INFO-F424 - Combinatorial Optimization
Project - The p-Center Problem

Erica Berghman
Charles Hamesse

École Polytechnique de Bruxelles

May 2017

Abstract

The purpose of this project is to implement two formulations of the same combinatorial optimization problem in Julia, using the JuMP package. We will start by describing the mathematical aspects of both formulations, then we will explain our implementations and discuss their performance.

Contents

1 Introduction

1.1 Implementation

1.2 Compiling and running

Compiling

Running Some example runs:

```
code
```

2 Formulations

2.1 Daskin (1995)

This is the formulation referred to as (P1) in the original paper.

According to the usual canvas, the mathematical formulation is given as follows.

Parameters

$$\begin{aligned} p &= \text{maximum number of centers;} \\ N &= \text{number of vertices of the instance.} \end{aligned}$$

Variables Two variables are used in this formulation:

$$y_j = \begin{cases} 1 & \text{if vertex } j \text{ is a center} \\ 0 & \text{otherwise} \end{cases}$$

$$x_{ij} = \begin{cases} 1 & \text{if vertex } i \text{ assigns to a center in vertex } j \\ 0 & \text{otherwise} \end{cases}$$

Both indices i and j have a range of $[1, N]$.

Objective function

$$min \quad z \tag{1}$$

$$\text{s.t.} \quad \sum_{j \in N} d_{ij} x_{ij} \leq z \tag{2}$$

These two expressions ensure that the objective value is no less than the maximum vertex-to-center distance, which we want to minimize. Note that (2) is actually implemented as a constraint but shown here for the sake of readability.

Constraints

$$\sum_{j \in N} x_{ij} = 1 \quad \forall i \in N \tag{3}$$

$$x_{ij} \leq y_i \quad \forall i, j \in N \tag{4}$$

$$\sum_{j \in N} y_j \leq p \tag{5}$$

$$y_j \in \{0, 1\} \quad \forall j \in N \tag{6}$$

$$x_{ij} \in \{0, 1\} \quad \forall i, j \in N \tag{7}$$

Constraint (3) assigns each vertex to exactly one center. Constraint (4) ensures that no vertex assigns to v_j unless there is a center at v_j. Constraint (5) restricts the number of centers to p. Constraints (6) and (7) are the binary restrictions for variables x and y.

2.2 Calik and Tansel (2013)

This is the formulation referred to as (P3) in the original paper. This method uses the fact that the distance d_{ij} are the only possible values for $r_p(F)$. It is thus possible to jump from one d_{ij} to another.

In this formulation, we define the set $R = \{\rho_1, \rho_2, ..., \rho_K\}$ where $\rho_1 < \rho_2 < ... < \rho_K$ is an ordering of the distinct distance values of the matrix of distances d_{ij}. One of these values determines the value of $r_p(F)$.

Parameters

$$\begin{aligned} p &= \text{maximum number of centers;} \\ N &= \text{number of vertices of the instance;} \\ K &= \text{number of distinct distance values in the instance.} \end{aligned}$$

Variables Three variables are used in this formulation:

$$a_{ijk} = \begin{cases} 1 & \text{if } d_{ij} \leq \rho_k \\ 0 & \text{otherwise} \end{cases}$$

$$y_j = \begin{cases} 1 & \text{if vertex } j \text{ is a center} \\ 0 & \text{otherwise} \end{cases}$$

$$z_k = \begin{cases} 1 & \text{if } r_p(F) = \rho_k \\ 0 & \text{otherwise} \end{cases}$$

Both indices i and j have a range of $[1, N]$.

Objective function

$$min \quad \sum_{k \in T} \rho_k z_k \tag{8}$$

The objective function determines the value of $r_p(F)$ as the corresponding value ρ_k.

Constraints

$$\sum_{j \in M} a_{ijk} y_j \geq z_k \quad \forall i \in N, \forall k \in T \tag{9}$$

$$\sum_{j \in M} y_j \leq p \tag{10}$$

$$\sum_{k \in T} z_k = 1 \tag{11}$$

$$y_j \in \{0,1\} \quad \forall j \in M \tag{12}$$

$$z_k \in \{0,1\} \quad \forall k \in T \tag{13}$$

Constraint (9) ensures that each vertex is covered within the selected radius by at least one center. Constraint (10) restricts the number of center to at most p centers. Constraint (11) ensures that exactly one of the variables z_k is selected. Constraints (12) and (13) are the binary restrictions for variables y and z.

3 Implementation

4 Optimization

4.1 Hot starts

Three hot starts have been tested: a random initialization, the 2-approximation algorithms and the initialization implemented in the Gurobi Optimizer.

4.1.1 Random initialization

The random initialization simply consists on choosing p points to be the centers.

As this algorithm is not demanding in computational resources, this initialization is done 1000 times and the best solution is kept as actual initial solution for the program linear.

4.1.2 2-approximation algorithm

The 2-approximation algorithm, also called the farthest-first traversal, is an heuristic that guaranties the solution will not be further away than 2 times the optimal value of the objective function if the triangular inequalities is respected.

The procedure is described in the pseudo-code here after. The first center is chosen randomly among the n points. The minimal distance between each of the points left and the already chosen centers is computed: this gives us for each point the smallest distance it is located from any of the centers. This distance is then maximized in order to chose the next center. This procedure is repeated until the number of centers chosen is equal to p.

```
2-approximation (π) :
    input:   problem instance π
             the number of points n
             the number of centers p
    output:  solution sol

    centers[p] = randomNumber(1,n)
    centersToFind = p - 1

    while (centersToFind)
        outer_max = inf
        centerIdx = -1
        for i = 1:n and i not in centers :
            inner_min = findMinDistance(centers, instance)
            if inner_min > outer_max
                outer_max = inner_min
                centerIdx = i
            end
        end
        centers[centersToFind--] = centerIdx
    end
    sol = zeros(n)
    sol[centers] = 1
    return sol
end 2-approximation
```

In this case, it is not clear if the distance metric considered has the triangular inequality, this is why no assumptions on the optimally of this solution can be done. The name farthest-first traversal will thus be preferred.

4.1.3 Gurobi Optimizer

This one was not implemented but was still part of the one explored as it is the default one when use the Gurobi Optimizer.

4.1.4 Comparison of the heuristics

4.2 Number of variables

4.2.1 Dual problem

The first idea to reduce the number of variables is to solve the dual problem. Indeed if a problem has l variables and m constraints, its dual problem has m variables and l constraints. If $l > m$, the dual problem does reduce the number of variables. Let's have a look at the formulations and their number of constraints and variables.

Formulation	# variables	# constraints
P_1	$n^2 + n + 1$	$n^2 + 2n + 2$
P_3	$n + K$	$nK + 2$

Considering that n and $K \in N$, the #variables < #constraints. For P_1 this is obvious. For P_2, we have to solve the inequality $n + K > nK + 2$, which would indicate that taking the dual would reduce the number of variables. Using WolframAlpha, the solution of this inequation for n > 1 is $K < \frac{n-2}{n-1}$ which is always smaller than 1. As K is an integer, this gives $K \leq 0$ which does not lead to any solution as $K > 0$ by definition.

4.3 Valid inequalities

4.3.1 Chvátal-Gomory procedure

Procedure to construct a valid inequality for the set $X = \{x \in R^n_+ : Ax \leq b\} \cap Z^n$, where A is a $m \times n$ matrix with columns $\{a_j, \text{j} = 1, \ldots, \text{n}\}$:

$$\sum_{j=1}^{n} \lfloor ua_j \rfloor \, x_j \quad \leq \quad ub \quad \text{where } u \in R^m_+ \text{ and } x \geq 0 \text{ is valid for P.}$$

This inequation is used in constraint (2) of P_1 with a factor called *divisor*:

$$\sum_{j \in N} \lfloor d_{ij}/divisor \rfloor \, x_{ij} \quad \leq \quad z/divisor \tag{14}$$

The divisor parameter has been optimized blablabla.

Multiple divisors have been used to add multiple ensemble of constraints.

With Gurobi there is not much improvement. However with another solver as Cbc we can see an improvement in the computational time.

4.4 Cutting planes

5 Results

Instance // Objective function // Example of solutions // computational time (mean) //

6 Conclusion

6.1 Comparison of the two formulations

6.2 Wrap-up