IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 9, 2020, accepted July 30, 2020, date of publication August 14, 2020, date of current version August 25, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3016724

Social Distancing as p-Dispersion Problem

JAKUB KUDELA
Institute of Automation and Computer Science, Brno University of Technology, 601 90 Brno, Czech Republic
e-mail: jakub.kudela@vutbr.cz

This work was supported in part by the Ministry of Education, Youth and Sports of the Czech Republic (Computer Simulations for Effective Low-Emission Energy) under Project CZ.02.1.01/0.0/0.0/16_026/0008392, and in part by IGA under Grant BUT:FSI-S-20-6538.

ABSTRACT The spread of COVID-19 and similar viruses poses new challenges for our society. There is a strong incentive towards safety measures that help to mitigate the outbreaks. Many countries have imposed social distancing measures that require a minimum distance between people in given places, such as schools, restaurants, shops, etc. This in turn creates complications for these places, as their function is to serve as many people as they were originally designed for. In this article, we pose the problem of using the available space in a given place, such that the social distancing measures are satisfied, as a *p*-dispersion problem. We use recent algorithmic advancements, that were developed for the *p*-dispersion problem, and combine them with discretization schemes to find computationally attainable solutions to the *p*-dispersion problem and investigate the trade-off between the level of discretization and computational efforts on one side, and the value of the optimal solution on the other.

INDEX TERMS Social distancing, p-dispersion problem, decremental clustering, COVID-19.

I. INTRODUCTION

The outbreak of the COVID-19 had an enormous impact on the world at large. To mitigate the spread of the virus, various technologies, such as Internet of Things, Unmanned Aerial Vehicles, blockchain, Artificial Intelligence, and 5G are already in use [1]. In this article, we take a look at the problem of positioning people in a given area, such as in a restaurant, school, office, etc., in order to minimize the spread of viruses such as COVID-19. After the initial lockdown, many countries imposed a set of social distancing measures that should help to slow down the spread of the virus. These measures impose a minimum distance between people in a given area. This means that spaces that could previously serve a large number people need to be adjusted for these new measures. As it seems unlikely that we will see the construction of new places that will be designed to abide by these (hopefully temporary) measures, it is only natural to try to find the best use of the ''facilities'' that are already available. However, as the social distancing measures do not have to be stable and can change over time, we will pose the problem of using the available space to its full extent in the following way: Given a fixed number p of people, fit them into a predefined space in such a way, that the minimum distance between any two persons is maximized. Afterwards, by varying p, we can get the optimal (largest) distance that the people can be separated by, and, given a particular social distancing rule, we can determine the maximum number (and placement) of people that will fit into the predefined space. The problem of selecting p points in order to maximize the minimum distance between any pair is called the *p*-dispersion problem [2] and it is one of the classical combinatorial optimization problems. Although easy to formulate, effective and provably optimal methods for solving this problem are quite a recent development. Most notably, the state-of-the-art methods are based on the formulation developed in 2017 in [3] and the most successful method is the decremental clustering scheme published in 2020 in [4]. It is these advancements that made it possible to solve instances of the size sufficient for our purpose. In this article, we devise a discretization scheme that is build on top of the decremental clustering to find computationally attainable solutions to the *p*-dispersion problem and investigate the trade-off between the level of discretization and computational efforts on one side, and the value of the optimal solution (the minimum distance between any two points) on the other. The investigation is carried out on two numerical examples, the first one is a place with a ''general'' shape, the second one with an ''auditorium-like'' shape.

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Liu.

II. THE p-DISPERSION PROBLEM

A. DEFINITION AND FORMULATIONS

In the *p*-dispersion problem (pDP), we are given a set of n points, a dissimilarity (or distance) matrix

$D = \{D(i,j) : 1 \le i,j \le n\}$ satisfying $D(i,j) \ge 0$ for every $1 \le i,j \le n$ and $D(i,i) = 0$ for every $1 \le i \le n$, and an integer $p \ge 2$. The goal is to choose p points from the set of n points in such a way, that the minimum pairwise dissimilarity (the distance between any two points) within the selected points is maximized. The pDP is an NP hard problem [5]. We denote this problem for given input parameters D and p as $\texttt{pDP}(D,p)$. One of the standard applications of the pDP is the location of nuclear power plants, where one is interested in minimizing the risk of losing multiple plants in the event that only one plant is subjected to an enemy attack. To achieve this, the desired selection of plants is that in which the interplant distances are as large as possible [3]. A more peaceful applications of the pDP can be found in location analysis of services, e.g., schools, hospitals, electoral districts, or waste collection plants. A comprehensive survey of the location applications of pDP can be found in [6] and [7]. Another application of the pDP is found in multiobjective optimization – if the Pareto frontier of a problem contains multiple solutions, one can solve a pDP to find p such solutions with most distinct features [8]. In the same paper, an application in portfolio optimization is presented – given a set of potential investment opportunities, one wishes to choose a subset that reduces the closeness in terms of features between the different investment options, which reduces the risk associated with the portfolio.

Within the methodological contributions to the solution of the pDP, several articles have dealt with the problem of solving the pDP to proven optimality. A mixed-integer quadratic formulation was introduced by [9], which can be partially solved by a series of relaxations and reformulation-linearization. A mixed-integer linear formulation of the problem using the "big M" constraints was defined in [6]. This formulation can be retroactively thought of as a linearization of the mixed integer quadratic model, that was developed 20 years afterward. Although the linear model is more compact than the quadratic one, it provides much weaker upper bounds.

Our attention will focus on a formulation introduced in [3], which is a novel pure binary compact formulation. Using this formulation, the authors reported substantial computational advancements when compared with the other formulations. Without loss of generality, we can assume that the dissimilarity matrix D is symmetric. Let (I,E) be the complete graph in which points $I = \{1,\dots,n\}$ are the vertices and $E = \{(i,j) \in I \times I : i < j\}$ are the edges. Given any distance d, we define subsets of edges as

$$E(d) = \{(i,j) \in E : D(i,j) < d\} \subseteq E.$$

The compact pure binary formulation exploits the fact that the optimal distance is identical to at least one of the entries in the dissimilarity matrix. Let $D^0 < D^1 < \cdots < D^{k_{max}}$ be the different non-zero values in D. The associated index sets are $K = \{1,2,\dots,k_{max}\}$ and $K_0 = \{0\} \cup K$. This formulation uses two types of binary variables: The binary location variable x_i indicates if the point $i \in I$ is selected. For $k \in K$, the binary variable z_k indicates if the location decisions (the particular selection of p points) satisfy a minimum distance of at least D^k. The pure binary program is the following:

$$\max D^0 + \sum_{k \in K} (D^k - D^{k-1}) z_k \tag{1}$$

$$\text{s.t.} \sum_{i \in I} x_i = p \tag{2}$$

$$z_k \le z_{k-1}, \quad k \in K, k > 1 \tag{3}$$

$$x_i + x_j + z_k \le 2, \quad k \in K, (i,j) \in E(D^k) \backslash E(D^{k-1}) \tag{4}$$

$$x_i \in \{0,1\}, \quad i \in I \tag{5}$$

$$z_k \in \{0,1\}, \quad k \in K \tag{6}$$

The formulation (1)-(6) can be further strengthen using clique-like inequalities and computation can be sped-up by exploiting valid lower and upper bounds [3].

B. DECREMENTAL CLUSTERING METHOD

The decremental clustering method introduced in [4] for the pDP utilizes the formulation (1)-(6). The usage of clustering techniques for finding feasible solutions for combinatorial problems is hardly new. For example, in vehicle routing and scheduling, the "cluster-first, route-second" (see [10], [11]) and "route-first, cluster-second" (see [12], [13]) paradigms were used to ease the computational burden of the hard combinatorial problem. What sets the decremental clustering method apart is that it provides guarantees for optimality. Decremental clustering was also proposed for the solution of the vertex p-center problem in [14] and [15].

We present the decremental clustering method with the same notation and vocabulary as it was developed in [4]. A clustering of the n nodes, denoted by $\mathcal{C}$ is a family $\{C_i : i = 1,\dots,m\}$ such that $C_i \cap C_j = \emptyset$ for every $1 \le i < j \le m$ and $\cup\{C_i : 1,\dots,m\} = I$. 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 $\texttt{pDP}(D,p)$. The correctness of the decremental clustering method is supported by the following result (proved in [4]).

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 $\texttt{pDP}(D^{\mathcal{C}},p)$ provides an upper bound of problem $\texttt{pDP}(D,p)$.

The decremental clustering method works as follows. A lower bound $L \le z^*$ is computed using a k-means algorithm [16], in a procedure named $\texttt{heuristicPDP}(D,p,s)$ whose pseudocode is described in Algorithm 1. Since the k-means clustering is a stochastic method, it is repeated multiple times as long as the value of the lower bound keeps increasing (the authors of [4] stop after $s = 10$ iterations without being able to improve its value). An initial upper bound U is computed as the largest dissimilarity between any two points. Using the lower bound L, we build an initial sufficiently refined clustering $\mathcal{C}$ and a reduced dissimilarity matrix $D^{\mathcal{C}}$, using the procedure $\texttt{initialClustering}(D,p,L)$,

Algorithm 1 `heuristicPDP` (D, p, s)

$D, p, s \leftarrow$ **inputs**
$L \leftarrow 0$
$streak \leftarrow 0$
repeat
  $\mathcal{C} \leftarrow$ `k-means`(D, p)
  for $i = [1 : p]$ **do**
    $k_i \leftarrow$ point in C_i closest to its center
  end for
  $d \leftarrow \min(D(k_i, k_j) : 1 \leq i < j \leq p)$
  if $L < d$ **then**
    $streak \leftarrow streak + 1$
  else
    $L \leftarrow d$
    $streak \leftarrow 0$
  end if
until $streak = s$
return L

Algorithm 2 `initialClustering` (D, p, L)

$D, p, L \leftarrow$ **inputs**
$m \leftarrow p$
$\mathcal{C} \leftarrow$ `k-means`(D, p)
`compute` $D^{\mathcal{C}}$
while $\max(D^{\mathcal{C}}(i, i) : 1 \leq i \leq m) > L$ **do**
  $i^* \leftarrow \arg\max(D^{\mathcal{C}}(i, i) : 1 \leq i \leq m)$
  $m \leftarrow m + 1$
  $C^1_*, C^2_* \leftarrow$ `k-means`$(C_{i^*}, 2)$
  $C_{i^*} \leftarrow C^1_*$
  $C_m \leftarrow C^2_*$
  `recompute` $D^{\mathcal{C}}$
end while
return $\mathcal{C}, D^{\mathcal{C}}$

Algorithm 3 `splitAndAdd` $(S, W, \mathcal{C}, D^{\mathcal{C}})$

$S, W, \mathcal{C}, D^{\mathcal{C}} \leftarrow$ **inputs**
if $S = \emptyset$ **then**
  $\mathcal{C} \leftarrow \mathcal{C}$, $D^{\mathcal{C}} \leftarrow D^{\mathcal{C}}$ (i.e., do nothing)
else
  $m \leftarrow$ `size` $D^{\mathcal{C}}$
  $(s^*, w^*) \leftarrow \arg\min(D^{\mathcal{C}}(s, w) : s \in S, w \in W)$
  if $w^* \in S$ **then**
    $i^* \leftarrow \arg\max(D^{\mathcal{C}}(u, u) : u \in \{s^*, w^*\})$
  else
    $i^* \leftarrow s^*$
  end if
  $C^1_*, C^2_* \leftarrow$ `k-means`$(C_{i^*}, 2)$
  $C_{i^*} \leftarrow C^1_*$
  $C_{m+1} \leftarrow C^2_*$
  `recompute` $D^{\mathcal{C}}$
end if
return $\mathcal{C}, D^{\mathcal{C}}$

Algorithm 4 `solvePDP` $(D^{\mathcal{C}}, p, U, W)$

$D^{\mathcal{C}}, p, U, W \leftarrow$ **inputs**
for $k = [1 : p + 1]$ **do**
  $U_h(k) \leftarrow \min(D^{\mathcal{C}}(i, j) : 1 \leq i, j \leq p + 1, i \neq k, j \neq k)$
end for
if $\max(U_h(k) : 1 \leq k \leq p + 1) = U$ **then**
  $k^* \leftarrow \arg\max(U_h(k) : 1 \leq k \leq p + 1)$
  $W \leftarrow \{W(i) : 1 \leq i \leq p + 1, i \neq k^*\}$
  $U \leftarrow U$
else
  $U, W \leftarrow$ `solve` (1)-(6)
end if
return U, W

whose pseudocode is described in Algorithm 2. The main idea of the procedure is to find the clusters with the largest inter-node distances and split them into two, until the inter-node distance in all clusters is less than L (which implies that it is less than z^*). After these initial steps, the method uses two auxiliary sets S and W (with $S \subseteq W$), where S represents the set of optimal nonsingleton clusters ($S = \{C_i : |C_i| \geq 2, i = 1, \ldots, m\}$), and W is the complete optimal solution to the restricted pDP (w.r.t. $D^{\mathcal{C}}$). Iteratively, the sets S and W are used to refine the current clustering, resulting in a refined clustering $\mathcal{C}$ and dissimilarity matrix $D^{\mathcal{C}}$, using the procedure `splitAndAdd`$(S, W, \mathcal{C}, D^{\mathcal{C}})$, which is described in Algorithm 3. The resulting reduced pDP is then solved, yielding an upper bound U on the full problem, and its optimal solution is used to update the sets S, W. The solution procedure `solvePDP`$(D^{\mathcal{C}}, p, U, W)$ has two parts – a heuristic ''preprocessing'' method and an exact solver. The heuristic procedure is based on the observation that in a large number of iterations, the optimal value of the pDP problem does not decrease from one iteration to the next, which is a common feature of decremental relaxation schemes [17]. Therefore, before executing the exact solver, the heuristic procedure checks the best possible selection of p points out of the $p + 1$ points obtained from the `splitAndAdd` procedure. If the value of this solution equals the upper bound U from the previous iteration, the associated solution is then optimal, and there is no need to execute the exact solver. In our implementation, the exact solver comprise of solving the model (1)-(6) using the modelling package `JuMP` [18] in `Julia` [19], and the `GUROBI` solver [20]. The pseudocode of the `solvePDP` procedure is described in Algorithm 4. The whole decremental clustering algorithm is described in Algorithm 5. It also incorporates a possible knowledge on a lower bound L of the optimal value of (1)-(6), which will be explained in the forthcoming section.

III. DISCRETIZATIONS

The continuous variant of the pDP is extremely difficult to solve and the techniques for approaching it are usually

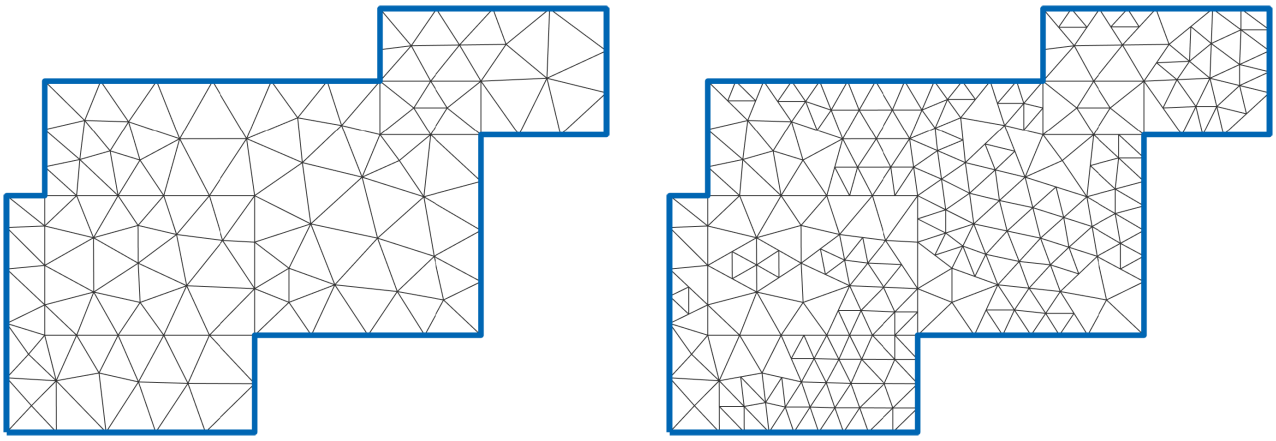

FIGURE 1. Initial mesh (left) and mesh after one round of refinement (right), first example.

Algorithm 5 `decrementalClustering` (D, p, L)

$D, p, L \leftarrow$ **inputs**
$L' \leftarrow$ `heuristicPDP`$(D, p, 10)$
$L \leftarrow \max(L, L'),\ U \leftarrow \max(D(i,j) : 1 \le i < j \le n)$
$\mathcal{C}, D^{\mathcal{C}} \leftarrow$ `initialClustering`(D, p, L)
$S \leftarrow \emptyset,\ W \leftarrow \emptyset$
repeat
  $\mathcal{C}, D^{\mathcal{C}} \leftarrow$ `splitAndAdd`$(S, W, \mathcal{C}, D^{\mathcal{C}})$
  $U, W \leftarrow$ `solvePDP` $(D^{\mathcal{C}}, p, U, W)$
  $S \leftarrow \{w \in W : |C_w| \ge 2\}$
until $S = \emptyset$ **or** $L = U$
return $U, X \leftarrow \{C_w : w \in W\}$

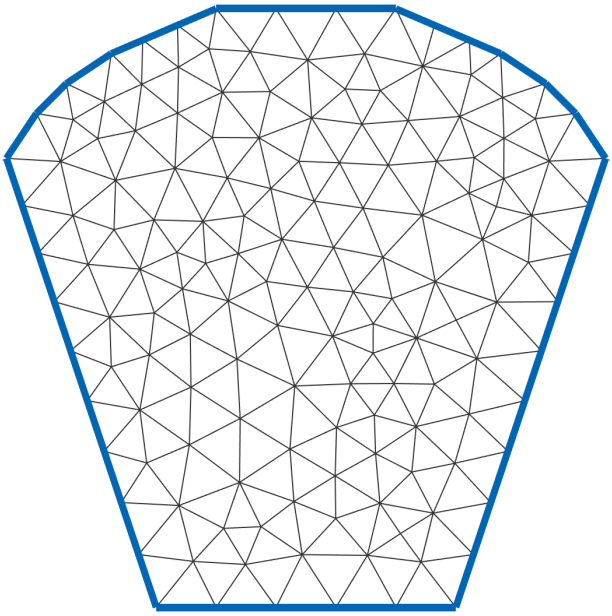

FIGURE 2. Initial mesh, second example.

bound to convex feasible spaces [21]. In order to apply the pDP framework developed earlier for general spaces, such as classrooms, restaurants, beaches, etc., the feasible space of the problem – the possible locations of the points – is discretized. This discretization is carried out by triangulation (or mesh generation) of the two dimensional feasible area, with the vertices of the triangles being the possible feasible points [22]. Naturally, a question arises about the relationship between the granularity of the triangulation and the objective value of the optimal solution of the associated pDP – the finer the mesh, the better the solution (with higher smallest distance between any two selected points). We address this issue by devising a mesh refinement scheme and solving a series of pDPs on a progressively finer mesh. The mesh refinement works as follows: First, the area of each triangle in the triangulation is computed. The triangles with larger than average area are then split into 4 triangles by adding points is the middle of their sides. The process is illustrated in Figure 1. A side effect of the refinement scheme is that by solving the pDP on a coarser mesh, we obtain a guaranteed lower bound on the optimal value of the pDP on a finer mesh.

Another "discretization" can be devised in the space of possible values of the dissimilarity matrix D. As the problem gets more difficult with more unique values in D, with each unique value having a separate binary variable in the model (1)-(6), we can greatly reduce the number of these added variables by rounding the elements of D. The trade-off between the optimal objective value and computational efforts of these two discretization schemes are investigated in the following section. The pseudocode of the method used in the computational experiments, called `refinePDP`(D, p, L, r, T), is described in Algorithm 6. It supposes that an initial mesh with p and D is available. The other inputs are a lower bound L (if there is no prior knowledge about a possible lower bound, then $L = 0$), a rounding factor r (with $r = -1$ being rounding to the nearest tenths, $r = 1$ to nearest integers, $r = 2$ to nearest tens, etc., and $r = 0$ no rounding) and a time limit for the computation T.

IV. COMPUTATIONAL EXPERIMENTS

We investigate the effect of discretization on two examples. The first one is a general shape depicted in Figure 1 and

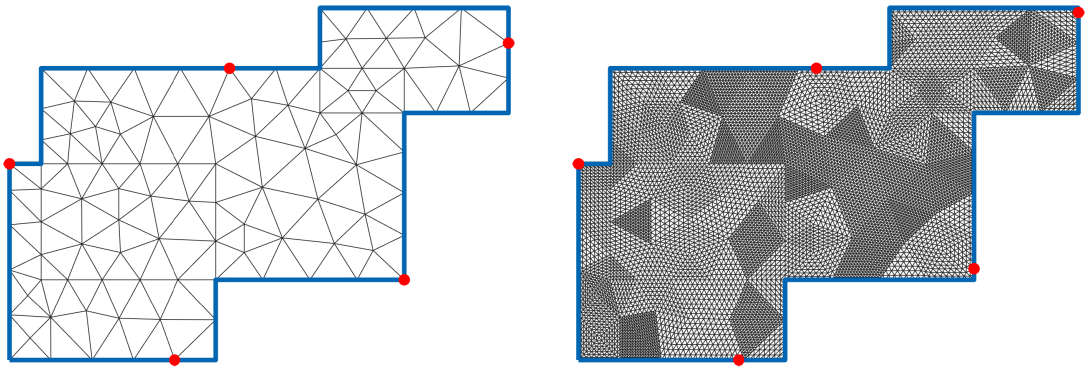

FIGURE 3. Optimal placements, $p = 5$, $r = 1$. Initial mesh with $n = 102$, $z^r = 1{,}273.88$ (left). Final mesh with $n = 8{,}745$, $z^r = 1{,}338.09$ (right).

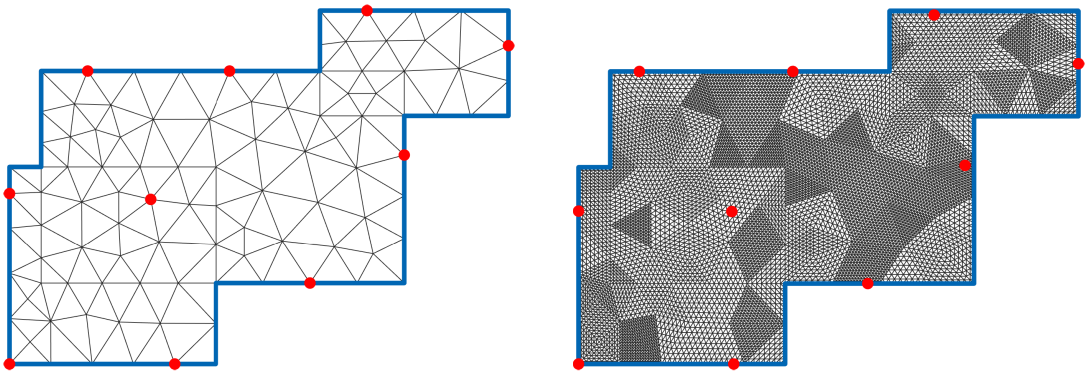

FIGURE 4. Optimal placements, $p = 10$, $r = 1$. Initial mesh with $n = 102$, $z^r = 749.72$ (left). Final mesh with $n = 8{,}745$, $z^r = 805.25$ (right).

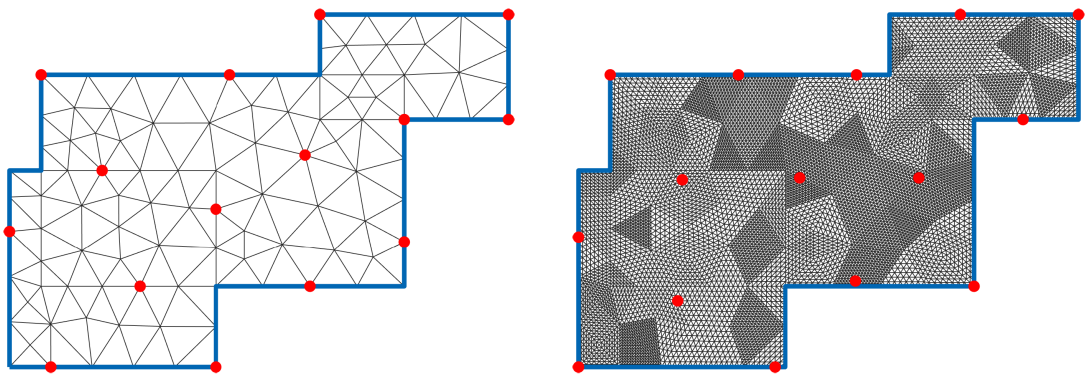

FIGURE 5. Optimal placements, $p = 15$, $r = 1$. Initial mesh with $n = 102$, $z^r = 552.63$ (left). Final mesh with $n = 8{,}745$, $z^r = 622.75$ (right).

the second one an auditorium-like shape shown in Figure 2. In both examples, we examine the computational efforts to solve the pDP problem for progressively finer mesh granularity and for different values of $p \in \{5, 10, 15, 30\}$ and $r \in \{0, 1, 2\}$. In both examples, the time limit T was set to 24 hours. For the first example, the maximal distance between any two points (''the problem diameter'') was $\max(D) = 3240.8$, for the second example, it was $\max(D) = 2180$. For each problem instance, we report: n the number of points, Δ the square root of the area of the largest triangle in the mesh (a useful measure of the granularity of the grid), r the rounding factor, k_{max} the number of distinct elements in D', z'^* the optimal objective value of the problem with D', z^r the ''real'' objective value (without rounding), and t the time it took to find the optimum. If the computations were not finished (the time limit was reached), the best upper bound U is reported in square brackets. If the computation of the instance was terminated prematurely, because during

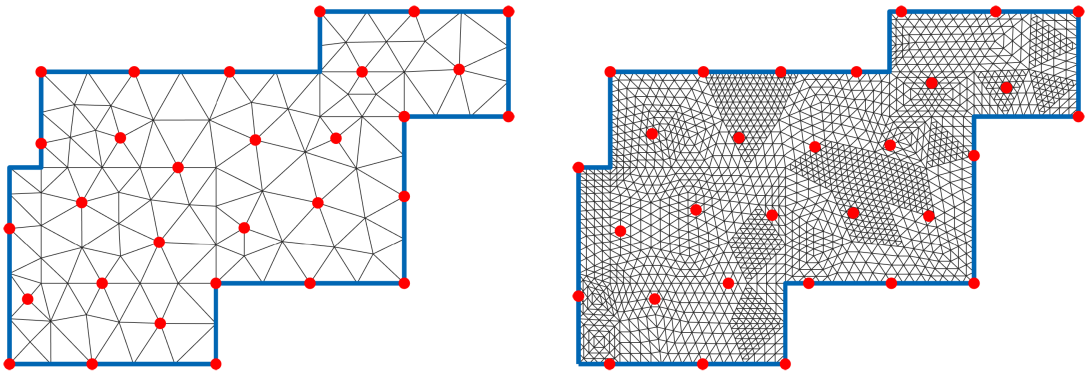

FIGURE 6. Optimal placements, $p = 30$, $r = 2$. Initial mesh with $n = 102$, $z^r = 325.77$ (left). Final mesh with $n = 1{,}891$, $z^r = 395.42$ (right).

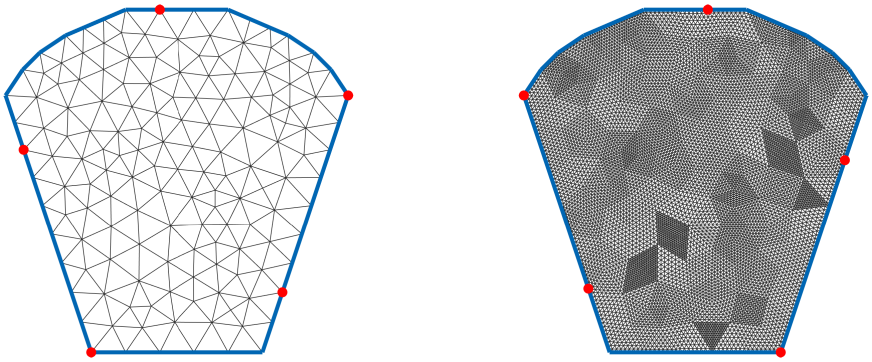

FIGURE 7. Optimal placements, $p = 5$, $r = 1$. Initial mesh with $n = 150$, $z^r = 1{,}139.49$ (left). Final mesh with $n = 9{,}313$, $z^r = 1{,}183.01$ (right).

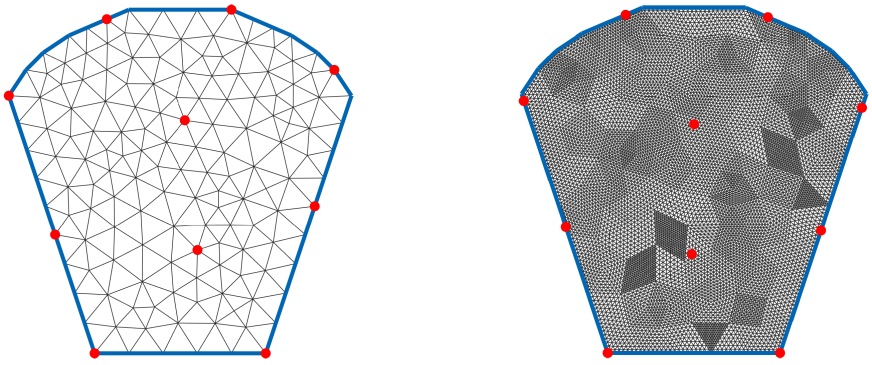

FIGURE 8. Optimal placements, $p = 10$, $r = 1$. Initial mesh with $n = 150$, $z^r = 694.62$ (left). Final mesh with $n = 9{,}313$, $z^r = 747.72$ (right).

the computation, the upper bound U was equal to the lower bound L from the solution of coarser discretization (i.e., the finer discretization did not improve on the optimal value of the solution) the instance is marked with a '*'. The instances that were not computed, because the time limit was already reached, are marked by a '–'. The computations were carried out on an ordinary computer with 3.2 GHz i5-4460 CPU and 16 GB RAM.

The numerical results of the computations are reported in Table 1 for the first example and Table 2 for the second one. The optimal placements (the ones with the best value of z^r) are shown in Figures 3-6 for the first example and in Figures 7-10 for the second one. The first general observations is that in order to decrease Δ by half, the number of points n needs to be roughly quadrupled (which follows from the way the mesh gets refined). The second general observation can be made about the impact of the rounding factor r: Apart from a single instance, there was no difference in the "real" objective value between instances with $r = 0$ and $r = 1$, while there was a huge difference

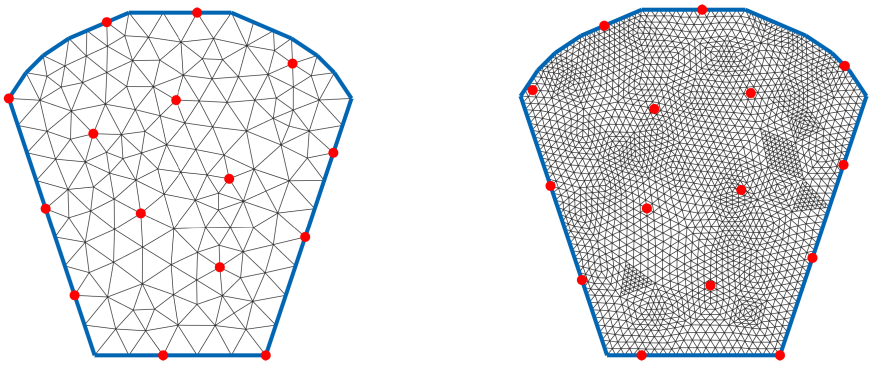

FIGURE 9. Optimal placements, $p = 15$, $r = 1$. Initial mesh with $n = 150$, $z^r = 518.40$ (left). Final mesh with $n = 2{,}273$, $z^r = 555.93$ (right).

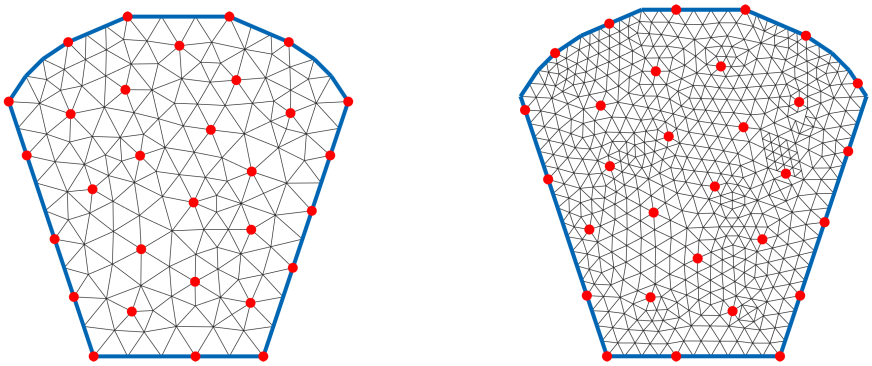

FIGURE 10. Optimal placements, $p = 30$, $r = 1$. Initial mesh with $n = 150$, $z^r = 323.69$ (left). Final mesh with $n = 575$, $z^r = 352.96$ (right).

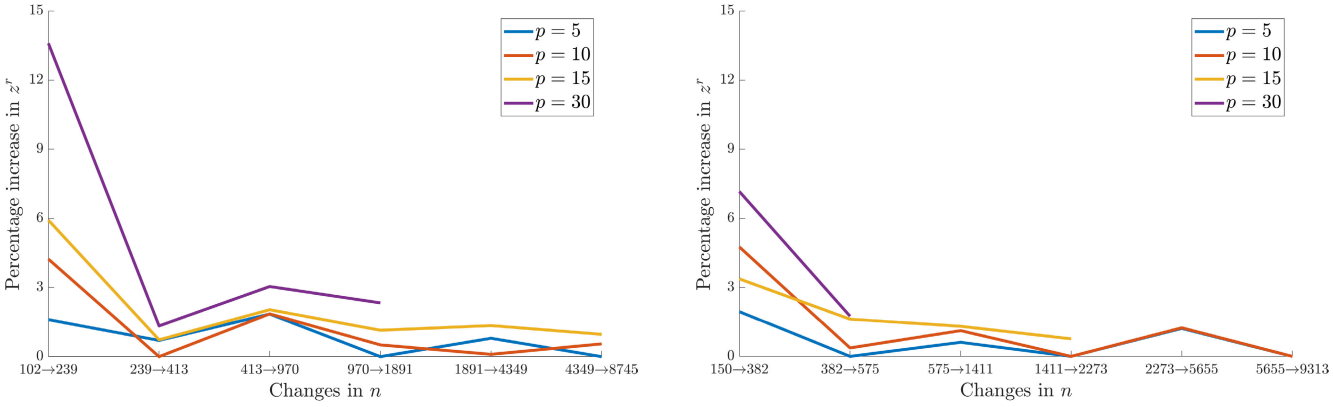

FIGURE 11. The effect of the mesh refinement on the value of z^r. First example on the left, second example on the right.

between the computational time t. From these computational experiments, there is no doubt that the rounding procedure presents a substantial benefit, as the instances with rounding were computed around an order of magnitude faster than the instances without rounding, and some large instances could not be computed within the time limit without the use of rounding.

The difference between $r = 1$ and $r = 2$ is much more nuanced. In 83 % of the instances, the computations with $r = 2$ were faster. On the other hand, using $r = 2$ instead of $r = 1$ results on average in 0.43 % worse value of z^r. Premature termination is also more prevalent in the $r = 2$ case. Of the 48 successfully computed instances it occurred 14 times for $r = 2$, compared to 8 times for $r = 1$.

The coarseness of the mesh naturally plays a crucial role in both the objective value z^r and the computational time t, with finer meshes having higher objective value z^r, but because of the increase in the number of variables, take progressively

TABLE 1. Results of the computation, first example, max(D) = 3,240.8.

				$p = 5$			$p = 10$		
n	Δ	r	k_{max}	z^*	z^r	t [s]	z^*	z^r	t [s]
		0	5,043	1,273.88	1,273.88	0.25	749.72	749.72	10.9
102	198.4	1	2,060	1,274	1,273.88	0.42	750	749.72	1.96
		2	293	1,270	1,273.88	0.95	750	749.72	2.57
		0	27,919	1,294.33	1,294.33	1.58	781.46	781.46	107
239	141.0	1	2,688	1,294	1,294.33	1.59	781	781.46	46.8
		2	304	1,290	1,287.46	0.62	780	781.46	7.07
		0	83,555	1,303.45	1,303.45	1.84	781.46*	781.46	475*
413	92.7	1	2,918	1,303	1,303.45	1.61	781*	781.46	35.4*
		2	314	1,300	1,303.45	0.50	780*	781.46	19.3*
		0	459,211	1,327.51	1,327.51	5.25	795.95	795.95	991
970	66.8	1	3,029	1,328	1,327.51	1.31	796	795.95	198
		2	317	1,330	1,327.51	1.00	800	795.95	66.6
		0	1,749,744	1,327.51	1,327.51	31.4	800.01	800.01	4,061
1,891	43.2	1	3,113	1,328*	1,327.51	2.51*	800	800.01	407
		2	322	1,330*	1,327.51	2.95*	800*	795.95	242*
		0	9,220,027	1,338.09	1,338.09	41.2	[802.98]	[802.98]	$T = 24$h
4,349	30.2	1	3,157	1,338	1,338.09	16.7	801	800.81	1,426
		2	324	1,340	1,336.61	15.2	800*	795.95	1,170*
		0	37,239,029	1,338.09*	1,338.09	4,134*	–	–	–
8,745	19.5	1	3,202	1,338*	1,338.09	43.5*	805	805.25	2,546
		2	325	1,340*	1,336.61	54.3*	810	805.07	1,922

				$p = 15$			$p = 30$		
n	Δ	r	k_{max}	z^*	z^r	t [s]	z^*	z^r	t [s]
		0	5,043	552.63	552.63	50.6	325.77	325.77	12.6
102	198.4	1	2,060	553	552.63	6.73	326	325.77	32.3
		2	293	550	546.54	4.41	330	325.77	23.3
		0	27,919	585.37	585.37	195	370.06	370.06	256
239	141.0	1	2,688	585	585.37	69	370	370.06	56.7
		2	304	590	585.37	22.9	370	365.37	33.2
		0	83,555	589.65	589.65	1,162	375.00	375.00	4,707
413	92.7	1	2,918	590	589.65	135	375	375.00	415
		2	314	590*	585.37	66.1*	380	375.00	206
		0	459,211	601.68	601.68	26,771	[390.82]	[390.82]	$T = 24$h
970	66.8	1	3,029	602	601.65	1,153	386	386.41	4,635
		2	317	600	595.33	791.8	390	385.07	3,334
		0	1,749,744	[614.82]	[614.82]	$T = 24$h	–	–	–
1,891	43.2	1	3,113	609	608.55	4,051	395	395.42	36,017
		2	322	610	606.28	2,837	400	395.42	24,706
		0	9,220,027	–	–	–	–	–	–
4,349	30.2	1	3,157	617	616.77	15,699	[413]	[413]	$T = 24$h
		2	324	620	615.21	16,695	[420]	[420]	$T = 24$h
		0	37,239,029	–	–	–	–	–	–
8,745	19.5	1	3,202	623	622.75	21,044	–	–	–
		2	325	620*	615.21	18,185*	–	–	–

longer to compute. The improvement of the objective value z^r based on the mesh refinement is captured in Figure 11, which shows the percentage improvements caused by the increases in the number of mesh points n. Additionally, the value of p also has an extensive impact on the computational time. Similar to the findings in [4], we also find that the decremental clustering scheme works very well for smaller values of p, but the computations become progressively more costly as p increases. The value of $p = 30$ seems to be close to the limit of applicability of the method. On the one hand, it means that in the context of social distancing it is well applicable for use in situations, when the available space does not allow for more than 30 persons, such as in classrooms, restaurants, or offices. On the other hand, it still can be used to compute valid upper bounds on the objective value even for larger problems, which can be explored by various heuristics.

There is also a significant difference in the "difficulty" between the two examples. Although the number of points n and unique values k_{max} were similar for both examples in the individual mesh refinement steps, the computational times differ quite a lot, mainly for larger values of p. This can be attributed to the "dual degeneracy" [4] occurring when a larger number of clusters can be rearranged from one iteration to the next to find solutions of the same cost. The second example is more symmetric than the first one, meaning that it has a larger number of the possible rearrangements. Naturally,

TABLE 2. Results of the computation, second example, max(D) = 2,180.

				$p=5$			$p=10$		
n	Δ	r	k_{max}	z^*	z^r	t [s]	z^*	z^r	t [s]
		0	10,901	1,139.49	1139.49	21.9	694.62	694.62	87.1
150	143.4	1	1,846	1,139	1139.49	4.05	695	694.62	37.0
		2	205	1,140	1139.49	0.59	690	687.56	5.28
		0	71,690	1,161.70	1161.70	61.4	727.68	727.68	26.3
382	107.2	1	2,010	1,162	1161.70	5.65	728	727.68	12.7
		2	212	1,160	1161.70	6.68	730	727.68	10.3
		0	163,067	1,161.70*	1161.70	36.4*	730.36	730.36	202
575	69.7	1	2,062	1,162*	1161.70	47.59*	730	730.36	100
		2	215	1,160*	1161.70	5.65*	730*	727.68	18.4
		0	983,031	1,168.8	1168.82	374	738.54	738.54	3,741
1,411	53.0	1	2,120	1,169	1168.82	169	739	738.54	312
		2	217	1,170	1168.82	26.3	740	737.39	240
		0	2,258,704	1,168.8*	1168.82	104	738.54*	738.54	2,342*
2,273	34.5	1	2,144	1,169*	1168.82	170*	739*	738.54	460*
		2	218	1,170*	1168.82	21.9*	740*	727.68	138*
		0	15,805,824	1,183.01	1183.01	2,272	[767.81]	[767.81]	T = 24h
5,655	26.2	1	2,159	1,183	1183.01	263	748	747.72	1,185
		2	218	1,180	1177.31	227	750	746.81	1,215
		0	42,931,260	1,183.01*	1183.01	1,541	–	–	–
9,313	16.7	1	2,168	1,183*	1183.01	252*	748*	747.72	3,818*
		2	219	1,180*	1177.31	178*	750*	746.81	2,720*

				$p=15$			$p=30$		
n	Δ	r	k_{max}	z^*	z^r	t [s]	z^*	z^r	t [s]
		0	10,901	518.40	518.40	84.2	323.69	323.69	96.0
150	143.4	1	1,846	518	518.40	47.4	324	323.69	65.2
		2	205	520	518.40	8.93	320	315.65	35.9
		0	71,690	535.89	535.89	3,754	346.88	346.88	5,231
382	107.2	1	2,010	536	535.89	415	347	346.88	439
		2	212	540	535.04	155	350	345.76	284
		0	163,067	[562.73]	[562.73]	T = 24h	[361.08]	[361.08]	T = 24h
575	69.7	1	2,062	545	544.59	793	353	352.96	2,763
		2	215	540*	535.04	611*	350	345.76	1,577*
		0	983,031	–	–	–	–	–	–
1,411	53.0	1	2,120	552	551.72	12,203	[366]	[366]	T = 24h
		2	217	550	546.01	9,190	[360]	[366]	T = 24h
		0	2,258,704	–	–	–	–	–	–
2,273	34.5	1	2,144	556	555.93	51,344	–	–	–
		2	218	560	555.21	55,682	–	–	–
		0	15,805,824	–	–	–		–	–
5,655	26.2	1	2,159	[572]	[572]	T = 24h	–	–	–
		2	218	[570]	[570]	T = 24h	–	–	–

Algorithm 6 `refinePDP` (D, p, L, r, T)

$D, p, L, r, t \leftarrow$ **inputs**
$D' \leftarrow$ `round`(D, r)
$U, X \leftarrow$ `decrementalClustering`(D', p, L)
$L \leftarrow U$
repeat
  $D, p \leftarrow$ `refineMesh`(D, p)
  $D' \leftarrow$ `round`(D, r)
  $U, X \leftarrow$ `decrementalClustering`(D', p, L)
  $L \leftarrow U$
until `runTime` $> T$
return U, X

the optimal placements in Figures 7-10 have a straightforward "symmetric" counterpart with the same objective value.

Lastly, the hexagonal pattern, that seems to emerge in Figures 6 and 10 (both with $p = 30$) is no accident – the hexagonal pattern is optimal for many location problems (including the p-dispersion problem) with numerous facilities covering a large area [23].

V. CONCLUSION

In this article we have studied the problem of locating persons in a given area, that should abide to social distancing measures such as those arising in the time of COVID-19 and similar viruses. We have argued that the $p-$dispersion problem can be used to efficiently model these situations. We devised a discretization scheme that was build on top of the decremental clustering method to get computationally attainable solutions, which worked very well in the computational study on the two artificial examples, especially for smaller values of p.

We have investigated the effect of rounding of the dissimilarity matrix D on the computational effort and conclude that it is an indispensable part of the discretization scheme that has virtually no disadvantages in terms of the quality of the obtained solution. We have also seen the substantial increase in computational efforts for higher values of p, which can be contributed to the ''dual degeneracy'' of the clustering scheme.

For future research, fast heuristics that run parallel to the decremental clustering scheme might further improve on the computational time and the size of the problems that are solvable by the presented method. Also, a mesh refinement scheme based on the current optimal placement (instead of the triangle size as presented here) could lead to improvements in the objective value.

REFERENCES

[1] V. Chamola, V. Hassija, V. Gupta, and M. Guizani, "A comprehensive review of the COVID-19 pandemic and the role of IoT, drones, AI, blockchain, and 5G in managing its impact," *IEEE Access*, vol. 8, pp. 90225–90265, 2020, doi: 10.1109/ACCESS.2020.2992341.

[2] I. D. Moon and S. S. Chaudhry, "An analysis of network location problems with distance constraints," *Manage. Sci.*, vol. 30, no. 3, pp. 290–307, Mar. 1984, doi: 10.1287/mnsc.30.3.290.

[3] D. Sayah and S. Irnich, "A new compact formulation for the discrete p-dispersion problem," *Eur. J. Oper. Res.*, vol. 256, no. 1, pp. 62–67, Jan. 2017, doi: 10.1016/j.ejor.2016.06.036.

[4] C. Contardo, "Decremental clustering for the solution of p-dispersion problems to proven optimality," *Informs J. Optim.*, vol. 2, no. 2, pp. 134–144, Apr. 2020, doi: 10.1287/ijoo.2019.0027.

[5] E. Erkut, "The discrete p-dispersion problem," *Eur. J. Oper. Res.*, vol. 46, no. 1, pp. 48–60, May 1990, doi: 10.1016/0377-2217(90)90297-O.

[6] M. J. Kuby, "Programming models for facility dispersion: The p-dispersion and maxisum dispersion problems," *Geograph. Anal.*, vol. 19, no. 4, pp. 315–329, Sep. 2010, doi: 10.1111/j.1538-4632.1987.tb00133.x.

[7] E. Erkut and S. Neuman, "Analytical models for locating undesirable facilities," *Eur. J. Oper. Res.*, vol. 40, no. 3, pp. 275–291, Jun. 1989, doi: 10.1016/0377-2217(89)90420-7.

[8] B. Saboonchi, P. Hansen, and S. Perron, "MaxMinMin p-dispersion problem: A variable neighborhood search approach," *Comput. Oper. Res.*, vol. 52B, pp. 251–259, Dec. 2014, doi: 10.1016/j.cor.2013.09.017.

[9] D. Pisinger, "Upper bounds and exact algorithms for p-dispersion problems," *Comput. Oper. Res.*, vol. 33, no. 5, pp. 1380–1398, May 2006, doi: 10.1016/j.cor.2004.09.033.

[10] M. M. Solomon, "Algorithms for the vehicle routing and scheduling problems with time window constraints," *Oper. Res.*, vol. 35, no. 2, pp. 254–265, Apr. 1987, doi: 10.1287/opre.35.2.254.

[11] O. Braysy and M. Gendreau, "Vehicle routing problem with time windows—Part I: Route construction and local search algorithms," *Transp. Sci.*, vol. 39, no. 1, pp. 104–118, Feb. 2005, doi: 10.1287/trsc.1030.0056.

[12] J. E. Beasley, "Route first–cluster second methods for vehicle routing," *Omega*, vol. 11, no. 4, pp. 403–408, 1983, doi: 10.1016/0305-0483(83)90033-6.

[13] C. Prins, P. Lacomme, and C. Prodhon, "Order-first split-second methods for vehicle routing problems: A review," *Transp. Res. C, Emerg. Technol.*, vol. 40, pp. 179–200, Mar. 2014, doi: 10.1016/j.trc.2014.01.011.

[14] D. Chen and R. Chen, "New relaxation-based algorithms for the optimal solution of the continuous and discrete p-center problems," *Comput. Oper. Res.*, vol. 36, no. 5, pp. 1646–1655, May 2009, doi: 10.1016/j.cor.2008.03.009.

[15] C. Contardo, M. Iori, and R. Kramer, "A scalable exact algorithm for the vertex p-center problem," *Comput. Oper. Res.*, vol. 103, pp. 211–220, Mar. 2019, doi: 10.1016/j.cor.2018.11.006.

[16] J. B. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math. Statist. Probab.*, Berkeley, CA, USA: Univ. California Press, vol. 1, 1967, pp. 281–297.

[17] D. Aloise and C. Contardo, "A sampling-based exact algorithm for the solution of the minimax diameter clustering problem," *J. Global Optim.*, vol. 71, pp. 613–630, Mar. 2018, doi: 10.1007/s10898-018-0634-1.

[18] I. Dunning, J. Huchette, and M. Lubin, "JuMP: A modeling language for mathematical optimization," *SIAM Rev.*, vol. 59, no. 2, pp. 295–320, Jan. 2017, doi: 10.1137/15M1020575.

[19] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, "Julia: A fresh approach to numerical computing," *SIAM Review*, vol. 59, no. 1, pp. 65–98, Feb. 2017, doi: 10.1137/141000671.

[20] *Gurobi Optimizer Reference Manual*. Gurobi Optimization, LLC, Beaverto, OR, USA, 2020. [Online]. Available: http://www.gurobi.com/

[21] Z. Drezner and E. Erkut, "Solving the continuous p-dispersion problem using non-linear programming," *J. Oper. Res. Soc.*, vol. 46, no. 4, pp. 516–520, Apr. 1995, doi: 10.1057/jors.1995.70.

[22] P. L. George, "Automatic mesh generation and finite element computation," *Handbook Numer. Anal.*, vol. 4, pp. 69–190, Jan. 1996, doi: 10.1016/S1570-8659(96)80003-2.

[23] T. Drezner and Z. Drezner, "Leader-follower models in facility location," in *Spatial Interaction Models: Facility Location Using Game Theory*, L. Mallozi, E. D'Amato, and P. M. Pardalos, Eds. Cham, Switzerland: Springer, 2017, pp. 73–104, doi: 10.1007/978-3-319-52654-6.

JAKUB KUDELA received the M.S. degree in mathematical engineering and the Ph.D. degree in applied mathematics from the Brno University of Technology, in 2014 and 2019, respectively.

Since 2018, he has been working as a Research Assistant at the Institute of Automation and Computer Science, Brno University of Technology. His research interest includes the development of computational methods for various optimization problems and engineering applications.