A GRASP metaheuristic for a territorial design problem in financial institutions

Eduardo Salazar, Roger Z. Ríos

Department of Mechanical and Electrical Engineering
Universidad Autónoma de Nuevo León
San Nicolás de los Garza, N.L.

X Congreso de la Sociedad Mexicana de Investigación de Operaciones
October 20th, 2022

Outline

Territorial Design Problem

Problem formulation

According to literature [3], the main goal of a classic Territorial Design Problem is to minimize the sum of the distances of the B Basic Units (BUs henceforth), which represent clients, to their P territory centers that are selected out of S possible centers, fulfilling the specific demands of the BUs whilst taking into consideration several constraints that represent things such as unique assignments, metrics of BUs and their centers.

Financial institution's special requirements

Problem formulation

Following the only available example in previous literature [1], financial institutions have specific needs which translate to unique constraints.

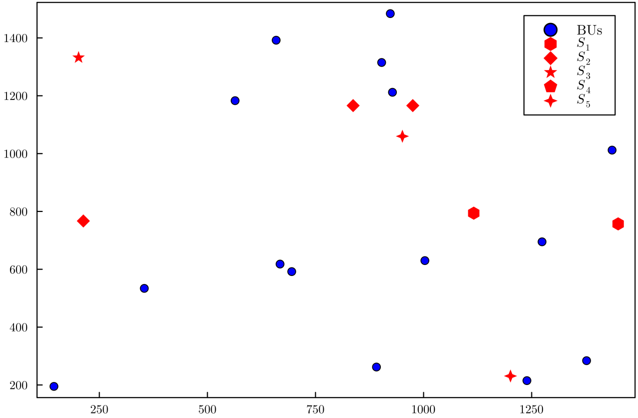
For example, each territory center has a specific S_k type of facility (gas station $= S_1$, supermarket $= S_2$, etc), with the number of centers with type k to be contained within lower and upper bounds L_k and U_k, respectively, for $k \in 1 \ldots 5$

Another value to be balanced between the territories is a risk parameter R associated with each BU, with each territory having a given β threshold of risk which cannot be exceeded.

Financial institution territory example

Instance example

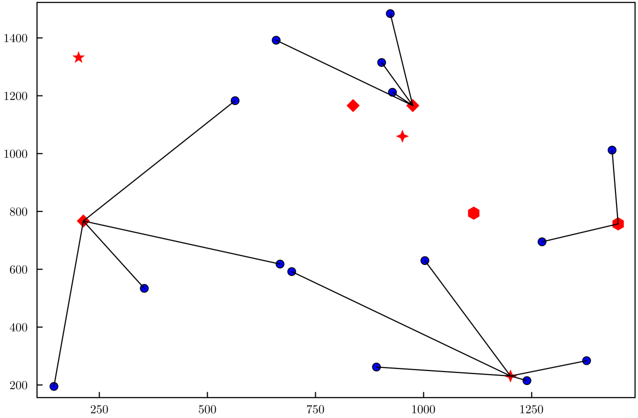
$B = 15$
$S = 8$
$P = 4$



Financial institution territory example

Solution example

$B = 15$
$S = 8$
$P = 4$



Applications

As stated before, the purpose of this model is to provide a generalization for financial institutions that wish to design the territories representing where to open a facility and which clients will be served by that facility. Risk balancing among the facilities is extremely important for such type of business, as a failure in a single facility node can compromise the entire network.

Motivation and Purpose

The model described in this work is motivated by a previous model described in [1]. In this model, the risk balancing is modelled as a constraint whereas the other model presents it as part of the objective function. Moreover, the other model was applied to an existing territorial design so the objective function also contained a term related to keeping as much as possible of the original design. This model can be viewed as a vertex p-center problem with multiple capacity constraints, given that even the uncapacitated vertex p-center problem is NP-hard [2], our TDP is also NP-hard therefore the perceived usefulness of heuristic approaches for solving it.

Combinatorial model

Problem formulation

Sets and parameters

- $S = \{1, 2, ..., s\}$, set of possible territory centers
- $B = \{1, 2, ..., b\}$, set of possible BUs
- D_{ij} = distance matrix between center i and BU j, $i \in S, j \in B$
- μ^i_m = target of activity m measure at center i
- v^j_m = measure of activity m at BU j
- t_m = tolerance of activity m measure
- R_j = risk measure at BU j
- β_i = risk threshold at center i
- $S_k = \{1, 2, ..., s\}$, 1 if s is of type k, 0 if not, $k \in 1 \ldots 5$
- L_k, U_k = Lower and upper bounds of centers to be used with type k
- P = Number of centers to be used

Combinatorial model

Problem formulation

Variables sets

- Y_i, binary variable vector where $Y_i = 1$ if the center i is used, 0 if not.
- X_{ij}, binary variable matrix where $X_{ij} = 1$ if the BU j is assigned to center i.

Mathematical programming

Formulation

Objective Function:

$$\textbf{min} \sum_{i \in S, j \in B} X_{ij} D_{ij} \tag{1}$$

Constraints:

$$\sum_{i \in S} X_{ij} = 1, \forall j \in B \tag{2}$$

Single assignment of a BU j to a center i

$$X_{ij} \leq Y_i, \forall i \in S, j \in B \tag{3}$$

Can only assign BUs to centers that are open

$$Y_i \mu_m^i (1 - t^m) \leq \sum_{j \in B} X_{ij} v_j^m \leq Y_i \mu_m^i (1 + t^m), \forall i \in S \tag{4}$$

Activity measures for each territory must be within a tolerance range

Mathematical programming

Formulation

$$l_k \leq \sum_{i \in S_k} Y_i \leq u_k \tag{5}$$

The selected centers' types must respect the lower and upper bound for each type

$$\sum_{i \in S} Y_i = P \tag{6}$$

The number of centers to be opened must be equal to P

$$\sum_{j \in B} X_{ij} R_j \leq \beta_i, \forall i \in S \tag{7}$$

The risk measure of each territory must not surpass the risk threshold

Proposed heuristic

Input: $P, \alpha, \gamma, i_{max}$, Instance
Output: X, Y = binary decision variables

$A^* \leftarrow \emptyset$
$f^* \leftarrow \infty$
while $i_{max} > 0$ **do**
    $X, Y \leftarrow$ Construct(α, P, Instance)
    $X, Y \leftarrow$ LocalSearch(X, Y, Instance)
    $A \leftarrow (X, Y)$
    if $f(A) < f^*$ **then**
        $f^* \leftarrow f(A)$
        $A^* \leftarrow A$
    end if
    $i_{max} \leftarrow i_{max} - 1$
end while
return A^*

A metaheuristic framework with a Greedy Randomized Adaptive Search Procedure (GRASP) using a value-based restricted candidate list (RCL). Parameters:

- α: Threshold quality parameter
- i_{max}: Number of iterations
- γ: Perturbation parameter.

Construction phase

The constructive heuristic used in this work consists of two phases: Location and Allocation.

In the Location phase, we must first determine which P centers are to be used out of all the available possible locations. This phase returns the decision variable vector Y.

The Allocation phase will allocate which center serves which BU, until all of the clients are allocated to a center. This phase will return the decision variable matrix X.

Construction phase

Location heuristics

- P-dispersion problem
- Relaxation of integer constraints
- Randomization

Construction phase

Location heuristics: P-Dispersion Problem

Input: P, S_coords, S
Output: $Y :=$ Binary vector of centers to be used

$S\_distance \leftarrow$ Euclidean Distance Matrix of all the centers with coordinates S_coords
$S\_sol \leftarrow argmax(S\_distance)$
$T \leftarrow 1...S$
$T \leftarrow T \setminus S\_sol$
while $|S\_sol| < P$ **do**
  $D \leftarrow []$
  for $i \in S$ **do**
    for $j \in T$ **do**
      $D[i] += S\_distance[i, j]$
    end for
  end for
  $max \leftarrow argmax(D)$
  $S\_sol \leftarrow S\_sol \cup max$
  $T \leftarrow T \setminus max$
end while
$Y = zeros(S)$
$Y \leftarrow Y[idx] = 1, \forall idx \in T$
return Y

Construction phase

Location heuristics: Relaxation of Integer Constraints

Input: Instance, P

Output: Y := Binary vector of centers to be used

1: $Model \leftarrow$ Build Mathematic Model from Instance

2: $Model.X \leftarrow$ Continous Value 0...1

3: $Model.Y \leftarrow$ Discrete Value 0,1

4: $Y \leftarrow Solve(Model)$

5: **return** Y

Construction phase

Location heuristics: Randomization

Input: P

Output: $Y :=$ Binary vector of centers to be used

1: $Y \leftarrow rand(0:1, P)$

2: **return** Y

Construction phase

Allocation heuristics

- Minimization of distances
- Cost of opportunity with restrictions in mind

Minimization of Distances

Input: D, Y, B

Output: $X :=$ binary decision matrix

$N \leftarrow [i, \forall_i \in Y == 0]$

for $j \in B$ **do**

    $i \leftarrow argmin(D[:, j]), i \notin N$

    $X[i, j] \leftarrow 1$

    continue

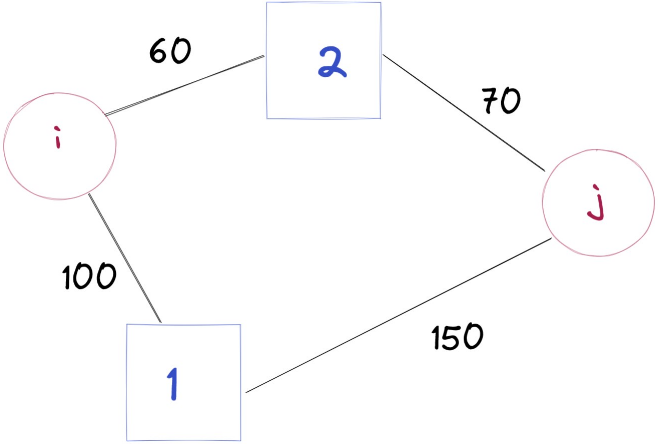
end for

return X

Cost of Opportunity

Computational results

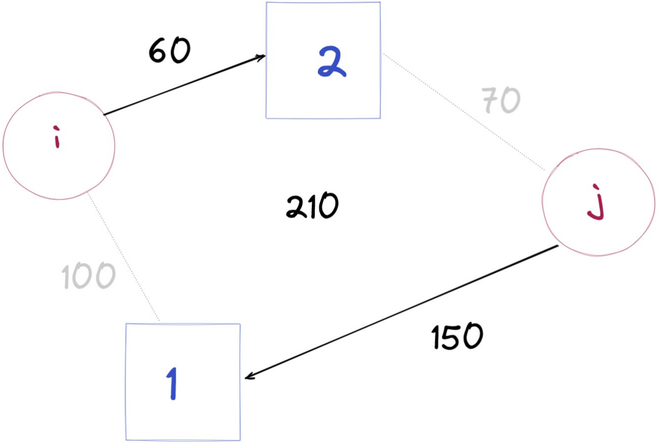
Let us propose a scenario with two centers i, j which must serve BUs 1 and 2. For illustrative purposes, the assignments of the BUs are exclusive, each center can only serve one BU.



Cost of Opportunity

Explanation

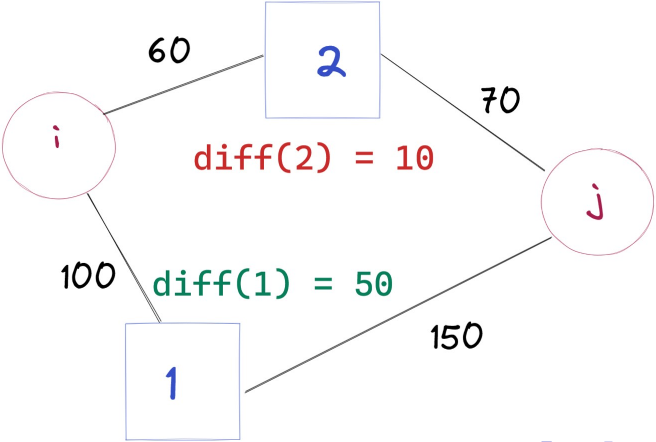
Following the minimization of distances approach, the assignments would be:



Cost of Opportunity

Explanation

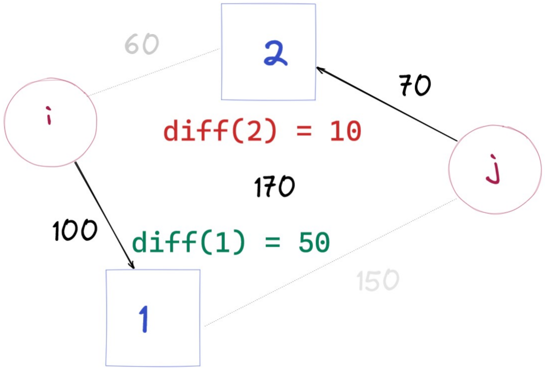
However, we may calculate the cost of opportunity as the difference between the optimal assignment and all the other possible assignments, selecting the largest opportunity lost:



Cost of Opportunity

Explanation

However, we may calculate the cost of opportunity as the difference between the optimal assignment and all the other possible assignments, selecting the largest opportunity lost:



Cost of opportunity Matrix

Input: D, P, Y, B
Output: C := cost of opportunity matrix

$C \leftarrow D$
$N \leftarrow [i, \forall i \in Y_i == 0]$
for all $j \in B$ **do**
  $i' \leftarrow \text{argmin}(D[:, i]), i \notin N$
  $C[i, j] \leftarrow D[i, j] - D[i', j], \forall i \in 1...S$
end for
return C

Search across the Cost Matrix

Input: $C, n, X, Instance$

Output: $CONS$:= Vector of constraints violated by possible asignation in X and information

IDXS $\leftarrow$ argmax(C, n)
for all IDX $\in$ IDXS **do**
  if IDX $\notin N$ **then**
    ROW $\leftarrow$ find$(C[:, IDX] = 0)$
    $X' \leftarrow X$
    $X'[ROW, IDX] \leftarrow 1$
    $CONS \leftarrow (ROW, IDX, constraintsCheck(Instance, X'))$
  end if
end for
return $CONS$

Upper Constraints Check

Input: Instance, X

Output: *cons* := Number of violated constraints

for Constraint in Constraints **do**
    Violated $\leftarrow$ Check(Instance.parameters, Solution)
    if Violated **then**
        $cons+=1$
    end if
end for
return *cons*

Cost of opportunity

Input: $D, P, Y, S, B, n,$ Instance
Output: X := binary decision variable

```
C ← GenerateCostMatrix(D, Y, B)
DONE ← FALSE
while NOT DONE do
    CONS ← SearchCost(C, n, X, Instance)
    i∗ ← argmin(CONS)
    if CONS[i*] = 0 then
        ROW, COL ← CONS[i∗]
        X[ROW, COL] ← 1
    else
        Candidates ← argmin(C[:, i∗], n)
        CONS_INNER ← []
        for Candidate ∈ Candidates do
            X∗ ← X
            X ∗ [Candidate, i∗] ← 1
            CONS_INNER ← (ROW, IDX, constraints(Instance, X∗))
        end for
        j∗ ← argmin(CONS_INNER)
        ROW ← CONS_INNER[j∗]
        COL ← CONS[i∗]
        X[ROW, COL] ← 1
    end if
    D[:, COL] ← ∅
    DONE ← Are All BUs Assigned?
end while
return X
```

Local search

Taking into consideration that the solution provided by the Constructive Heuristic needs to be "repaired", the local search moves first try to minimize constraints violated.

BU_SimpleReassign(ψ, τ)

Move a BU j from center ψ to the center τ, where $\psi \neq \tau$

BU_Interchange(ψ, τ)

Interchange the assignments of BUs ψ and τ

Center_SimpleReassign(ψ, η)

Change Y_ψ from 1 to 0 and Y_η from 0 to 1, allocate all the orphaned BUs from center ψ to η

Center_SmartReassign(ψ, η)

Change Y_ψ from 1 to 0 and Y_η from 0 to 1, allocate all the orphaned BUs from center ψ using the same allocation process of the Cost of Opportunity strategy.

Experiments layout

Computational results

The heuristics were coded in Julia 1.8 The source code can be found in the following repository: `https://github.com/eduardosalaz/tesis`.
The platform is Intel Core i5-9300 2.5 GHz, 8 GB RAM under Windows 10. For the experiments, instances were generated with BUs and Centers' coordinates located randomly between 5 and 4500 along with the center types, risk and activity measures.

Constructive algorithms comparison

Computational results

Data set

- Size 1: 25 instances with 310 BUs, 60 centers and 20 to be located.
- Size 2: 10 instances with 800 BUs, 150 centers and 50 to be located.

Constructive algorithms comparison

Computational results

Dataset	Location h.	Avg. constraints		% Factible sols.		Avg. time alloc. (s)		Avg. time loc. (s)
		M	C	M	C	M	C	
1	Rlx	5.92	0.08	0	96	0.004	2.05	1.2
	Pdp	42.2	8.44	0	0	0.005	1.114	0.0006
	Rnd	21.76	4.92	0	0	0.004	0.954	0
2	Rlx	12.4	0	0	100	0.014	47.7	16.6
	Pdp	102.4	19.4	0	0	0.016	69.54	0.0009
	Rnd	43.5	8.9	0	0	0.013	56.84	0

Table: Summary of the experiment results

Based on these results, it was decided to use the Integer Relaxation as the Location Phase and the Opportunity Cost as the Allocation Phase for the Constructive Phase of the GRASP Procedure.

Constructive algorithms comparison

Comparison with the Solver

Dataset	Avg. Gap% to Optim
1	51.7
2	51.2

Table: Summary of the experiment results

Part of the methodology of the experiments involved programming the model in order to feed it to an exact solver, in this case CPLEX to provide a baseline of the optimal value and how much time it takes to solve with a cutoff time of 300 seconds.

Local Search algorithms comparison

Computational results with violated constraints

Dataset	Move	Avg. constraints improv. %
1	BU_Simple	39.31
	BU_Interchange	0
	Center_Simple	6.22
	Center_Smart	8.03
2	BU_Simple	29.63
	BU_Interchange	0
	Center_Simple	2.67
	Center_Smart	0

Table: Summary of the experiment results

With the application of the local search focused on minimizing the number of constraints violated in unfactible solutions, we find that there is an improvement on the value mainly coming from the BU_Simple move, whereas BU_Interchange proves inefective.

Local Search algorithms comparison

Computational results

Dataset	Move	Avg. Time	Avg. rel. improv. %
1	BU_Simple	4.4	48.35
	BU_Interchange	2.09	2.94
	Center_Simple	66.4	0.02
	Center_Smart	1.68	0
2	BU_Simple	16.5	48.32
	BU_Interchange	1.67	1.67
	Center_Simple	15.68	0
	Center_Smart	31.12	0

Table: Summary of the experiment results

Each Move had a fixed number of iterations, for the first dataset, $max\_iters = 1000 : 1500$ but we suspect there is a bug somewhere, as for the same number of iterations, the second dataset would exhaust the memory, so max_iters had to be lowered to 200 : 250. Based on these results, it was decided to use BU_Simple and BU_Interchange moves in the GRASP procedure with around 200 iterations for each move.

Constructive and Local Search comparison

Comparison with the Solver

Dataset	Avg. Gap% to Optim
1	5.86
2	5.25

Table: Summary of the experiment results

Even though there is a considerable improvement in the objective function, not all moves are useful for improving the objective function value.

GRASP Pseudocode

Input: $P, \alpha, \gamma, i_{max}$, Instance

Output: X, Y = binary decision variables

$A^* \leftarrow \emptyset$

$f^* \leftarrow \infty$

while $i_{max} > 0$ **do**

  $X, Y \leftarrow$ Construct(α, P, Instance)

  $X, Y \leftarrow$ LocalSearch(X, Y, Instance)

  $A \leftarrow (X, Y)$

  if $f(A) < f^*$ **then**

    $f^* \leftarrow f(A)$

    $A^* \leftarrow A$

  end if

  $i_{max} \leftarrow i_{max} - 1$

end while

return A^*

GRASP Location

Input: $P, \gamma,$, Instance

Output: $Y =$ binary decision variable

1: $Y \leftarrow \mathit{IntegerRelaxation}(\mathit{Instance})$

2: $Y' \leftarrow$ Perturbate Y (turn off γ centers previously on and turn on γ centers previously off)

3: **return** Y'

Construct Pseudocode

```
Input: P, α, γ,, Instance
Output: X, Y = binary decision variables
Y′ ←Location(P, γ, Instance)
C ← GenerateCostMatrix(D, Y, B)
DONE ← FALSE
while NOT DONE do
   CONS ← SearchCost(C, n, X, Instance)
   i∗ ← argmin(CONS)
   if CONS[i*] = 0 then
      ROW, COL ← CONS[i∗]
      BestVal ← C[ROW, COL]
      Cutoff ← (BestVal + (BestVal ∗ α))
      RCL ← [IDX, ifC[IDX] <= Cutoff]
      ROW, COL ← Rand(RCL)
      X[ROW, COL] ← 1
   else
      BestVal ← C[ROW, COL]
      Cutoff ← (BestVal + (BestVal ∗ α))
      RCL ← [IDX, ifC[IDX] <= Cutoff]
      i∗ ← Rand(RCL)
      Candidates ← argmin(C[:, i∗], n)
      CONS_INNER ← []
      for Candidate ∈ Candidates do
         X∗ ← X
         X ∗ [Candidate, i∗] ← 1
         CONS_INNER ← (ROW, IDX, constraints(Instance, X∗))
      end for
      j∗ ← argmin(CONS_INNER)
      ROW ← CONS_INNER[j∗]
      COL ← CONS[i∗]
      X[ROW, COL] ← 1
   end if
   D[:, COL] ← ∅
   DONE ← Are All BUs Assigned?
end while
return X, Y
```

LocalSearch Pseudocode

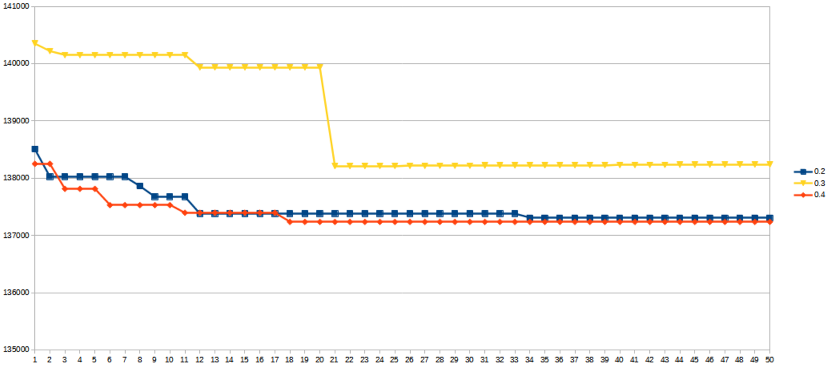
Input: *Instance*, X, Y
Output: X', Y' = binary decision variables

```
Fac ← isFactible(X, Y, Instance)
if Fac = false then
    X', Y' ← BU_Simple(X, Y, Instance, Repair)
    Fac' ← isFactible(X', Y', Instance)
    if Fac' = false then
        X', Y' ← BU_Interchange(X', Y', Instance, Repair)
        Fac∗ ← isFactible(X', Y', Instance)
    end if
end if
if Fac∗ = false then
    Skip iteration
else
    X', Y' ← BU_Simple(X, Y, Instance, Improve)
    X', Y' ← BU_Interchange(X', Y', Instance, Improve)
end if
return X', Y'
```

Calibrating GRASP iterations

Computational results

In order to gain insight about when the solutions yielded by the GRASP algorithm stop improving, we run an experiment with $i_{max} = 50$ and $\alpha = 0.2, 0.3, 0.4$ for the objective function average of 5 instances of the first dataset.



GRASP runtime

Computational results

As we tested the GRASP procedure speed, we found the following results:

Dataset	Loc. time	Alloc. time.	Move time.	Avg. rel. improv. %	Avg. Gap% to optim
1	2.1	23	8.4	49.21	2.9
2	17.5	40.6	20.6	47.34	5.2

Table: Summary of the experiment results

Comparing GRASP with the Exact Solver

Computational results

With an $\alpha = 0.3$ and 30 iterations, we decided to try and solve larger instances to test the capabilities of the metaheuristic. So, the last test dataset consisted of a single instance with 1300 BUs, 190 centers and $P = 65$. The solver did not finish optimally at the cutoff of 300 seconds, whereas the GRASP ran out of memory at the third iteration. Still, the GRASP took 86 seconds to provide a solution within 3.7% of the solver's solution.

Wrap-up

Conclusions

- Using an integer relaxation of the problem and the cost of opportunity assignment we get factible solutions from the constructive heuristic, with the simple BU reassignment move proving the most effective for the Local Search.
- On fine-tuning stage of GRASP, it was observed that a value of $i_{max} = 30, \alpha = 0.3$ was good enough
- GRASP provides good solutions, however the time and memory spent in the local search phase made it unfeasible to test on larger instances.

Future work

- Explore the usage of P-Dispersion as the Location Phase instead of Relaxation.
- Model the risk as an uncertainty measure.
- Optimize code, profile performance, detect bottlenecks and memory leaks
- Improve the algorithm data structures to reduce running times
- Explore other location phase heuristics
- Explore other strategies (TS, ILS, IGLS, VNS, SS)

Acknowledgments

- Thank you very much for your attention!
- Feedback is welcome :)
- Email: eduardosalaz@outlook.com
- GitHub: `https://github.com/eduardosalaz`

References

[1] Jesús Fabián López Pérez et al. "Risk-balanced territory design optimization for a Micro finance institution". In: *Journal of Industrial and Management Optimization* 16.2 (2020), pp. 741–758.

[2] Roger Z. Ríos-Mercado and Hugo Jair Escalante. "GRASP with path relinking for commercial districting". In: *Expert Systems with Applications* 44 (2016), pp. 102–113. ISSN: 0957-4174. URL: https://www.sciencedirect.com/science/article/pii/S0957417415006338.

[3] Roger Z. Ríos-Mercado and Elena Fernández. "A reactive GRASP for a commercial territory design problem with multiple balancing requirements". In: *Computers & Operations Research* 36.3 (2009), pp. 755–776. ISSN: 0305-0548. URL: https://www.sciencedirect.com/science/article/pii/S0305054807002249.