UNIVERSITÉ LIBRE DE BRUXELLES

INFO-F-424 Combinatorial Optimization

The p-Center Problem #1

Antoine Passemiers
Cédric Simar

April 30, 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 and command line usage

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. Each `txt` file contains solver name, algorithm/formulation name, solution found by the solver (values of the y variables), value of the objective function and elapsed time.
- the `src` folder contains the source files in Julia
  - `p1.jl` implementation of P1 formulation
  - `p3.jl` implementation of P3 and P(S) formulations, BINARY and double bound (DB3) algorithm
  - `solve.jl` run p-center solver with command-line arguments
  - `test_all.jl` run p-center solver for all solvers and algorithms on all instances. This has been principally used to produce the results shown in present report

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 solve.jl` is used with the following arguments:
  - path to the instance file (required)
  - formulation of the p-center problem (either `p1`, `p3`, `p3-binary` or `p3-db3`) (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.

- `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.

- Because running both solvers on all instances for each algorithm can be very time-consuming, as well as calling the solver with command-line arguments many times in a row (because of the very long Julia warm up time), you may prefer to automate runs youself. If that is the case, just import `solve.jl` and call `solve_p_center(parameters::Dict{String, Any})` with proper parameters. Everything is documented in docstrings.

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:

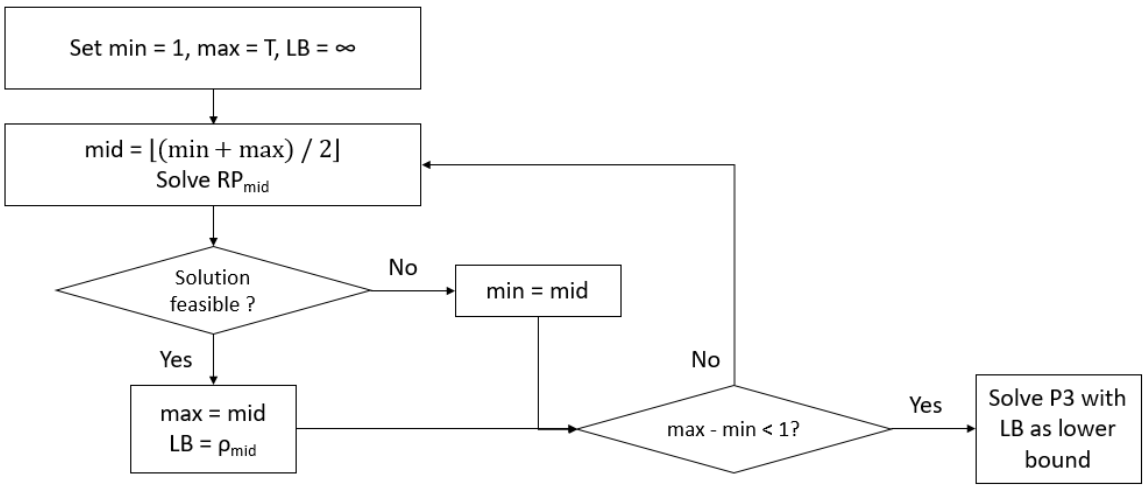

Figure4.1: Flow chart of BINARY algorithm

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

(b) If val$(P(S)) = \rho_k$ 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$(P(S)) = \infty$, 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:

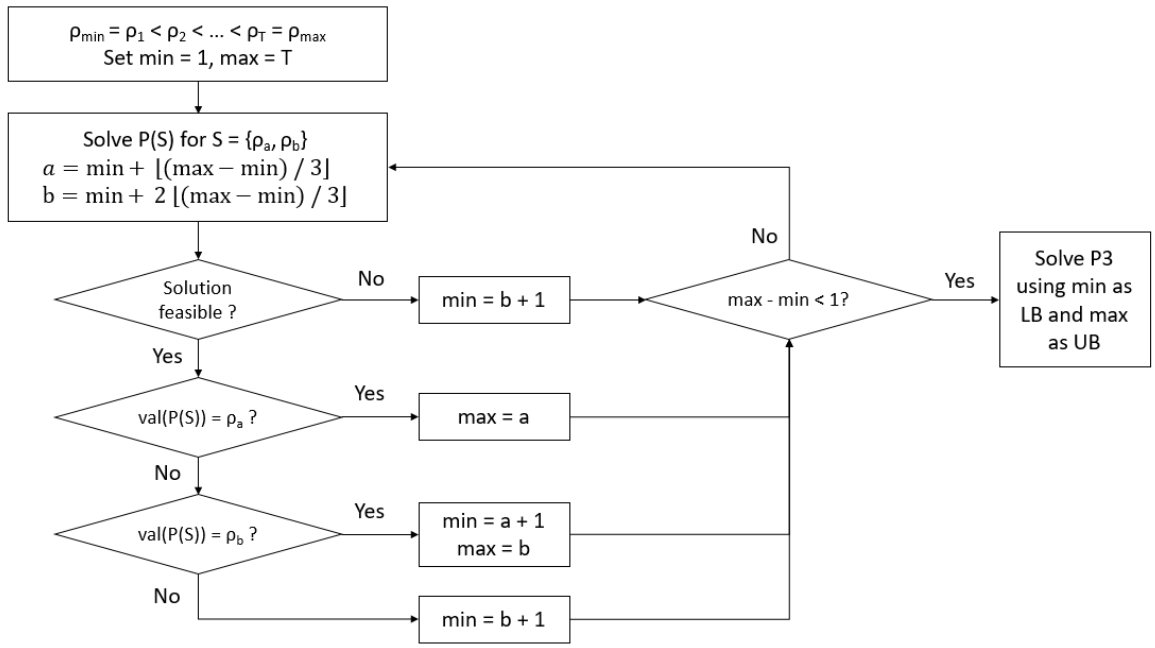

Figure4.2: Flow chart of DB3 algorithm

5 Results

These results were computed using the `julia test_all.jl` script which runs the two Cbc and GLPK solvers on all instances (30 easy instances and 10 hard instances) and writes the minimized objective function value and performance result of each instance in respective files in the `results` folder. For a better readability, we chose to make a separate result table per solver.

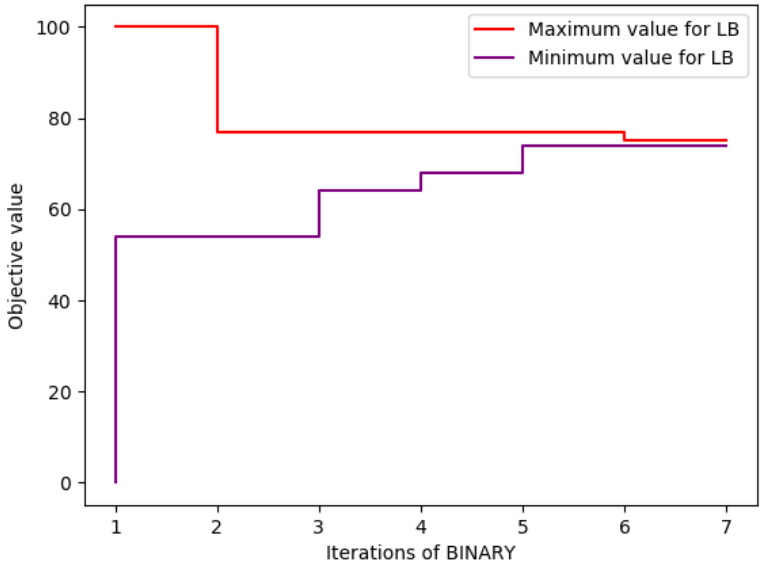

Figure5.1: Search space of BINARY algorithm for finding the lower bound on easy/instance10_1_1.dat

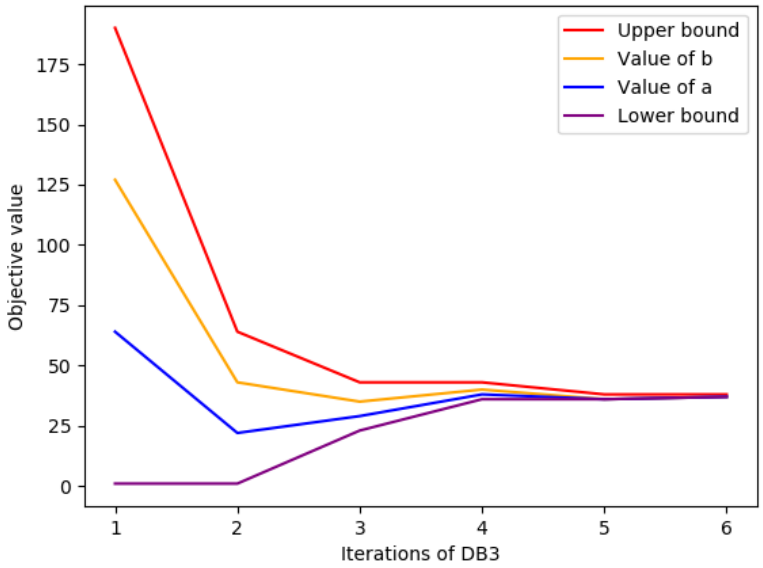

Figure5.2: Evolution of lower and upper bounds using DB3 algorithm on instance hard/instance9.dat

6 Discussion

6.1 Cbc

	P1		P3 BIN		P3 DB3	
	Obj	Time(s)	Obj	Time(s)	Obj	Time(s)
instance10_1_1.dat	75	2.422	75	0.899	75	0.214
instance10_1_2.dat	72	0.077	72	0.018	72	0.020
instance10_1_3.dat	85	0.080	85	0.013	85	0.018
instance10_1_4.dat	72	0.090	72	0.014	72	0.023
instance10_1_5.dat	70	0.089	70	0.015	70	0.020
instance10_2_1.dat	34	0.184	34	0.019	34	0.024
instance10_2_2.dat	43	0.123	43	0.113	43	0.050
instance10_2_3.dat	46	0.183	46	0.039	46	0.031
instance10_2_4.dat	42	0.103	42	0.073	42	0.032
instance10_2_5.dat	29	0.166	29	0.040	29	0.040
instance10_5_1.dat	9	0.132	9	0.200	9	0.015
instance10_5_2.dat	10	0.111	10	0.100	10	0.016
instance10_5_3.dat	9	0.100	9	0.103	9	0.030
instance10_5_4.dat	9	0.088	9	0.132	9	0.021
instance10_5_5.dat	11	0.097	11	0.104	11	0.015
instance20_2_1.dat	39	0.375	39	0.293	39	0.106
instance20_2_2.dat	53	1.570	53	1.249	53	0.136
instance20_2_3.dat	55	1.312	55	0.365	55	0.121
instance20_2_4.dat	51	1.163	51	0.163	51	0.166
instance20_2_5.dat	55	1.333	55	0.920	55	0.104
instance20_4_1.dat	21	0.260	21	0.311	21	0.043
instance20_4_2.dat	24	0.327	24	0.061	24	0.053
instance20_4_3.dat	23	0.261	23	0.067	23	0.088
instance20_4_4.dat	17	0.448	17	0.074	17	0.096
instance20_4_5.dat	22	0.342	22	0.524	22	0.080

Average computation time

	P1		P3 BIN		P3 DB3	
	Obj	Time(s)	Obj	Time(s)	Obj	Time(s)
instance20_10_1.dat	3	0.046	3	0.863	3	0.057
instance20_10_2.dat	4	0.152	4	0.892	4	0.026
instance20_10_3.dat	7	0.235	7	0.874	7	0.029
instance20_10_4.dat	4	0.136	4	1.533	4	0.033
instance20_10_5.dat	7	0.372	7	1.616	7	0.037
instance1.dat	127	140.100	127	450.646	127	3.105
instance2.dat	98	54.714	98	12.602	98	2.828
instance3.dat	93	71.143	93	15.900	93	1.524
instance4.dat	74	9.899	74	19.966	74	0.620
instance5.dat	48	6.591	48	26.465	48	0.923
instance6.dat	84	1109.202	84	103.477	84	2.178
instance7.dat	64	706.315	64	40.686	64	0.828
instance8.dat	55	572.170	55	74.232	55	3.316
instance9.dat	37	143.099	37	63.442	37	1.209
instance10.dat	20	32.296	20	60.019	20	0.821

	P1	P3 BIN	P3 DB3
	Time(s)	Time(s)	Time(s)
easy instances	0.413	0.389	0.058
hard instances	284.553	86.743	1.735

6.2 GLPK

	P1		P3 BIN		P3 DB3	
	Obj	Time(s)	Obj	Time(s)	Obj	Time(s)
instance10_1_1.dat	75	3.143	75	0.917	75	0.206
instance10_1_2.dat	72	0.026	72	0.002	72	0.003
instance10_1_3.dat	85	0.024	85	0.004	85	0.002
instance10_1_4.dat	72	0.017	72	0.003	72	0.004
instance10_1_5.dat	70	0.027	70	0.005	70	0.005
instance10_2_1.dat	34	0.043	34	0.006	34	0.003
instance10_2_2.dat	43	0.042	43	0.059	43	0.003
instance10_2_3.dat	46	0.037	46	0.004	46	0.003
instance10_2_4.dat	42	0.035	42	0.012	42	0.003
instance10_2_5.dat	29	0.036	29	0.005	29	0.008
instance10_5_1.dat	9	0.008	9	0.006	9	0.003
instance10_5_2.dat	10	0.021	10	0.007	10	0.003
instance10_5_3.dat	9	0.030	9	0.007	9	0.003
instance10_5_4.dat	9	0.019	9	0.005	9	0.004
instance10_5_5.dat	11	0.015	11	0.006	11	0.003
instance20_2_1.dat	39	0.864	39	0.018	39	0.012
instance20_2_2.dat	53	1.000	53	0.122	53	0.013
instance20_2_3.dat	55	1.252	55	0.057	55	0.013
instance20_2_4.dat	51	1.227	51	0.064	51	0.009
instance20_2_5.dat	55	1.113	55	0.120	55	0.012
instance20_4_1.dat	21	0.093	21	0.024	21	0.011
instance20_4_2.dat	24	2.430	24	0.012	24	0.007
instance20_4_3.dat	23	0.841	23	0.020	23	0.009
instance20_4_4.dat	17	1.529	17	0.031	17	0.009
instance20_4_5.dat	22	0.427	22	0.088	22	0.011
instance20_10_1.dat	3	0.546	3	0.027	3	0.007
instance20_10_2.dat	4	0.148	4	0.030	4	0.005
instance20_10_3.dat	7	0.346	7	0.024	7	0.012
instance20_10_4.dat	4	0.320	4	0.031	4	0.007
instance20_10_5.dat	7	0.219	7	0.023	7	0.012

Average computation time

Important note: unlike with Cbc, P1 forumulations of hard instances couldn't be solved in a reasonable time using GLPK

	P1 Time(s)	P3 BIN Time(s)	P3 DB3 Time(s)
easy instances	0.529	0.058	0.013

7 Discussion