Laboratoire d'Analyse et Modélisation de Systèmes pour l'Aide à la Décision
UMR 724

CAHIER DU LAMSADE

387

Mars 2019

A new robust p-center problem to locate shelters in wild fire context

M. Demange, V. Gabrel, M. Haddad, C. Murat

A new robust p-center problem to locate shelters in wild fire context

Marc Demange [*,2], Virginie Gabrel [†,1], Marcel Haddad [‡,1 2], and Cécile Murat [§,1]

[1]Université Paris-Dauphine, PSL Research University, CNRS, UMR 7243, LAMSADE, Paris, France
[2]School of Science RMIT University, Melbourne, Vic., Australia

Abstract

The p-center problem on a graph is to find a set C of p vertices that minimizes the radius which is the maximum distance between any vertex and C. We propose a robust optimization model for this problem with uncertainty on vertices. This problem is inspired by a wild fire management problem. The graph represents the adjacency of zones of a landscape, where each vertex represents a zone. We consider a finite set of fire scenarios, where the fire is restricted to one zone. Given a solution C, its radius may change in some scenarios since some evacuation paths become impracticable. The objective is to find a robust p-center that minimizes the worst effective radius over all scenarios. We introduce this new problem, the Robust p-Center Problem, and propose a first formulation based on 0-1 Linear Programming. We give preliminary experimental results and show how to adapt our model to other fire configurations. Finally, we present a specific realistic case relevant to our approach.

1 Introduction

In the prevention phase of wild fires, an important problem is to determine shelters location in a given area. The problem is basically to place p shelters minimizing the maximum

[*]marc.demange@rmit.edu.au
[†]gabrel@lamsade.dauphine.fr
[‡]marcel.haddad@dauphine.psl.eu
[§] murat@lamsade.dauphine.fr
Work supported by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 691161

distance people have to go to get one of these shelters in case of fire. From an Operations Research perspective, the problem of locating p centers (warehouses, shelters ...) has been extensively studied. In particular, mathematical programming approaches enable to determine optimal solution to the classical p-center problem [ELP04].

In our model, the landscape is represented by a directed adjacency graph $G = (V, E)$. Each node corresponds to a zone and two nodes i and j are connected by a directed edge if it is possible for somebody to go directly from i to j without passing through another area. Each edge (i, j) is valued by a positive weight l_{ij} that can be seen as a distance or a travel time. Since vertices represent a zone, possibly large, the distance (or travelling time) between adjacent vertices can be measured between median points in these areas or as a maximum distance; this choice does not affect the model further. We assume that shelters have to be located on vertices. Then, considering a shelter on vertex i and a zone represented by vertex j, the distance d_{ji} from j to i is the length of a shortest path from j to i in G. Given a solution defined as a set of p vertices, the evacuation distance of zone j is the shortest distance between j and one of the p shelters. The objective value of a solution, called the safety distance, corresponds to the longest evacuation distance of vertices. The p-center problem is then to compute a solution that minimizes the safety distance.

In the context of wild fires, one difficulty is due to the uncertainty associated to the fire ignition and spread. To address this difficulty, the robust version of the p-center problem is usually defined by introducing uncertainty on weights: each weight may vary in an uncertainty set (usually an interval) and the problem is to determine a p-center minimizing the worst case or the maximum regret [AB97, AB00, Ave03]. In these previous works, weights vary independently one from each other. In our context, this independency hypothesis is not relevant since, if a fire ignites on a vertex, all weights of its adjacent edges are modified in the same way. For this reason, we represent uncertainty with a set of scenarios. This leads to a new problem called the *robust p-center problem* (RpCP) that takes into account this representation of uncertainty.

This article is organised as follows: in section 2 we define the robust p-center problem, in Section 3 we propose a new formulation based on 0-1 Linear Programming, in section 4 we give some preliminary experimental results showing the tractability of our approach. Our approach can easily be adapted to other fire configurations, as mentioned in section 5. In the last section we present a specific realistic case and its polynomial complexity.

2 Definition of the robust p-center problem

To model fire hazards, a scenario is associated to a specific fire outbreak in the landscape represented by a graph $G = (V, E)$ with n vertices. First, we restrict ourselves to single fire outbreak events and each scenario s corresponds to a single vertex s on fire. This

restriction is motivated by our primary focus on a relatively short time period after ignition and corresponds to the case where everybody can escape to a shelter before the fire spreads on adjacent zones. The *operational graph* associated with the scenario s, denoted G^s, is obtained by removing all edges arriving on s: $G^s = (V, E^s)$ with $E^s = E \setminus \omega^-(s)$ where $\omega^-(s)$ is the set of edges (j, s) with ending vertex s. In G^s, vertex s is no longer accessible from another vertex. For this reason, some shortest paths in G^s, thus avoiding s, will differ from the original ones obtained in the graph G. For all i and j, d^s_{ij} denotes the length of the shortest path from i to j in graph G^s. By convention, for all $j \in V \setminus \{s\}$, we have $d^s_{js} = +\infty$.

Given a solution with p vertices and a scenario s, the "classical evacuation distance" of a vertex j is the shortest distance between j and its nearest shelter. This strategy is not adapted to our context and we consider a more realistic evacuation strategy inducing new evacuation distances to shelters. If s is on fire, we have:

- for people on s, two cases have to be considered. If a shelter is located on s, then people present on node s are considered as safely sheltered in it, otherwise we assume that they first run away from the fire in any direction and after they reach a neighbor j, they evacuate to the shortest shelter of j in G^s. The distance from zone s to a shelter k is then the worst case defined by: $\max_{(s,j)\in E^s}(l_{sj} + d^s_{jk})$.
- for people who are not on s, say on $j \neq s$, the evacuation distance from j to shelter k corresponds to d^s_{jk} in graph G^s, i.e., avoiding vertex s.

This evaluation of evacuation distances makes our problem specific compared to the literature and induces some additional complexity. The justification is twofold. First, since the area s may be relatively large, a single scenario may correspond to many possible fire configurations, each prohibiting some paths in the zone. The second motivation is to represent decision under stress, a very important characteristic in emergency management: somebody close to the fire may not take rational decisions when selecting a direction while people in another zone can be assumed to behave more rationally.

For a given solution, corresponding to a location of p shelters on some vertices, and a given scenario s, the evacuation distance of zone j is denoted r^s_j. This evacuation distance remains the shortest distance between j and one of the p shelters but considering the new distances. The radius associated to scenario s, and denoted r^s is defined as $r^s = \max_{j\in V} r^s_j$. A given solution is then valued by n radius $r^1, \ldots, r^n$. In the context of emergency evacuation, the value of a solution is obtained by considering the worst (max) radius. The robust p-center problem is to determine the solution with the minimum worst value.

For example, let us consider the graph G in the figure 1 and the solution corresponding to shelters located on vertices 3 and 11. In this figure, a line between two vertices i and j represents both directed edges (i, j) and (j, i). In case of fire in vertex 2 (scenario 2), the modification of the graph and the evacuation strategy induce:

- The shortest path value from 1 to 3 is no longer 3 because we can't go through the fire in 2, but is now 23, with the shortest path $1, 6, 7, 8, 3$. Consequently, the nearest shelter of vertex 1 is 11 with a distance of 8. Thus the evacuation distance of 1 in scenario 2, is equal to 8 and vertex 1 is evacuated to vertex 11.

- To compute the evacuation distance of vertex 2 in scenario 2, we have to consider three neighbors:
  - for neighbor 1, the distance to the nearest shelter 11 is 1 + 8 =9
  - for neighbor 7, the distance to the nearest shelter 11 is 3 + 9 = 12
  - for neighbor 3 with a shelter, the distance is 2
  Consequently, $r_2^2 = 12$.

- The radius of the scenario 2 is given by $r^2 = \max_{j=1,\dots,15} r_j^2 = 12$.

Finally, the radius of this solution is defined by $r = \max_{s=1,\dots,15} r^s = 20$ induced by scenario 4.

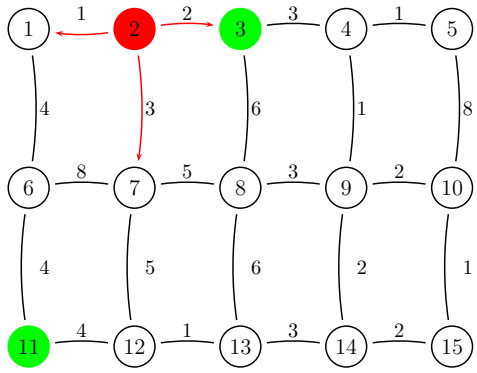

Figure 1: Illustration for the evacuation strategy

In the following section, we propose an integer linear programming formulation for RpCP. This model is inspired by the model proposed in [ELP04, CT13] for the deterministic p-center problem. We generalize this model to take into account different fire scenarios and our evacuation strategy.

3 Mathematical model for the robust p-center problem

We propose a mathematical model with a linear objective function on 0-1 variables representing the maximal radius over all the scenarios to be minimized under linear constraints. The starting point of our model is the deterministic p-center model presented in [CT13]. We recall this model in the next sub-section.

3.1 Mathematical model for the p-center problem

The deterministic p-center model presented in [CT13] is based on the model proposed by Elloumi, Labbé and Pochet in [ELP04]. In [ELP04], the formulation is based on the

observation that the optimal value of the p-center problem corresponds to one of the distances between two vertices in G. We denote by D the finite list of distinct distances between vertices: we first compute the matrix $SP = (d_{ij})$ of the shortest path lengths between every couple of vertices (with $d_{ii} = 0$ and $d_{ij} = +\infty$ if there is no path from i to j); then, D is obtained by sorting in increasing order the T different finite values of the matrix SP: $D_{\min} = D_1 < D_2 < D_3 < \ldots < D_T = D_{\max}$.
In [ELP04] and [CT13], two kinds of binary variables are introduced. More specifically, in [CT13], the following variables are used:

- for all $j = 1, \ldots, n$, y_j is a binary variable with $y_j = 1$ if a center is located on j and 0 otherwise,
- for all $t = 1, \ldots, T$, u_t is a binary variable with $u_t = 1$ if the solution value is equal to D_t and 0 otherwise.

In [CT13], Calik and Tansel introduce the following formulation (P^{det}):

$$(P^{det}) \begin{cases} \min & \displaystyle\sum_{t=1}^{T} D_t u_t & & (1^{det}) \\ s.t. & \displaystyle\sum_{j=1}^{n} y_j = p & & (2^{det}) \\ & \displaystyle\sum_{t=1}^{T} u_t = 1 & & (3^{det}) \\ & \displaystyle\sum_{j:d_{ij} \leq D_t} y_j \geq \sum_{q=1}^{t} u_q & i = 1, \ldots n,\ t = 1, \ldots, T, & (4^{det}) \\ & y_j \in \{0,1\} & j = 1, \ldots, n & \\ & u_t \in \{0,1\} & t = 1, \ldots, T & \end{cases}$$

Constraint (2^{det}) fixes the number of centers to be located. Constraint (3^{det}) ensures that exactly one variable u_t have to be equal to 1 and the corresponding D_t value is selected as the solution value according to the objective function (1^{det}). If $u_t = 1$, then $\sum_{q=1}^{t} u_q = 1$ and constraints (4^{det}) ensure for each vertex i that at least one center is located at a distance less or equal than D_t.

The number of binary variables is equal to $n + T$ and the number of constraints is $nT + 2$. The size of this model can be huge since it depends to the number T of distinct shortest path lengths. But, as explained in [ELP04, CT13], it is possible to reduce this size: knowing a lower bound LB and an upper bound UB for the optimal solution value of (P^{det}), we can delete some variables since:

$$\begin{aligned} u_t &= 0 \quad , \forall t : D_t < LB \\ u_t &= 0 \quad , \forall t : D_t > UB \end{aligned}$$

In the following section, we present an extension of this model to RpCP.

3.2 New model for the robust p-center problem

We extend the formulation of (P^{det}) to RpCP. First, we have to replace D by D^{rob}, the list of distinct finite distance values in all G^s considering the evacuation strategy. The list D^{rob} is obtained by merging and ordering all the ordered sets D^s of distinct finite distances between vertices in G^s, for $s \in V$. The elements of D^s are denoted $D^s_{\min} = D^s_1 < D^s_2 < D^s_3 < \ldots < D^s_{T^s} = D^s_{\max}$. For each s, D^s is computed in two stages:

- step 1: compute $SP^s = (d^s_{ij})$ the matrix of shortest path lengths from i to j in G^s; extract from SP^s the different finite shortest path lengths for all $i \neq s$ and initialize D^s with these values;
- step 2: compute the distance from s to k for all $k \neq s \in V$ according to the worst case value presented in the evacuation strategy: $\max_{(s,j)\in E^s}(l_{sj} + d^s_{jk})$ and add these worst case values to D^s.

Finally, we merge all the lists D^s in one ordered set $D^{rob} = \{D^{rob}_1, \ldots, D^{rob}_{T^{rob}}\}$, with $D^{rob}_{\min} = D^{rob}_1 < D^{rob}_2 < D^{rob}_3 < \ldots < D^{rob}_{T^{rob}} = D^{rob}_{\max}$. In our formulation (P^{rob}), the decision variables are the y variables (similar to (P^{det})) and the u variables with the following interpretation: $u_t = 1$ if and only if, for any given scenario, all the vertices are at a distance to a center less or equal than D^{rob}_t, with $t = 1, \ldots, T^{rob}$. We introduce then the following formulation for RpCP:

$$
(P^{rob}) \begin{cases}
\min \quad \displaystyle\sum_{t=1}^{T^{rob}} D^{rob}_t u_t & & (1)\\
s.c. \quad \displaystyle\sum_{j=1}^{n} y_j = p & & (2)\\
\displaystyle\sum_{t=1}^{T^{rob}} u_t = 1 & & (3)\\
\displaystyle\sum_{j:d^s_{ij}\leq D^{rob}_t} y_j \geq \sum_{q=1}^{t} u_q & s = 1,\ldots,n, i = 1,\ldots,s-1,s+1,\ldots,n, & \\
& t = 1,\ldots,T^{rob} & (4)\\
\displaystyle\sum_{j:l_{sv}+d^s_{vj}\leq D^{rob}_t} y_j \geq \sum_{q=1}^{t} u_q - y_s & s = 1,\ldots,n, \forall v \in N^+(s), t = 1,\ldots,T^{rob} & (5)\\
y_j \in \{0,1\} & j = 1,\ldots,n & \\
u_t \in \{0,1\} & t = 1,\ldots,T^{rob} &
\end{cases}
$$

where $N^+(s)$ is the set of all vertices v such that $(s,v) \in E^s$.
Constraints (2), and (3) are similar to constraints (2^{det}) and (3^{det}) with the only difference that T is replaced by T^{rob}. Constraints (4) ensure for each vertex i that at least one center is located at a distance less or equal than D^{rob}_t in every scenario s. Constraints (5) are specific to RpCP and are necessary to model the chosen evacuation strategy:

- if $y_s = 1$, then a shelter is located on s and constraints (5) are relaxed
- if $y_s = 0$, then no shelter is located on s and the set of constraints (5) on all neighbors of s ensure that the evacuation distance (worst case value) on scenario s is less or equal than D_t^{rob}

The number of binary variables is equal to $n + T^{rob}$ and the number of constraints is $n^2T^{rob} + mT^{rob}$ with m the number of directed edges of G. As for formulation (P^{det}), we can reduce the size of (P^{rob}) if we are able to identify lower and upper bounds for the optimal robust solution value as shown in the following section.

4 Resolution and first experimental results

The size of model (P^{rob}) depends on the size of the list D^{rob}. Similarly to (P^{det}), we can reduce the size of (P^{rob}) knowing a lower bound LB and an upper bound UB for the optimal solution value of (P^{rob}) since we can fix some variables in this way:

$$\begin{aligned} u_t &= 0 \quad ,\forall t : D_t^{rob} < LB \\ u_t &= 0 \quad ,\forall t : D_t^{rob} > UB \end{aligned}$$

4.1 Heuristic bounds

We propose three different methods to compute such bounds:

- The first method uses the optimal solution of (P^{det}). Obviously, the value of the optimal solution of RpCP can not be less than the optimal solution value of (P^{det}). Thus, we denote LB^1 the optimal solution value of (P^{det}). This solution is feasible for RpCP, and its value gives an upper bound UB^1 for RpCP.
- A second method consists in considering (P^{rob}) without the constraints (4). The obtained model (RP^{rob}) corresponds to the problem where only the evacuation distance of the vertex s is taken into account for scenario s. It reduces the number of constraints by n^2T^{rob}. The value of the obtained solution is a lower bound LB^2 for RpCP. Similarly to the first method, the optimal solution of (RP^{rob}) is feasible for RpCP and gives a second upper bound UB^2 for RpCP.
- In a third method, we randomly construct solutions and compute their value for RpCP. The lowest obtained value represents a third upper bound UB^3.

In our preliminary experiments, despite using these bounds for (P^{rob}) the number of constraints and variables were still too high in order to solve exactly the problem, more precisely even to write the PL instance. Thus we propose a general scheme using a generalization of binary search.

4.2 Exact solution method

We propose Algorithm 3 to solve exactly (P^{rob}). This method uses a quantile search algorithm presented below as Algorithm 2, which is a generalization of the binary search algorithm.

Suppose $(P(D))$ a linear programming formulation whose objective value (to be minimized) takes a value from an ordered set $D = \{D_1, D_2, \ldots, D_T\}$ and whose number of variables and constraints depends on T. Denote LB and UB two precomputed lower and upper bounds for $P(D)$. A *σ-quantile search* (implemented in $\sigma - Qsearch$ Algorithm 2) can be used to solve $(P(D))$ by solving at most $\lceil \log_\sigma(T+1) \rceil$ instances of $(P(D'))$:

- First we compute a restricted set $D' \subseteq D$ with function $Fkernel$ (Algorithm 1), where the values of D are selected such that the set of values of D between UB and LB are evenly spread between the values of D'.
- Next we solve $(P(D'))$ which gives us a new upper and lower bound for $(P(D))$

Note that for $\sigma = 3$, the σ-quantile search is actually a binary search.

Algorithm 1 *$Fkernel$*

Require: $D = \{D_1, D_2, \ldots, D_T\}, LB, UB, \sigma \in \mathbb{N}^+$

Ensure: Return a subset of D with LB, UB and the values of the $(\sigma - 2)$-quantiles between them.

1: Find $k_1 \in \{1, \ldots, T\}$ such that $D_{k_1} = LB$
2: Find $k_\sigma \in \{1, \ldots, T\}$ such that $D_{k_\sigma} = UB$
3: $step \leftarrow \lfloor (k_1 + k_\sigma)/(\sigma - 1) \rfloor$
4: **for** $i \leftarrow 2$ **to** $\sigma - 1$ **do**
5: $\quad k_i = k_{i-1} + step$
6: **end for**
7: Return $\{D_{k_1}, \ldots, D_{k_i}, \ldots, D_{k_\sigma}\}$

Algorithm 2 is also used to solve (RP^{rob}) in the exact solution method in Algorithm 3. More precisely we use LB^1 to initialize the σ-search for LB^2.

In the next section, we evaluate the computational merits of the proposed algorithm using test instances from the OR-Library.

4.3 First experimental results

The input data used for the computations is a sample of 28 instances of the p-center problem from the OR-Library ([Bea90]) with n varying between 100 and 600 and p varying between 5 and $(n/3)$. The original data in the OR-Library consists of a listing of edges and their lengths. We implement the Exact Algorithm in Python 3.5, and we generate the distance matrices d_{ij} and d^s_{ij} for all scenario $s \in V$ using networkx library 2.2. Using these matrices, we can compute the list $D = \{D_1, \ldots, D_T\}$ for (P^{det}), and the list D^{rob} for

Algorithm 2 $\sigma - Qsearch$

Require: $P(D), LB, UB, \sigma \in \mathbb{N}^+$

Ensure: Return the optimal value and an optimal solution to $P(D)$

while $UB \neq LB$ **do**
  $D' = \{D_{k_1}, \ldots, D_{k_\sigma}\} \leftarrow Fkernel(D, LB, UB, \sigma)$
  Solve $P(D')$
  Set $q \in \mathbb{N}^+$ such that $D_{k_q} \leftarrow$ Optimal value of $P(D')$
  $sol \leftarrow$ Optimal solution of $P(D')$
  if $q = 1$ **then**
    $UB \leftarrow LB$
  else
    $UB \leftarrow D_{k_q}$
    $LB \leftarrow D_{k_{q-1}}$
  end if
end while
Return LB and sol

Algorithm 3 Exact Algorithm

Require: $G = (V, E), p, x, \sigma_1, \sigma_2$

Begin
Generate D^{rob}
Solve (P^{det}) and generate LB^1 and UB^1
Generate UB^3 by constructing x random solutions
$LB^2, sol \leftarrow \sigma - Qsearch((RP^{rob}(D^{rob})), LB^1, \min\{UB^1, UB^3\}, \sigma_1)$
$UB^2 \leftarrow$ Value of sol for (P^{rob})
$UB \leftarrow \min\{UB^1, UB^2, UB^3\}$
$optValue, optSolution \leftarrow \sigma - Qsearch((P^{rob}(D^{rob})), LB^2, UB, \sigma_2)$
Return $optValue$
End

(P^{rob}). We execute our experiments on a server with 254Gb of RAM and 14 Intel Core (Haswell; no TSX) Processor at 2.3 Ghz. Mathematical models are solved with CPEX 12 (with MIPEmphasis option set to 0).All the results are presented in details in tables in the appendix A. In the following, we present our synthetic analysis of these results.

We solve (P^{rob}) using the Exact Algorithm with $\sigma_1 = \sigma_2 = 3$ and $x = 10$.

First, we observe that LB^2 corresponds to the value of the optimal solution on almost all considered instances.

Figure 2 shows the value of the upper bounds computed for the instances ordered by the number of their vertices and by the value of p (see for details Table 1 in appendix).

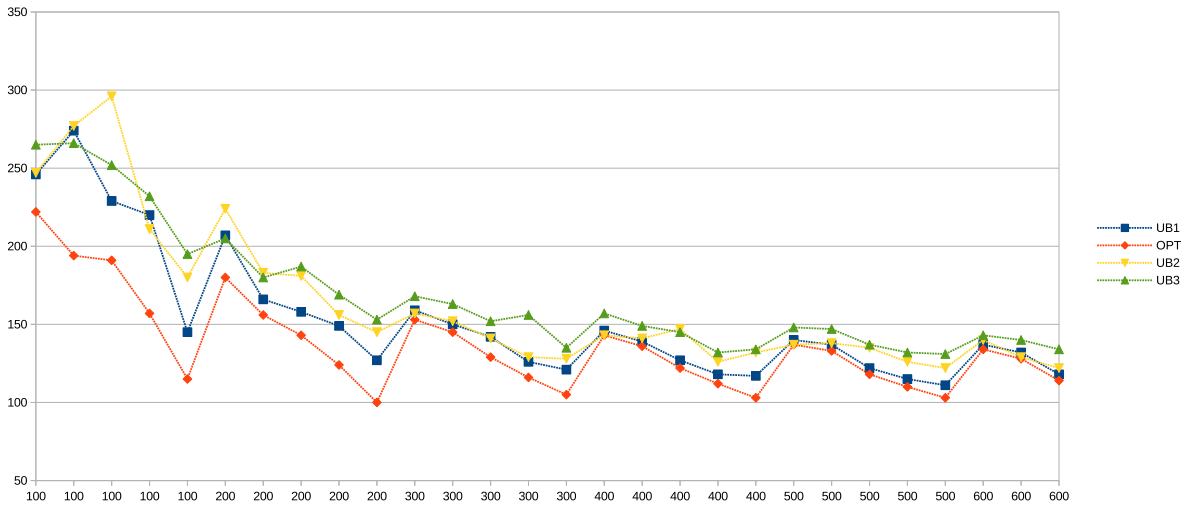

Figure 2: Evolution of the computed bounds given the size of the instance

The up and down movements of the lines for a same value of n is due to the different values of p. We observe that UB^1 often gives the best upper bound.

Given the quality of the lower bound obtained, we observe that increasing σ_2 to values greater than 3 was counterproductive: indeed, one iteration in $\sigma - Qsearch$ is enough to prove that the lower bound is an upper bound. In this case, increasing σ_2 would only lead to an increased size of the LP model to construct in the step 8 of Exact Algorithm. However, we still need to try different values for σ_1.

In Figure 3 (see for details Table 2 in appendix), the red curve represents the total processing time of the Exact Algorithm on the instances. We observe that we can solve even the biggest instances (with $n = 800$ and $m = 7200$ and $p = 60$) in less than 2 hours. In order to explain these computational time, we show in Figure 3 the percentage of the computation time taken by:

- The time to generate D^{rob} (line 2 of Exact Algorithm) denoted by T Init
- The time to solve (P^{det}) (line 3) denoted by T Det
- The time to solve UB^3 (line 4) denoted by T UB3
- The time to solve ($RP^{rob}(D^{rob})$) (line 5) denoted by T RProb
- The time to solve (P^{rob}) (line 8) denoted by T Exact

Observe that the initialization of the instance is the main time consuming process. These preliminary results show that our $\sigma - Qsearch$ process is specifically efficient to solve both (P^{rob}) and (RP^{rob}). Note also that the proportion of T Exact to the total

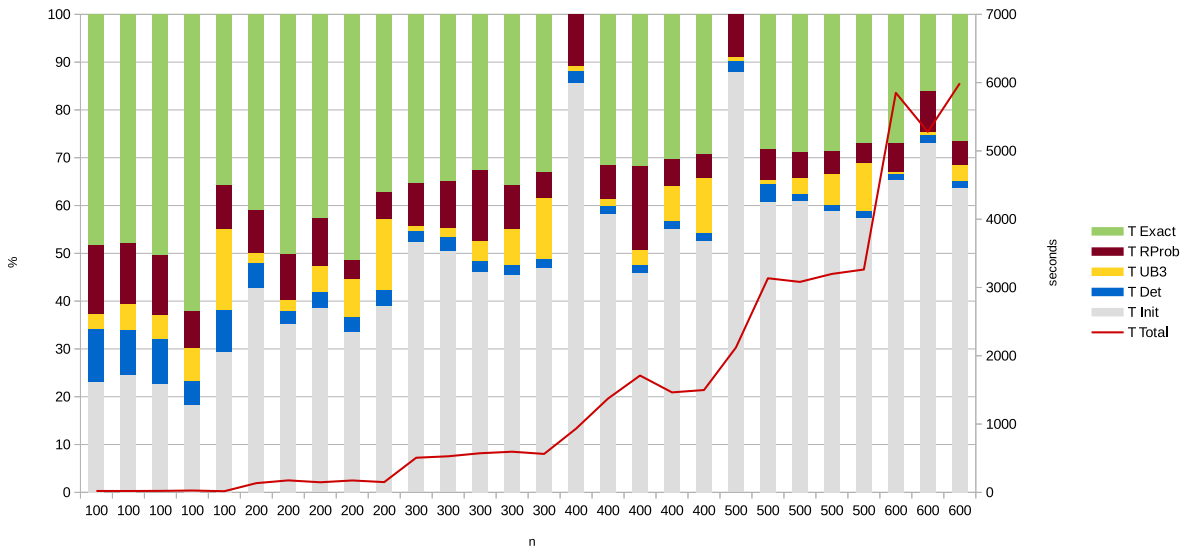

Figure 3: Evolution of the processing time given the size of the instance

processing time remains constant. Moreover, T Exact is equal to zero when $UB = LB^2$, it occurs twice in our instances. The time to solve (P^{det}) is negligible, which highlights the increase of complexity between the deterministic and our robust version.

In addition, we examine the impact of p on the overall computing time by solving some particular instances for p ranging from 2 to $n/3$. We give the results for instance *pmed*4 with 100 vertices and 200 edges, which seems to us typical of the observed trend. Figure 4 (see for details Table 3 in appendix) shows the time performance of the different part of the Exact Algorithm on instance *pmed*4 in relation to p. It also indicates the number of iterations performed inside step 8 of the Exact Algorithm. The number of iterations exceeds 1 when LB^2 is not equal to the optimal solution value, which happens very sparingly as we have seen in the previous results. We can see that the overall processing time does not increase regarding the size of p, with the exception of the cases where step 8 involves more than one iteration.

This integer linear programming approach can be adapted to solve some variants that are interesting from a practical perspective. All these variants have been discussed with practitioners and have been identified as relevant for some regions.

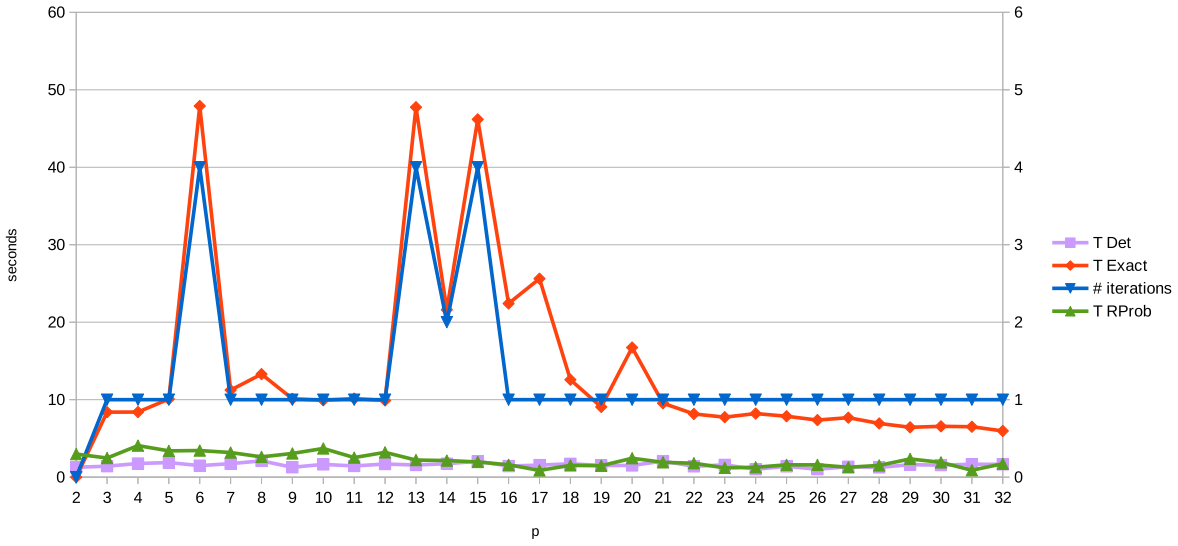

Figure 4: pmed4 with 100 vertices

5 Some extensions and variants of the robust p-center problem

5.1 Concerning shelters

The first variant is motivated by the fact that, under some circumstances, a shelter can be located only in some specific places. We denote Z the subset of vertices on which it is possible to locate a shelter. This variant is easy to model as follows: $\forall j \in V \setminus Z$, $y_j = 0$ since we consider only y variables associated to vertices that are suitable for shelters belonging to Z.

5.2 Concerning scenario

In the second variant, a scenario s is no longer a single burning vertex but a subset V^s of vertices in fire. This variant enables us to take into account the fire spread (in a static way). As previously, if a vertex v is in V^s, this vertex is no longer reachable. Given a subset V^s, the operational graph $G^s = (V, E^s)$ is deduced from the initial graph G by removing all incoming arcs of the form (j, v) for all $v \in V^s$. We have $E^s = E \setminus \cup_{v \in V^s} \omega^-(v)$. Consequently, in G^s the subset V^s is an independent set without any incoming arcs from $V \setminus V^s$. In this static context, it is quite natural to assume that for all scenarios s and s', we have $V^s \cap V^{s'} = \emptyset$ since V^s represents a fire ignition and its spread. Indeed, if a vertex is common to two distinct scenarios s and s' all vertices of s and all vertices of s'

belong to the same unique scenario. Our model can also easily integrate this variant:

$$
(P_2^{rob}) \begin{cases}
\min & \displaystyle\sum_{t=1}^{T^{rob}} D_t^{rob} u_t & & (1)\\
s.c. & \displaystyle\sum_{j\in Z} y_j = p & & (2)\\
& \displaystyle\sum_{t=1}^{T^{rob}} u_t = 1 & & (3)\\
& \displaystyle\sum_{j\in Z: d_{ij}^s \le D_t^{rob}} y_j \ge \sum_{q=1}^{t} u_q & \forall s\in S, \forall i \notin V^s,\ t=1,\dots,T^{rob} & (4)\\
& \displaystyle\sum_{j\in Z: l_{iv}+d_{vj}^s \le D_t^{rob}} y_j \ge \sum_{q=1}^{t} u_q - y_i & \forall s\in S, \forall i\in V^s, \forall v\in N_{G^s}^+(i), & \\
& & t=1,\dots,T^{rob} & (5)\\
& y_j = 0 & \forall j \in V\setminus Z & (6)\\
& y_j \in \{0,1\} & j=1,\dots,n & \\
& u_t \in \{0,1\} & t=1,\dots,T^{rob} &
\end{cases}
$$

where S represents the set of scenarios and $N_{G^s}^+(i)$ is the set of all vertices v such that $(i,v)\in E^s$.

5.3 Concerning evacuation strategy

In the third variant, only the vertices associated to a scenario must be evacuated: if a fire occurs in a zone, no need to evacuate other zones. If we consider a more dynamic context, it could be interesting to assume that we need to evacuate only people where fire ignites and we have to locate shelters in order to manage this emergency first step. This variant only requires to remove constraints (4) from models (P^{rob}) or (P_2^{rob}).

5.4 Concerning objective value

Finally, we propose an alternative model considering the probability π_s that scenario s happens. In this case, the goal is to minimize the expectation value of radius on all scenarios and we obtain the Probabilistic p-Center Problem (PpCP) [DHM18]. For this version, as we need to compute the value of the radius of scenario s, we have to introduce variables u_t^s per scenario, where $u_t^s = 1$ if and only if all vertices are at a distance of a shelter less or equal than D_t^s in scenario s for $t = 1,\dots,T^s$. The mathematical model

denoted (P_2^{pro}) is:

$$
(P_2^{pro}) \begin{cases}
\min & \displaystyle\sum_{s\in S} \pi_s \sum_{t=1}^{T^s} D_t^s u_t^s & & (1)\\
s.c. & \displaystyle\sum_{j\in Z} y_j = p & & (2)\\
& \displaystyle\sum_{t=1}^{T^s} u_t^s = 1 & \forall s \in S & (3)\\
& \displaystyle\sum_{j\in Z: d_{ij}^s \le D_t^s} y_j \ge \sum_{q=1}^{t} u_q^s & \forall s \in S, \forall i \notin V^s,\ t = 1,\dots,T^s & (4)\\
& \displaystyle\sum_{j\in Z: l_{iv}+d_{vj}^s \le D_t^s} y_j \ge \sum_{q=1}^{t} u_q^s - y_i & \forall s \in S, \forall i \in V^s, \forall v \in N_{G^s}^+(i), & \\
& & t = 1,\dots,T^s & (5)\\
& y_j = 0 & \forall j \in V \setminus Z & (6)\\
& y_j \in \{0,1\} & j = 1,\dots,n & \\
& u_t^s \in \{0,1\} & \forall s \in S,\ t = 1,\dots,T^s &
\end{cases}
$$

Thus, it appears that our model is able to take into account several interesting variants. In the following section, we present a particular realistic case and we show its polynomial complexity.

6 One particular realistic case

Let us consider a particular case motivated by a real case in Spain. It concerns a touristic valley where the evacuation graph is a caterpillar. We recall that a caterpillar is a tree for which removing the leaves and incident edges produces a path, called the spin. In our context, the spin corresponds to the valley and the leaves are associated to touristic areas in the mountain. Shelters can be located only in the valley and the considered set of scenarios include only subsets of leaves.

See for example the figure 5 with a spin of length 9 and 13 leaves. The set Z is equal to $\{1,\dots,9\}$ and five scenarios are defined:

- scenario 1: $V^1 = \{10, 11, 12\}$
- scenario 2: $V^2 = \{14, 15\}$
- scenario 3: $V^3 = \{16, 18\}$
- scenario 4: $V^4 = \{19, 20, 21\}$
- scenario 5: $V^5 = \{22\}$

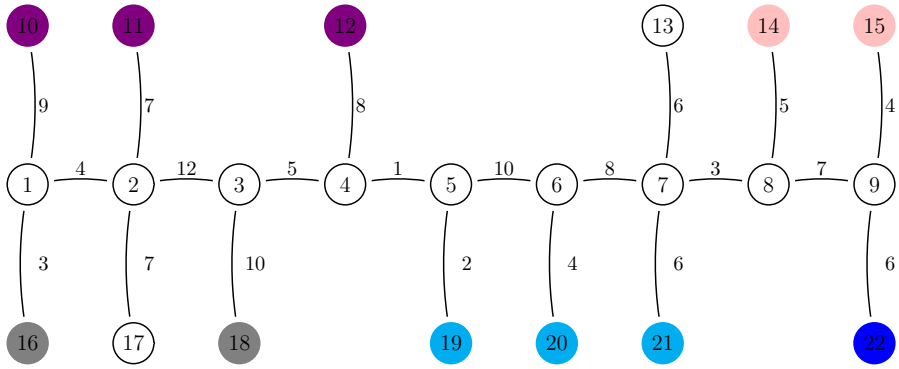

Figure 5: A caterpillar example

If a fire occurs on some vertices of a scenario, everybody must be evacuated. The optimal solution of this problem with $p = 2$ is to locate shelters on vertices 2 and 6. Its value is 24, coming from the vertex 22. In fact, this solution is also optimal for the deterministic p-center problem as showed in the following proposition.

Let us denote $v(P)$ the optimal value of a mathematical problem (P).

Proposition 1 *In a connected graph G, if the two following conditions are satisfied:*

- *(C1): for each $s \in S$, V^s only includes vertices with exactly one neighbor*
- *(C2): $Z \cap (\cup_{s \in S} V^s) = \emptyset$*

then $v(P^{det}) = v(P_2^{rob})$.

Proof 1 *Let us remark that for all v in V^s, $\omega^-(v) = \{(x_v, v)\}$ where x_v is the unique neighbor of vertex v and G^s is obtained from G by removing $|V^s|$ directed edges, so $G^s = (V, E \setminus \cup_{v \in V^s}(x_v, v))$. Then, all shortest paths are equivalent in G^s and in G excepted those ending in v for each $v \in V^s$:*

$$d_{ij}^s = \begin{cases} d_{ij} & \forall s \in S,\ \forall i \in V,\ \forall j \notin V^s \\ +\infty & \forall j \in V^s \end{cases}$$

We first show that $D = D^{rob}$. For any $d_{ij} \in D$, two cases have to be considered:

- *if $j \notin \cup_{s \in S} V^s$, then $d_{ij} \in D^s$ since $d_{ij}^s = d_{ij}$*
- *if $j \in \cup_{s \in S} V^s$, then it exists s' such that $d_{ij} \in D^{s'}$ with $j \notin V^{s'}$,*

Thus $D \subseteq \cup_{s \in S} D^s$.
In each D^s, we have to add (see Step 2 in section 3.2) the worst case values induced by the strategy evacuation: $\max_{(v,j) \in E^s}(l_{vj} + d_{jk}^s)$ for all v in V^s. Since v has only one neighbor

x_v, we have: $\max_{(v,j)\in E^s}(l_{vj} + d^s_{jk}) = l_{vx_v} + d^s_{x_vk} = d^s_{vk}$ *and this value is already in* D^s. *Consequently,* $D = D^{rob}$.
Next, we show that the set of constraints of (P_2^{rob}) *and* (P^{det}) *define the same set of feasible solutions. Due to condition (C2) and constraints* (6), *we have* $y_i = 0$ *in constraints* (5) *which become:*

$$\sum_{j\in Z: d^s_{ij}\leq D^{rob}_t} y_j \geq \sum_{q=1}^{t} u_q \;\; \forall s \in S,\; \forall i \in V^s,\; t = 1, \ldots, T^{rob}$$

Then constraints (4) *and* (5) *in* (P_2^{rob}) *can be rewritten in this way:*

$$\sum_{j\in Z: d^s_{ij}\leq D^{rob}_t} y_j \geq \sum_{q=1}^{t} u_q \;\; \forall s \in S,\; \forall i \in V,\; t = 1, \ldots, T^{rob}$$

Since $D^{rob} = D$ *and* $d^s_{ij} = d_{ij}$ *for finite values, this set of constraints defines exactly the same set of feasible solutions than constraints* (4^{det}). *Consequently, the optimal solutions for* (P^{det}) *and* (P_2^{rob}) *are the same.*

As the p-center problem is polynomial for trees [KH79], we obtain the following result.

Corollary 1 *In a tree G, if the two conditions (C1) and (C2) are satisfied then RpCP is polynomial.*

7 Conclusion

We introduce a new version of the p-center problem appropriate to the specific context of evacuation in case of wild fires. The problem can be modeled with an integer linear program. The mathematical programming approach enables us to extend our model to other fire configurations and to solve reasonably large instances of our model. The results we have obtained open many questions and research tracks. So, they need to be considered as preliminary results that essentially validate this problem and illustrate its richness and potential. Further research directions will include the design of more efficient heuristics and the study of relevant classes of topologies for real case applications. These classes of instances motivate two main challenges: analyzing the problems and their difficulty in such particular cases and how to take into account specific structures of real case instances in the linear programming approaches so as to make the related models much more tractable.

References

[AB97] Igor Averbakh and Oded Berman. Minimax regret p-center location on a network with demand uncertainty. *Location Science*, 5(4):247–254, 1997.

[AB00] Igor Averbakh and Oded Berman. Algorithms for the robust 1-center problem on a tree. *European Journal of Operational Research*, 123(2):292–302, 2000.

[Ave03] Igor Averbakh. Complexity of robust single facility location problems on networks with uncertain edge lengths. *Discrete Applied Mathematics*, 127(3):505–522, 2003.

[Bea90] John E Beasley. Or-library: distributing test problems by electronic mail. *Journal of the operational research society*, 41(11):1069–1072, 1990.

[CT13] H. Calik and B.C. Tansel. Double bound method for solving the p-center location problem. *Computers and Operations Research*, 40:2991–2999, 2013.

[DHM18] Marc Demange, Marcel Adonis Haddad, and Cécile Murat. The probabilistic k-center problem. In *Proceedings of the GEOSAFE Workshop on Robust Solutions for Fire Fighting*, pages 62–74, L'Aquila, Italy, 2018.

[ELP04] Sourour Elloumi, Martine Labbé, and Yves Pochet. A new formulation and resolution method for the p-center problem. *INFORMS Journal on Computing*, 16(1):84–94, 2004.

[KH79] Oded Kariv and S Louis Hakimi. An algorithmic approach to network location problems. i: The p-centers. *SIAM Journal on Applied Mathematics*, 37(3):513–538, 1979.

A Table of Results

n	m	p	OPT	LB1	UB1	LB2	UB2	UB3
100	200	5	222	127	246	222	247	265
100	200	10	194	98	274	194	277	266
100	200	10	191	93	229	191	296	252
100	200	20	157	74	220	157	211	232
100	200	33	115	48	145	115	180	195
200	800	5	180	84	207	180	224	205
200	800	10	156	64	166	156	183	180
200	800	20	143	55	158	143	181	187
200	800	40	124	37	149	124	156	169
200	800	67	100	20	127	100	145	153
300	1800	5	153	59	159	153	157	168
300	1800	10	145	51	150	145	152	163
300	1800	30	129	36	142	129	141	152
300	1800	60	116	26	126	116	129	156
300	1800	100	105	18	121	105	128	135
400	3200	5	143	47	146	143	143	157
400	3200	10	136	39	139	136	141	149
400	3200	40	122	28	127	122	147	145
400	3200	80	112	18	118	112	126	132
400	3200	133	103	13	117	103	132	134
500	5000	5	137	40	140	137	137	148
500	5000	10	133	38	137	133	138	147
500	5000	50	118	22	122	118	135	137
500	5000	100	110	15	115	110	126	132
500	5000	167	103	11	111	103	122	131
600	7200	5	134	38	137	134	140	143
600	7200	10	128	32	132	128	129	140
600	7200	60	114	18	118	114	122	134

Table 1: Value of the bounds and of the optimal robust solution computed with the Exact Algorithm for OR Library instances

n	m	p	T Init	T Det	T UB3	T RProb	T Exact	T Total
100	200	5	4.8	2.2	0.6	3.0	9.9	20.5
100	200	10	4.8	1.8	1.0	2.5	9.3	19.6
100	200	10	4.8	2.0	1.0	2.7	10.7	21.3
100	200	20	4.9	1.4	1.8	2.1	16.6	26.8
100	200	33	4.9	1.5	2.8	1.5	6.0	16.7
200	800	5	57.6	7.0	2.7	12.4	55.0	134.7
200	800	10	61.8	4.5	4.3	16.8	87.8	175.3
200	800	20	57.2	5.1	7.8	14.9	63.0	147.9
200	800	40	58.4	5.6	13.9	6.9	89.4	174.1
200	800	67	58.6	5.1	22.0	8.4	55.8	149.9
300	1800	5	266.1	10.8	6.1	44.9	179.2	507.0
300	1800	10	267.1	15.3	10.1	51.7	183.9	528.1
300	1800	30	264.3	12.6	24.7	84.4	186.8	572.8
300	1800	60	270.9	12.2	45.4	54.1	212.4	595.0
300	1800	100	264.6	10.2	72.9	29.6	185.6	562.8
400	3200	5	798.6	23.3	10.9	100.2	0.0	933.0
400	3200	10	800.4	24.1	18.5	97.4	433.1	1373.6
400	3200	40	784.5	28.2	57.2	298.4	542.3	1710.6
400	3200	80	806.6	23.6	106.7	85.3	441.5	1463.7
400	3200	133	790.0	23.0	172.7	74.1	438.4	1498.1
500	5000	5	1867.0	45.9	17.1	190.9	0.0	2120.9
500	5000	10	1906.9	115.9	28.9	199.3	884.4	3135.3
500	5000	50	1878.3	44.6	109.0	163.0	886.5	3081.3
500	5000	100	1883.6	40.1	205.8	157.1	912.1	3198.8
500	5000	167	1875.7	44.4	332.9	132.3	878.3	3263.6
600	7200	5	3821.9	77.9	25.2	348.3	1577.4	5850.6
600	7200	10	3859.8	85.0	41.3	448.6	845.4	5280.2
600	7200	60	3816.7	96.3	189.8	303.3	1584.5	5990.6

Table 2: Computing time of the different parts of the Exact Algorithm for OR Library instances

p	T Det	T RProb	T Exact	Nb. Iterations
2	1.3	3.0	0.0	0
3	1.4	2.5	8.4	1
4	1.8	4.1	8.4	1
5	1.9	3.4	10.1	1
6	1.5	3.4	47.9	4
7	1.7	3.2	11.2	1
8	2.1	2.6	13.3	1
9	1.3	3.1	10.1	1
10	1.6	3.7	9.9	1
11	1.4	2.5	10.1	1
12	1.7	3.2	9.9	1
13	1.6	2.2	47.8	4
14	1.7	2.1	21.6	2
15	2.0	2.0	46.2	4
16	1.4	1.6	22.4	1
17	1.6	0.9	25.6	1
18	1.7	1.5	12.6	1
19	1.5	1.5	9.1	1
20	1.5	2.4	16.7	1
21	2.1	1.9	9.5	1
22	1.4	1.8	8.2	1
23	1.6	1.2	7.7	1
24	1.0	1.3	8.2	1
25	1.4	1.6	7.9	1
26	1.0	1.6	7.4	1
27	1.3	1.3	7.7	1
28	1.3	1.5	6.9	1
29	1.6	2.3	6.4	1
30	1.6	1.9	6.6	1
31	1.7	0.9	6.5	1
32	1.7	1.7	6.0	1

Table 3: Computing time of (P^{det}), (RP^{rob}) and (P^{rob}) in the Exact Algorithm for instance pmed4, and the number of iterations done for solving (P^{rob})