Decremental clustering for the exact solution of some large-scale p-dispersion problems

Claudio Contardo

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

March 23, 2019

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 in selecting exactly p out of the n points in such a way that the minimum dissimilarity between any pair of selected points is maximum. This problem 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. 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.

1 Introduction

In the p-dispersion problem (pDP) we are given a set of n points, a symmetric dissmilarity matrix $D = \{D(i,j) : 1 \leq i,j \leq n\}$ satisfying $D(i,j) > 0$ if $i < j$ and $D(i,i) = 0$ for every $1 \leq i,j \leq n$, 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 reduce the investment 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 $O(n^2)$ variables and constraints. 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.

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's contributions. 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.

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 dissmilarity 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 linearization-reformulation. 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 to allocate the remaining points to its closest center 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 problem can thus be modeled as a smaller VPCP. Contardo et al. (2019) report the solution of problems containing up to 1M observations. The minimax diameter clustering problem (MMDCP) is another problem for which the decremental relaxation mechanism has proven useful. Aloise and Contardo (2018) introduced a sampling mechanism to solve 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. Our technique differs from those mentioned in this paragraph in the fundamental property that

our mechanism is capable of providing solutions with proven 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. 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 dissimlarity between any two points in the dataset. Using the lower bound L, we build an initial sufficiently refined clustering $\mathcal{C}$ and a reduced dissmilarity matrix $D^{\mathcal{C}}$ (using procedure $\texttt{initialClustering}(\texttt{D}, \texttt{p}, \texttt{L})$, see Section 3.3). We initially let $S, X \leftarrow \emptyset$, where S represents the set of clusters of non-zero dissimilarities (this is the case of any cluster containing two nodes or more), and X an optimal solution to the restricted pDP. In an iterative fashion, we use the sets S, X to refine the current clustering, yielding a refined clustering $\mathcal{C}$ and dissimilarity matrix $D^{\mathcal{C}}$ (using procedure $\texttt{splitAndAdd}(\texttt{S}, \texttt{X}, \mathcal{C}, \texttt{D}^{\mathcal{C}})$, 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, X (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
  $L \leftarrow \texttt{heuristicPDP}(\texttt{D}, \texttt{p})$, $U \leftarrow \max\{D(i, j) : 1 \leq i < j \leq n\}$
  $\mathcal{C}, D^{\mathcal{C}} \leftarrow \texttt{initialClustering}(\texttt{D}, \texttt{p}, \texttt{L})$
  $S \leftarrow \emptyset, X \leftarrow \emptyset$
  repeat
    $\mathcal{C}, D^{\mathcal{C}} \leftarrow \texttt{splitAndAdd}(\texttt{S}, \texttt{X}, \mathcal{C}, \texttt{D}^{\mathcal{C}})$
    $U, X \leftarrow \texttt{solvePDP}(\texttt{D}^{\mathcal{C}}, \texttt{p})$
    $S \leftarrow \{x \in X : D^{\mathcal{C}}(x, x) > 0\}$
  until $S = \emptyset$
  return X

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. 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 $D(i, i) > 0$ 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 `rw1621.tsp` from the TSPLIB for $p = 5$. In the left, we plot all the 1,621 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 57. This means that the largest reduced `pDP` solved by our method contained 57 points and the associated dissimilarity matrix was of dimensions 57×57, 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 line representing the optimal dissimilarity of 971.

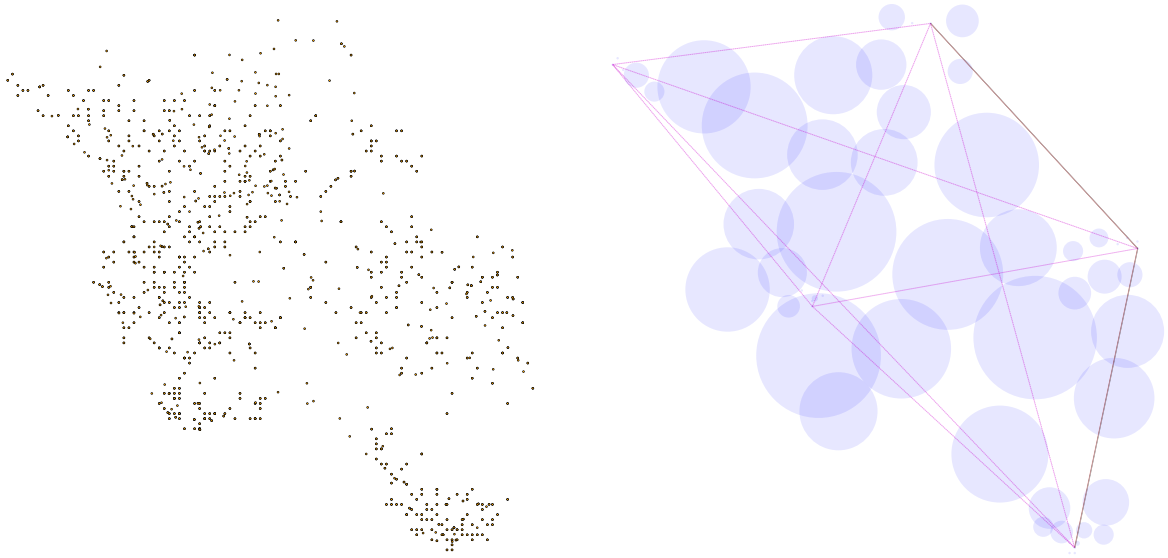

Figure 1: Decremental clustering on instance rw1621.tsp for $p = 5$

3.2 Procedure `heuristicPDP`$(\mathtt{D}, \mathtt{p})$

In this section we describe a simple procedure to compute a non-trivial lower bound L of problem `pDP`$(\mathtt{D}, \mathtt{p})$. This procedure is far from producing a near-optimal solution to the problem, but is sufficient to feed the procedure `initialClustering`$(\mathtt{D}, \mathtt{p}, \mathtt{L})$ to be described later in Section 3.3. We execute a k-means algorithm using the dissimlarity 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. 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—, we check if the sizes of each cluster are strictly lower than L. If yes, the current clustering $\mathcal{C}$ and dissimlarity 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 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`$(\mathtt{S}, \mathtt{X}, \mathcal{C}, \mathtt{D}^{\mathcal{C}})$

In this section we describe a procedure that, given a clustering $\mathcal{C}$, a dissmilarity matrix $D^{\mathcal{C}}$, a family S of cluster indices with $D^{\mathcal{C}}(i, i) > 0$ for every $i \in S$ and a set of optimal cluster locations (with $S \subseteq X$), selects one cluster

from those indexed in S and splits it into two separate clusters. The extended clustering and dissimilarity matrix are returned. We first compute $(s^*, x^*) \leftarrow \arg\min\{D^{\mathcal{C}}(s,x), s \in S, x \in X\}$, which is the pair of indices in $S \times X$ with minimum dissimilarity. This computation excludes on purpose the pairs with both indices in $X \setminus S$ as both associated nodes are —by construction of set S— of zero dissimilarity. If $x^* \in S$, then for the following, the index with highest value of $D^{\mathcal{C}}(u,u)$ is kept, with $u \in \{s^*, x^*\}$. For the retained index, we execute a k-means algorithm 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 $\texttt{solvePDP}(\texttt{D}^{\mathcal{C}}, \texttt{p})$

In this section we introduce a heuristic and an exact solver for problem $\texttt{pDP}(\texttt{D}^{\mathcal{C}}, \texttt{p})$. 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 feeded 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 $l \leftarrow l - 2^t, u \leftarrow l - 1$, where t is the iteration number. When the problem becomes feasible, we abort the optimization 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 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 v8.0 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 instances from the TSPLIB containing between 1,621 and 104,815 points in the euclidean plane. The dissimilarity between two points is computed according to the TSPLIB standard and considers only integral distances.

For each instance in the 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 variation of Sayah and Irnich (2017)'s algorithm embedded within the same binary search method described in Section 3.5. Using the notation described in our paper, this method resorts to executing procedure $\texttt{solvePDP}(\texttt{D}, \texttt{p})$ 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 a comparison between our method and our implementation of Sayah and Irnich's method, restricted to the problems containing strictly less than 3,000 nodes. We report, for each method, 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	**971**	2,010.3	**558**	700.7	**407**	1,030.3	**339**	932.3	**971**	22.5	**558**	26.8	**407**	35.9	**339**	50.4
u1817	**1,535**	1,109.4	**881**	2,384.8	678	TL	1,077	TL	**1,535**	27.3	**881**	52.8	**665**	513.0	**559**	1,168.6
rl1889	**10,166**	4,826.6	**5,846**	22,330.2	4,591	TL	**3,727**	23,249.5	**10,166**	28.4	**5,846**	63.7	**4,478**	183.0	**3,727**	292.1
mu1979	**3,845**	1,712.8	**2,159**	1,101.9	**1,562**	1,907.4	**1,229**	2,068.8	**3,845**	26.0	**2,159**	28.0	**1,562**	34.5	**1,229**	44.9
pr2392	**8,086**	9,225.5	4,977	TL	3,790	TL	3,184	TL	**8,086**	38.1	**4,976**	125.6	**3,788**	478.4	**3,150**	6,577.1
d15112-modif-2500	**12,217**	18,518.2	7,153	TL	**5,771**	16,777.8	4,813	TL	**12,217**	38.7	**7,132**	92.3	**5,771**	257.0	**4,773**	974.5

Table 1: Method comparison on small instances

In Table 2 we report the results obtained by our method for the problems 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. As the results show, our method is robust for solving `pDP` for small values of p. Only one in 68 problems could not be solved within the time limit for $p \leq 10$. For larger p, the method is less robust but still capable of handling large problems. 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 to store the full dissimilarity matrix. Our method, however, avoids this storage and did never require more than a 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 iteratively clustering nodes that are close to each other and solving a restricted `pDP` to compute a non-increasing upper bound. In practice, for small values of p, we are capable of proving optimality within a few iterations. The method is capable of handling and solving `pDP`s containing up to 100,000 nodes within a day. This is orders of magnitude larger than the limits of previous methods. As an avenue of further research, we believe that the algorithm could be adapted to solve variants of the `pDP` or other problems with a potential to benefit from clustering techniques. While he potential of clustering techniques has been widely studied in the scientific literature in non-supervised learning and in heuristics for combinatorial optimization, their use within exact methods is rather new and their full potential is yet to be understood in more depth.

Acknowledgments

The author thanks the Natural Sciences and Engineering Research Council of Canada (NSERC) under Discovery Grant 435824-2013.

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.

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

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

R. Chandrasekaran and A. Daughety. Location on tree networks: P-centre and n-dispersion problems. *Mathematics of Operations Research*, 6(1):50–57, 1981.

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.

Instance	$p = 5$			$p = 10$			$p = 15$			$p = 20$		
	UB	CPU	C	UB	CPU	C	UB	CPU	C	UB	CPU	C
pcb3038	**2,390**	48.2	58	**1,414**	356.8	474	**1,075**	4,002.8	775	**898**	16,815.1	1072
nu3496	**2,462**	58.0	59	**1,524**	72.0	162	**1,092**	114.4	310	**926**	117.0	362
ca4663	**34,256**	87.8	54	**20,267**	115.2	149	**15,467**	153.4	263	**12,376**	165.1	376
rl5915	**9,793**	171.8	105	**6,160**	267.8	302	**4,544**	15,495.5	1058	**3,887**	15,291.5	1055
rl5934	**10,396**	147.4	77	**5,951**	495.8	395	**4,576**	2,703.7	677	**3,817**	20,639.4	1140
tz6117	**6,116**	189.7	104	**3,818**	244.9	277	**2,887**	997.2	591	**2,401**	3,943.3	881
eg7146	**5,247**	230.3	66	**3,187**	277.0	150	**2,377**	282.1	237	**1,833**	312.4	325
pla7397	**374,026**	294.8	106	**238,412**	426.7	274	**183,522**	640.3	398	**148,000**	899.0	685
ym7663	**4,974**	207.2	86	**2,743**	335.8	196	**1,987**	357.6	318	**1,578**	678.1	535
pm8079	**2,078**	248.7	53	**1,347**	305.5	174	**941**	358.4	318	**805**	408.3	482
ei8246	**2,426**	326.7	158	**1,500**	360.5	292	**1,113**	2,597.3	750	**939**	11,699.1	1140
ar9152	**13,820**	339.4	64	**8,117**	392.5	227	**6,371**	553.4	360	**5,019**	8,751.5	934
ja9847	**10,651**	367.4	66	**5,405**	431.3	130	**3,907**	483.4	238	**3,055**	531.9	453
gr9882	**4,295**	447.9	114	**2,633**	522.8	289	**1,969**	655.2	472	**1,625**	758.4	597
kz9976	**13,969**	402.2	75	**8,607**	493.3	239	**6,360**	736.9	479	**5,230**	5,475.8	971
fi10639	**6,284**	396.4	70	**3,767**	589.0	262	**2,806**	3,388.6	752	**2,322**	13,091.2	1041
rl11849	**10,736**	598.4	102	**6,243**	994.9	435	**4,719**	8,749.8	951	**4,000**	44,051.3	1176
brd14051	**4,379**	726.2	73	**2,465**	1,130.0	377	**1,862**	1,547.2	624	**1,569**	4,896.5	857
mo14185	**4,748**	677.5	57	**2,803**	1,007.9	296	**2,125**	1,726.8	645	**1,746**	6,967.1	1063
ho14473	**2,357**	870.9	75	**1,427**	1,157.8	302	**1,104**	1,225.2	435	**914**	3,747.3	763
it16862	**5,855**	1,156.1	86	**3,407**	1,147.7	145	**2,468**	1,364.0	364	**2,100**	2,057.7	714
d18512	**4,396**	1,624.9	172	**2,599**	5,478.7	792	**2,109**	16,337.8	1211	**1,762**	55,518.6	1312
vm22775	**5,348**	1,668.9	73	**2,789**	2,232.3	217	**2,237**	2,419.9	423	**1,817**	2,815.5	599
sw24978	**7,128**	2,556.5	95	**4,196**	3,160.8	341	**3,149**	6,431.2	914	**2,681**	20,827.2	1186
fyg28534	**565**	3,524.3	92	**340**	17,071.7	1209	**276**	10,940.5	1012	230	TL	1538
bm33708	**7,094**	4,350.2	104	**3,867**	5,548.2	290	**2,876**	14,543.6	1080	**2,390**	24,265.0	1231
pla33810	**417,437**	5,460.1	152	**262,557**	42,357.6	864	208,785	TL	998	178,157	TL	829
bby34656	**623**	5,261.0	103	**377**	10,361.3	906	**299**	27,055.6	1287	250	TL	1547
pba38478	**698**	5,607.6	84	**407**	10,580.4	738	**311**	35,803.4	1437	266	TL	1600
ch71009	**22,263**	19,992.8	112	**14,353**	28,012.0	506	**10,845**	35,813.0	1034	**9,311**	43,515.4	1247
pla85900	**553,829**	35,265.8	162	348,796	TL	809	280,109	TL	715	242,069	TL	659
sra104815	**1,066**	52,743.1	211	**669**	65,996.8	614	**518**	76,573.0	1167	432	TL	1333
Optimal	32/32			31/32			30/32			26/32		

Table 2: Decremental clustering on large instances

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

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

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

G. Laporte, S. Nickel, and F. S. da Gama. *Location science*, volume 528. Springer, 2015.

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

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

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

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

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