Université Libre de Bruxelles

INFO-F-424 Combinatorial Optimization

The p-Center Problem #1

Antoine Passemiers
Cédric Simar

April 29, 2018

Table of contents

1 Introduction

For this project, we chose to implement the problem described in the statement entitled "The p-Center Problem #1", i.e. the vertex restricted p-center problem, an optimization problem that requires the location of p centers on the vertices of a given network and the allocation of the nodes to the selected centers in order to minimize the distance between the nodes and their assigned centers. In the case of tree networks, low order polynomial time algorithms exist to solve the p-center problem. However, for general networks, the problem is NP-hard.

The formal definition of the problem is provided in the project statement as follows: "Let $G = (N, E)$ be a given network with vertex set $N = \{1, ..., n\}$ and edge set E. Define d_{ij} as the length of a shortest path from vertex $i \in N$ to vertex $j \in N$ in the given network and $f(X) = \max_{i \in N} \min_{x \in X} d_{xi}$ for any point set $X \subset G$. Then, the vertex restricted p-center problem is to find a set $X^* \subseteq N$ with $|X^*| = p$ so that $f(X^*) \leq f(X)$ for any $X \subseteq N$ with $|X| = p$".

In order to solve the vertex restricted p-center problem, this work implements two integer programming (IP) formulations, i.e. a first formulation (P1) proposed by Daskin (1995) and a second formulation (P3) proposed by Calik and Tansel (2013).

In this report, we will first describe both P1 and P3 mathematical formulations, our computational experiments as well as discuss the results obtained using two different solvers interfaced by the JuMP modeling language.

2 Mathematical Formulations

2.1 P1

The P1 formulation to solve the vertex restricted p-center problem was proposed by Daskin (1995).

2.1.1 Constants

Let:

- $G = (N, E)$ be a given network
- $N = \{1, ..., n\}$ be the vertex set of G
- E be the edge set of G (not used in the vertex restricted p-center problem)
- $d_{ij} \in \mathbb{N}^+$ be the length of a shortest path in G from vertex $i \in N$ to vertex $j \in N$
- $p \in \mathbb{N}^+$ be the maximum number of centers

2.1.2 Variables

Let:

- $x_{ij} \in \{0, 1\}$ be 1 if vertex i assigns to a center placed at vertex j, 0 otherwise
- $y_j \in \{0, 1\}$ be 1 if a center is placed at vertex j, 0 otherwise

2.1.3 Objective function and constraints

The objective is to minimize

$$z$$

with respect to the variables x_{ij} and y_j under the following constraints:

$\sum_{j \in N} d_{ij} x_{ij} \leq z$	$\forall i \in N$	ensures that the objective value is no less than the maximum vertex-to-center distance	(1)
$\sum_{j \in N} x_{ij} = 1$	$\forall i \in N$	assigns each vertex to exactly one center	(2)
$x_{ij} \leq y_j$	$\forall i, j \in N$	ensures that no vertex assigns to vertex j unless there is a center at vertex j	(3)
$\sum_{j \in N} y_j \leq p$		restricts the number of centers to at most p	(4)
$y_j \in \{0, 1\}$	$\forall j \in N$	binary restriction	(5)
$x_{ij} \in \{0, 1\}$	$\forall i, j \in N$	binary restriction	(6)

2.2 P3

The P3 formulation to solve the p-center problem was proposed by Calik and Tansel (2013). P3 is based on a previous Integer Programming formulation (P2) by Elloumi et al. (2004). We note that P3 extends P1 formulation to p-FC, i.e. the F-restricted p-center problem with F being any finite subset of G, by replacing "$j \in N$" by "$j \in M$". P3 used a set-covering based approach that takes advantage of the fact that the distances d_{ij} are the only possible values for $r_p(F)$, i.e. the optimal p-radii of the F-restricted p-centers. This approach led to reduce the number of constraints from 6 (P1) to 5 (P2 and P3), thus improving performances.

2.2.1 Constants

Let:

- $G = (N, E)$ be a given network
- E be the edge set of G (not used in the F-restricted p-center problem)
- F be any finite subset of G
- $N = \{1, ..., n\}$ be the vertex set of G
- $M = \{1, ..., m\}$ be the vertex set of F (in P1, $m = n$)
- f_j be an enumeration of the points in F, $j \in M$
- $d_{ij} \in \mathbb{N}^+$ be the length of a shortest path in G from vertex $i \in N$ to vertex $j \in M$
- $p \in \mathbb{N}^+$ be the maximum number of centers
- $\rho_1 < \rho_2 < ... < \rho_K$ be an ordering of the K distinct distance values of the d_{ij} matrix
- $R = \{\rho_1, \rho_2, ..., \rho_K\}$ be the set of ordered ρ values
- $k \in T \equiv \{1, ..., K\}$ be an index of a ρ value
- $a_{ijk} \in \{0, 1\}$ be 1 if $d_{ij} \leq \rho_k$, 0 otherwise

2.2.2 Variables

Let:

- $z_k \in \{0, 1\}$ be 1 if $r_p(F) = \rho_k$, 0 otherwise
- $y_j \in \{0, 1\}$ be 1 if a center is placed at site f_j, 0 otherwise

2.2.3 Objective function and constraints

The objective is to minimize

$$\sum_{k \in T} \rho_k z_k$$

with respect to the variables z_k and y_j under the following constraints:

$\sum_{j \in M} a_{ijk} y_j \geq z_k$	$\forall i \in N, \forall k \in T$	ensures that each vertex is covered within the selected radius by at least one center	(7)
$\sum_{j \in M} y_j \leq p$		restricts the number of centers to at most p	(8)
$\sum_{k \in T} z_k = 1$		ensures that exactly one of the variables z_k is selected as 1 while the objective function determines the value $r_p(F)$ as the corresponding value ρ_k	(9)
$y_j \in \{0,1\}$	$\forall j \in M$	binary restriction	(10)
$z_k \in \{0,1\}$	$\forall k \in T$	binary restriction	(11)

3 Implementation

The project was implemented exclusively in Julia v0.6.2, a high-level dynamic programming language as requested in the assignment guidelines. In addition to the requested default use of the open-source Coin-OR Branch-and-Cut solver (Cbc), our implementation also offers the possibility to compare the results and performance of Cbc with the GNU Linear Programming Kit (GLPK), another solver interfaced by JuMP.

3.1 Project Structure

The source folder of the project is structured as follows:

- the `instances` folder contains:
  - a folder `easy` containing the easy instances provided with the assignment
  - a folder `hard` containing the hard instances provided with the assignment
- the `report` folder contains the report in pdf and LaTeX formats
- the `results` folder contains the results (objective and elapsed time) of the program execution on the selected instances
- the `src` folder contains the source files in Julia

3.2 Execution

3.2.1 Extra modules

In order to properly run the project, several modules have first to be installed using the following commands in a Julia environment:

- `Pkg.add("ArgParse")`
- `Pkg.add("Cbc")`
- `Pkg.add("GLPKMathProgInterface")`
- `Pkg.add("JuMP")`

3.2.2 Arguments

The project can be executed using two different scripts:

- `julia test_all.jl` is used without argument, runs the two Cbc and GLPK solvers on all instances and writes the minimized objective function value and performance result of each instance in respective files in the `results` folder.
- `julia solve.jl` is used with the following arguments:
  - path to the instance file (required)

- formulation of the p-center problem (either `p1` or `p3`) (required)
- solver (either `cbc` or `glpk`) (optional, using `--solver`)

For example: the following command

```
julia solve.jl ./instances/easy/instance10_1_1.dat p1 --solver glpk
```

runs the `solve.jl` script to solve a unique instance `instance10_1_1.dat` using the GLPK solver with the P1 formulation.

4 Additional improvements

Since we observed that the computation time to solve hard instances significantly increased compared to easy instances, we decided to implement the improvements described in chapters 4 and 5 of Calik and Tansel (2013), i.e. the "BINARY" and "Double bound" algorithms.

Let:

- LPx denote the relaxation of Px
- val(LPx) denote the optimal value of Px.
- RPx denote the relaxation that retain the objective function and all constraints of Px

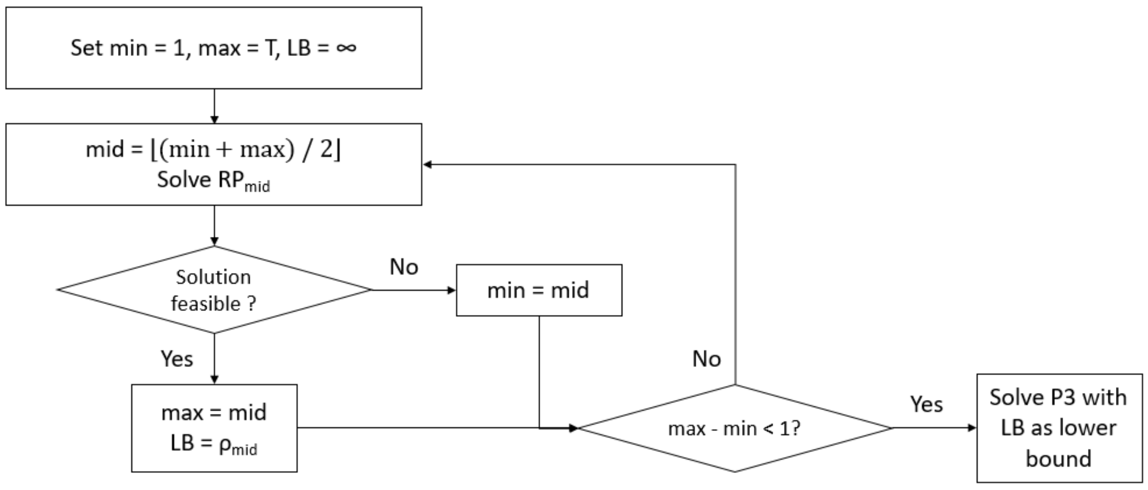
4.1 BINARY algorithm

Let:

- P_h with $h \in T$ be the problem P3 with the additional constraint $z_h = 1$
- RP_h with $h \in T$ be the problem RP3 with the additional constraint $z_h = 1$
- val(P_h) be the optimal value of P_h
- val(RP_h) be the optimal value of RP_h

Calik and Tansel (2013) proposes and proves that val($RP3$)= $\min_{h \in T}$ val(RP_h). They show that computing $\min_{h \in T}$ val(RP_h) is achieved in polynomial time by solving $O(\log_2 K)$ linear programs RP_h for each ρ_h selected from the set of ordered ρ values $R = \{\rho_1, \rho_2, ..., \rho_K\}$ (as defined in P3) during a binary seach.

The BINARY algorithm proposed by Calik and Tansel (2013) can be schematized as follows:



4.2 Double bound algorithm

Let:

- S be any nonempty subset of $T = \{1, ..., K\}$
- $P(S)$ be the problem which is exactly the same as P3 except that all variables $z_k, k \in T$
  S are dropped from P3

Calik and Tansel (2013) proposes and proves the following:

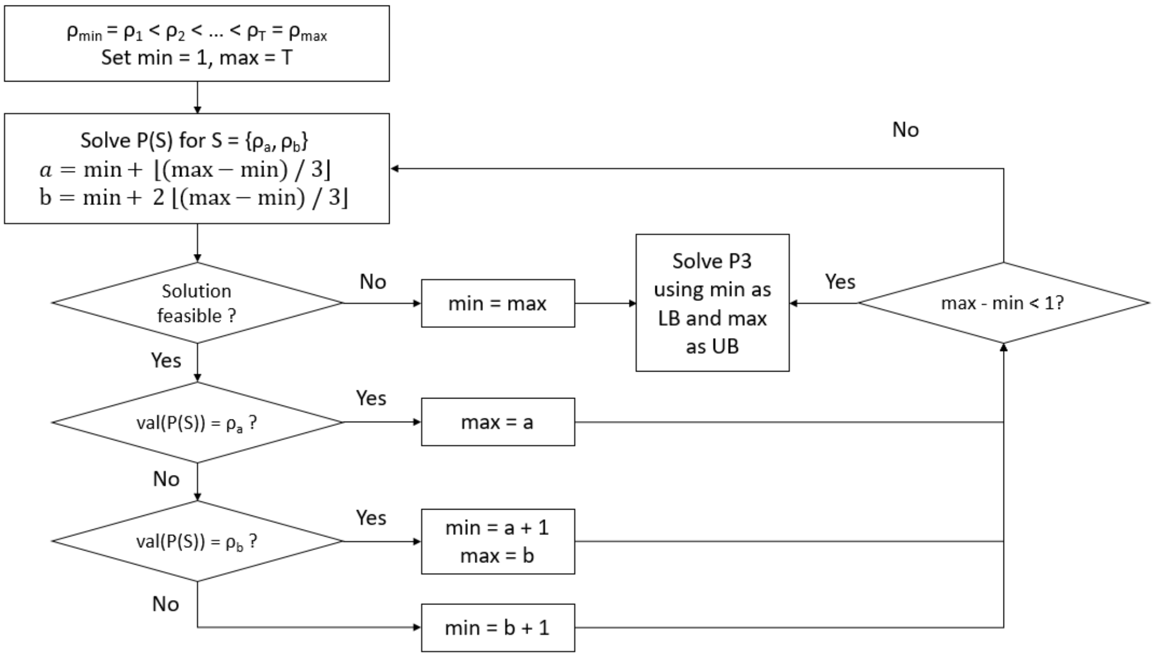
"Suppose $|S| > 2$. Let a and b be the smallest ad largest indices in S, respectively.

(a) If val($P(S)$) = ρ_a, then $r_p(F) \in \{\rho_1, ..., \rho_a\}$

(b) If val($P(S)$) = ρ_k for some k with $a < k \leq b$, then $r_p(F) \in \{\rho_{k'+1}, ..., \rho_k\}$ where k' is the largest index in S which is smaller than k

(c) If val($P(S)$) = ∞, then $r_p(F) \in \{\rho_{b+1}, ..., \rho_K\}$"

This proposition allows the construction of a more efficient search strategy based on the restriction of P3. The Double bound algorithm proposed by Calik and Tansel (2013) is based on this proposition and can be schematized as follows:


$\rho_{min} = \rho_1 < \rho_2 < ... < \rho_T = \rho_{max}$
Set min = 1, max = T
Solve P(S) for S = {ρ_a, ρ_b}
a = min + ⌊(max − min) / 3⌋
b = min + 2 ⌊(max − min) / 3⌋
No
Solution feasible ?
No
min = max
Solve P3 using min as LB and max as UB
Yes
max - min < 1?
Yes
val(P(S)) = ρ_a ?
Yes
max = a
No
val(P(S)) = ρ_b ?
Yes
min = a + 1
max = b
No
min = b + 1

5 Results

5.1 P1

5.2 P3

5.2.1 Without relaxation

5.2.2 With relaxation

6 Discussion