Algorithmic Methods for Mathematical Models
– COURSE PROJECT –
Diego Campos Milian

1. Problem statement

Given:

• The set T of center types. For each center type t its cost c_t, maximum distance between the center and the cities it serves $dcity_t$ and its maximum capacity c_t are specified..

• The set C of cities. For each city c its population p_c and position in (x,y) coordinates are specified.

• The set L of locations. For each location l its position in (x,y) coordinates are specified.

• The minimum distance between centers dcenter.

Find the assignment of centers to locations subject to the following constraints:

• Each city is assigned to exactly one primary and one secondary center, and those have to be in different locations.

• Each location can have assigned at most one center.

• Foreach center the distance between the center and the cities it servers cannot surpass $dcity_t$ for primary centers and 3 times $dcity_t$ for secondary centers.

• The distance between two centers have to be at least dcenter.

With the objective to minimize the total cost, sum of the cost of the installed centers.

2. MILP formulation

Sets and parameters:

T	Set of center types, index t.
C	Set of cities, index c.
L	Set of locations, index l.
p_c	Population of city c.
cap_t	Capacity of center type t.
$cost_t$	Cost of center type t.
$dcity_t$	Maximum distance between center type t and the cities it serves.
dcent	Minimum distance between centers
$dll_{l1,l2}$	Distance between locations l1 and l2.
$dcl_{c,l}$	Distance between city c and location l.

Decision variables:

$prim_{c,l}$ binary. Equal to 1 if city c is primary served from center at location l; 0 otherwise.
$sec_{c,l}$ binary. Equal to 1 if city c is secondary served from center at location l; 0 otherwise.
$inst_{l,t}$ binary. Equal to 1 if location l have center of type t; 0 otherwise.

Constraints:

$$\text{minimize} \sum_{l \in L} \sum_{t \in T} inst_{l,t} * cost_t$$

(1)

Subject to:

$$\sum_{l \in L} prim_{c,l} = 1 \quad \forall c \in C \tag{2}$$

$$\sum_{l \in L} sec_{c,l} = 1 \quad \forall c \in C \tag{3}$$

$$prim_{c,l} + sec_{c,l} \leq 1 \quad \forall c \in C,\ \forall l \in L \tag{4}$$

$$prim_{c,l} + sec_{c,l} - \sum_{t \in T} inst_{l,t} \leq 0 \quad \forall c \in C,\ \forall l \in L \tag{5}$$

$$\sum_{t \in T} inst_{l,t} \leq 1 \quad \forall l \in L \tag{6}$$

$$dll_{l1,l2} \geq dcent - M*(1 - (inst_{l1,t1} + inst_{l2,t2} - 1)) \quad \forall t1, t2 \in T,\ \forall l1, l2 \in L \tag{7}$$

$$cap_t \geq \left(\sum_{c \in C} p_c * prim_{c,l} + 1/10 * \sum_{c \in C} p_c * sec_{c,l} \right) - M*(1 - inst_{l,t}) \quad \forall t \in T,\ \forall l \in L \tag{8}$$

$$dcl_{c,l} * (inst_{l,t} + prim_{c,l} - 1) \leq dcity_t \quad \forall c \in C,\ \forall l \in L,\ \forall t \in T \tag{9}$$

$$dcl_{c,l} * (inst_{l,t} + sec_{c,l} - 1) \leq 3 * dcity_t \quad \forall c \in C,\ \forall l \in L,\ \forall t \in T \tag{10}$$

Explanations:

(1) Objective function: Set to minimize the sum of the costs of the installed centers.
(2) This constraint defines that all cities will have always a primary center.
(3) In the same way as (1) this constraint ensures that all cities will have only one secondary center.
(4) This one states that the primary and secondary center that served a certain city will be in different location.
(5) This one ensures that for all city c in cities if c is served by a center at location l there is a center of certain type in that l.
(6) This constraint ensures that for all locations the sum of centers assigned to this location will be at most 1.
(7) This one states that for every pair of installed centers the distance between them (dll) will be at least dcent.
(8) Ensures that for every center the sum of the cities population that serves as primary center plus the 10 per cent the sum of the cities population it serves as secondary center do not exceed its capacity.
(9) For all combination of city, location and center type this constraints assures that the distance between a city c and its primary center will not exceed $dcity_t$.
(10) For all combination of city, location and center type this constraints assures that the distance between a city c and its secondary center will not exceed 3 times $dcity_t$.

3. GRASP pseudo codes

3.1 Constructive phase

INPUT candidateList, α
OUTPÙT candidate

```
$sortedCandidateList = sortCandidateListByCost(candidateList)$
$lenCL = len(candidateList)$ //Number of candidatres in the list
$minCost = sortedCandidateList[0].cost$
$maxCost = sortedCandidateList[lenCL - 1].cost$
$boundaryCost = minCost + (maxCost - minCost) * \alpha$
$RCL = \{c \in sortedCandidateList \mid c.cost \leq boundaryCost\}$
return $random.choice(RCL)$
```

3.2 Local search phase

INPUT solution as bestSolution
OUTPÙT solution as bestSolution

```
$keepIterating = True$
while $keepIterating$ :
    $keepIterating = False$
    $neighbor = exploreNeighborhood(bestSolution)$
    $curCost = neighbor.cost$
    if $bestCost > curCost$ :
        $bestSolution = neighbor$
        $bestCost = curCost$
        $keepIterating = True$
    endif
return $bestSolution$
```

4. BRKGA

4.1. Chromosome structure

For this problem I have state that the number of genes of a chromosome is equal to the number of cities in the problem, and the genes will have a value between 0 and 1 that defines the priority of the city. So the chromosome will be an array of length equal to the number of cities which its elements contains the priority of the city, so if the index 0 of the array, first gene, have value 0.5 means that the city with id 0 have a priority value of 0.5. Important to say that the lower the priority value the higher the priority at the hour to compute the solution will be.

4.2. Decoder

INPUT individual, cities
OUTPÙT individual fitness

```
$genes\ =\ idividual.chromosome$
$citiesGenes\ =\ zip(cities,\ genes)$
$citiesGenes.sortBy(cityPoblation * geneValue)$
$individual.solution\ =\ emptySolution$
for $cityGene\ in\ citiesGenes$ :
        $cityId\ =\ cityGene[0].getId()$
        $assignment\ = findBestFeasibleAssignment(cityId)$
        $bestPrimId\ =\ assignment.primId$
        $bestSecId\ =\ assignment.secId$
        if $bestPrimId\ is\ None\ or\ bestSecId\ is\ None$ :
                $individual.solution\ =\ Infeasible$
                break
        endif
        $individual.solution.assign(cityId,\ assignment.primId,\ assignment.secId)$
endfor
$individual.fitness\ =\ individual.solution.getCost()$
```