dataset, 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 (1 day). Our implementation of Sayah and Irnich's method could not handle problems containing 3,000 nodes or more (it rapidly ran out of memory), so 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 dataset. 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 as are reportedly not solved by earlier methods either. In this Table, we report the optimal value (under column labeled OPT) and the total CPU time elapsed in seconds (under column labeled CPU). Our method was able to prove optimality in all six of them.

Instance	OPT	CPU
pmed29	22	154
pmed30	15	27
pmed33	27	395
pmed34	19	100
pmed37	27	586
pmed40	23	1,284

Table 1 Method performance on some hard OR-Library instances

In Table 2 we report a comparison between our method and our implementation of Sayah and Irnich's method, restricted to the problems of the TSPLIB dataset containing strictly less than 3,000 nodes. We report, for each method and for each value of p, the final upper bounds (under column labeled `UB`) and the elapsed CPU times in seconds (under column labeled `CPU`). We highlight in bold characters the upper bounds that match a proven optimal value. As the results show, our method is more robust and is capable of solving to proven optimality all the problems in this restricted testbed, something that our implementation of Sayah and Irnich's method did not. For the problems solved by both methods, ours is always substantially faster.

Instance	Sayah and Irnich (2017)								This paper							
	$p=5$		$p=10$		$p=15$		$p=20$		$p=5$		$p=10$		$p=15$		$p=20$	
	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**	16.1	**558**	17.8	**407**	24.4	**339**	32.2
u1817.tsp	**1,535**	690.6	**881**	1,897.4	**665**	7,046.3	1,077	TL	**1,535**	27.6	**881**	52.6	**665**	371.9	**559**	1,578.9
rl1889.tsp	**10,166**	4,475.0	**5,846**	3,630.3	**4,478**	72,237.8	4,706	TL	**10,166**	28.6	**5,846**	74.8	**4,478**	221.3	**3,727**	400.3
mu1979.tsp	**3,845**	1,327.9	**2,159**	1,100.2	**1,562**	1,781.7	**1,229**	2,085.2	**3,845**	28.2	**2,159**	30.7	**1,562**	33.3	**1,229**	63.4
pr2392.tsp	**8,086**	9,830.1	**4,976**	18,112.0	**3,788**	73,647.6	3,173	TL	**8,086**	38.7	**4,976**	121.9	**3,788**	375.0	**3,150**	6,990.6
d15112-modif-2500.tsp	**12,217**	24,402.8	**7,132**	22,290.7	**5,771**	12,580.9	4,776	TL	**12,217**	46.1	**7,132**	95.9	**5,771**	222.8	**4,773**	583.7

Table 2 Method comparison on small instances

In Table 3 we report the results obtained by our method for the problems of the TSPLIB dataset containing 3,000 or more nodes. We report, for each value of p, the final upper bound (under column labeled `UB`), the CPU time in seconds (under column labeled `CPU`), and the final number of clusters at the final iteration (under column labeled `C`). Once again, we mark in bold characters whenever a problem is solved to proven optimality. In the last two rows we report —for each value of p— the total number of problems solved to proven optimality as well as the average CPU times and the average number of clusters restricted to the problems solved to proven optimality. As the results show, our method is robust for solving `pDP`s for small values of p. Only one 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 has already been observed and reported for other relaxation-based methods for minimax and maximin combinatorial optimization problems (Aloise and Contardo 2018, Contardo et al. 2019), and seems to be related to the dual degeneracy occurring when a larger number of clusters can be re-arranged 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. 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 had to be stored in RAM, let alone to load and solve the associated integer program required to execute Sayah and Irnich's method. 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 p-dispersion problem (`pDP`). Our method works by building an initial clustering of the nodes and by 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

Instance	$p=5$			$p=10$			$p=15$			$p=20$		
	UB	CPU	C	UB	CPU	C	UB	CPU	C	UB	CPU	C
pcb3038.tsp	**2,390**	49.8	65	**1,414**	478.1	452	**1,075**	4,275.8	781	**898**	15,851.1	1050
nu3496.tsp	**2,462**	33.8	59	**1,524**	47.2	161	**1,092**	91.6	334	**926**	105.8	410
ca4663.tsp	**34,256**	97.1	71	**20,267**	142.8	227	**15,467**	165.8	276	**12,376**	208.3	357
rl5915.tsp	**9,793**	166.9	108	**6,160**	217.5	234	**4,544**	14,236.0	1024	**3,887**	16,894.9	987
rl5934.tsp	**10,396**	171.9	110	**5,951**	713.7	454	**4,576**	3,904.0	762	**3,817**	25,333.0	1096
tz6117.tsp	**6,116**	207.8	157	**3,818**	249.7	307	**2,887**	1,100.9	619	**2,401**	3,190.1	828
eg7146.tsp	**5,247**	236.4	83	**3,187**	265.0	172	**2,377**	264.0	213	**1,833**	433.0	390
pla7397.tsp	**374,026**	273.3	78	**238,412**	342.5	229	**183,522**	544.8	420	**148,000**	1,218.7	654
ym7663.tsp	**4,974**	242.0	71	**2,743**	274.1	143	**1,987**	332.8	292	**1,578**	723.3	554
pm8079.tsp	**2,078**	113.3	60	**1,347**	121.0	167	**941**	157.0	309	**805**	168.1	417
ei8246.tsp	**2,426**	308.1	118	**1,500**	361.9	311	**1,113**	3,542.0	863	**939**	12,292.2	1156
ar9152.tsp	**13,820**	173.0	57	**8,117**	239.4	216	**6,371**	429.9	422	**5,019**	5,184.0	833
ja9847.tsp	**10,651**	352.6	69	**5,405**	393.8	153	**3,907**	436.7	198	**3,055**	531.0	441
gr9882.tsp	**4,295**	432.6	113	**2,633**	501.3	274	**1,969**	616.8	443	**1,625**	677.4	525
kz9976.tsp	**13,969**	415.9	94	**8,607**	495.5	285	**6,360**	835.2	479	**5,230**	5,189.4	937
fi10639.tsp	**6,284**	479.3	83	**3,767**	704.6	407	**2,806**	2,588.7	716	**2,322**	17,955.2	1146
rl11849.tsp	**10,736**	556.3	91	**6,243**	1,065.9	477	**4,719**	7,836.8	908	**4,000**	32,583.2	1214
usa13509.tsp	**229,767**	712.3	66	**133,500**	1,082.1	330	**99,689**	7,022.5	726	83,538	TL	1413
brd14051.tsp	**4,379**	694.7	68	**2,465**	1,161.8	358	**1,862**	2,025.2	681	**1,569**	4,317.6	872
mo14185.tsp	**4,748**	784.8	76	**2,803**	949.9	299	**2,125**	1,785.6	641	**1,746**	5,526.7	960
ho14473.tsp	**2,357**	215.5	75	**1,427**	315.0	274	**1,104**	429.0	452	**914**	5,563.1	934
d15112.tsp	**12,348**	1,127.7	168	**7,319**	4,003.8	704	**5,907**	9,635.4	932	**4,944**	84,652.7	1384
it16862.tsp	**5,855**	1,187.3	98	**3,407**	1,246.1	171	**2,468**	1,572.3	425	**2,100**	1,870.6	610
d18512.tsp	**4,396**	1,570.8	164	**2,599**	4,752.5	801	**2,109**	10,771.2	1050	**1,762**	38,044.8	1269
vm22775.tsp	**5,348**	1,708.1	72	**2,789**	2,134.8	177	**2,237**	2,676.0	386	**1,817**	2,934.2	612
sw24978.tsp	**7,128**	2,181.0	81	**4,196**	3,030.5	348	**3,149**	4,869.8	714	**2,681**	20,742.0	1276
fyg28534.tsp	**565**	3,374.2	91	**340**	35,256.4	1314	**276**	12,926.6	1131	230	TL	1556
bm33708.tsp	**7,094**	4,195.4	116	**3,867**	5,662.3	235	**2,876**	16,009.8	1113	**2,390**	23,630.3	1229
pla33810.tsp	**417,437**	5,544.0	164	**262,557**	44,046.0	833	207,885	TL	1017	178,213	TL	860
bby34656.tsp	**623**	4,639.8	92	**377**	10,758.9	861	**299**	23,676.6	1268	251	TL	1488
pba38478.tsp	**698**	6,203.4	94	**407**	9,829.3	669	**311**	38,457.8	1464	266	TL	1596
ch71009.tsp	**22,263**	21,811.4	117	**14,353**	25,522.2	547	**10,845**	38,990.8	1011	**9,311**	38,499.4	1221
pla85900.tsp	**553,829**	32,147.4	161	348,661	TL	843	278,770	TL	807	240,465	TL	705
sra104815.tsp	**1,066**	52,593.4	200	**669**	64,275.4	567	**518**	76,409.9	1113	432	TL	1332
Optimal	34/34			33/34			32/34			27/34		
Average		4,265	100		6,686	399		9,019	693		13,493	865

Table 3 Decremental clustering on large instances

location problems with a potential to benefit from clustering techniques. While this potential is well understood in the scientific literature on non-supervised learning and heuristics for combinatorial optimization, their use within exact methods is rather new and its full potential is yet to be understood in more depth.

Acknowledgments

The author thanks the two anonymous reviewers whose helpful comments and suggestions greatly helped 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. *Journal of Global Optimization* 1–18.

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

Beasley JE (1990) Or-library: Distributing test problems by electronic mail. *The Journal of the Operational Research Society* 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 Science* 39(1):104–118.

Chandrasekaran R, Daughety A (1981) Location on tree networks: P-centre and n-dispersion problems. *Mathematics of Operations Research* 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. *Computers & Operations Research* 36(5):1646–1655.

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

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

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

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

Laporte G, Nickel S, da Gama FS (2015) *Location science*, volume 528 (Springer).

Melo MT, Nickel S, Saldanha-Da-Gama F (2009) Facility location and supply chain management–a review. *European Journal of Operational Research* 196(2):401–412.

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

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

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

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

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

Statman M (1987) How many stocks make a diversified portfolio? *Journal of Financial and Quantitative Analysis* 22(3):353–363, URL `http://dx.doi.org/10.2307/2330969`.