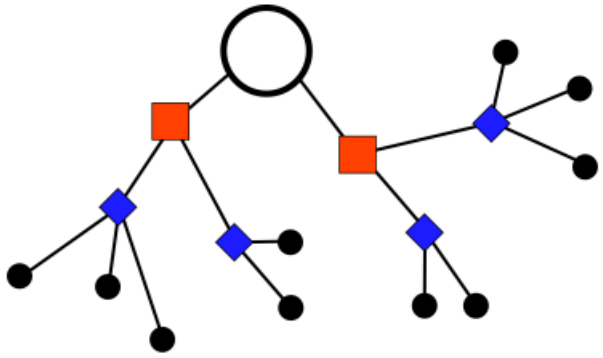
LABORATOIRE DES SCIENCES DU NUMÉRIQUE DE NANTES

Nantes Université

Scatter search for the TSUFLP (Two-Stages Uncapacited Facility Location Problem)



Multi-objective Metaheuristics
Project report

Juanfer MERCIER — Adrien PICHON

30 December 2023

Nantes Université — UFR Sciences et Techniques

Contents

Chapter 1

Introduction and TSUFLP

In this report, we will present the Two-Stage Uncapacitated Facility Location (TSUFLP) problem, the instances that we used to solve the problem and the Scatter Search procedure as presented in the paper "A New Scatter Search Design for Multiobjective Combinatorial Optimization with an Application to Facility Location" [1]. In this section we will present the TSUFLP problem, an example of this problem and the numerical instances that we used to solve the problem. Section 2 details our implementation of the Scatter Search procedure with its main components (i.e. GRASP, Tabu Search and Path Relinking). Section 3 describes our experiments and the computational results.

The following presentation of the problem is based on the paper "A memetic algorithm for solving two variants of the Two-Stage Uncapacitated Facility Location Problem" [2]. The Two-Stage Uncapacitated Facility Location Problem (TSUFLP) has important applications in designing telecommunication systems. Given a set of demand points and a set of possible locations for the first and second level concentrators (switches, multiplexors), the goal of the TSUFLP is to define the structure of two level concentrator access network, such that the total cost of establishing such a network is minimized [2]. Other objectives with other natures may be proposed to reach a balanced workload between all the concentrators or a fairness distance between the demand points and the first level concentrators.

Consider a given set of locations of terminals, a set of potential locations for installing the first-level concentrators and a set of potential sites for locating the second-level concentrators. An efficient solution given multiple objectives aims to choose locations on the first and second level for installing certain number of concentrators, and to make necessary assignments of each terminal to an installed first-level concentrator, which further have to be assigned to one of the concentrators established at locations on the second-level. The problem in its standard form involves a single allocation scheme, which means that each terminal is assigned to exactly one, previously established concentrator on the first-level, while each concentrator is assigned to at most one, previously located concentrator on the second level. Many papers in the literature are devoted to the TSUFLP and its variants. The problem can be viewed as an extension of the uncapacitated facility location problem (UFLP) [3] or as a set packing problem (SPP) [4].

1.1 Model

We consider a single objective integer formulation of the TSUFLP given in [3].
Data:

- $I = \{1, ..., nI\}$, a set of locations of terminals;
- $J = \{1, ..., nJ\}$, a set of possible locations for installing first-level concentrators;
- $K = \{1, ..., nK\}$, a set of possible locations for establishing second-level concentrators;
- c_{ij} is the cost of assigning terminal at location $i \in I$ to a concentrator at location $j \in J$ on the first-level;
- b_{jk} is the cost of installing concentrator at location $j \in J$ on the first-level and connecting it to the concentrator at location $k \in K$ on the second-level;
- s_k is the cost of setting up a concentrator at site $k \in K$ on the second-level.

Decision variables:

- $x_{ij} \in \{0, 1\}, i \in I, j \in J$ is equal to 1 if terminal at $i \in I$ is assigned to concentrator installed at location $j \in J$ on the first level, 0 otherwise;
- $y_{jk} \in \{0, 1\}, j \in J, k \in K$ is equal to 1 if a concentrator at location $j \in J$ on the first level is connected to a concentrator at location $k \in K$ on the second level, 0 otherwise;
- $z_k \in \{0, 1\}, k \in K$ is equal to 1 if a concentrator is installed at location $k \in K$ on the second level, 0 otherwise;

Formulation:

$$\min\ f_1(x, y, z) = \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} + \sum_{j \in J} \sum_{k \in K} b_{jk} y_{jk} + \sum_{k \in K} s_k z_k \quad (1)$$

s.t.

$$\sum_{j \in J} x_{ij} = 1 \quad , \forall i \in I \quad (2)$$

$$\sum_{k \in K} y_{jk} \geq x_{ij} \quad , \forall i \in I, \forall j \in J \quad (3)$$

$$z_k \geq y_{zk} \quad , \forall j \in J, \forall k \in K \quad (4)$$

$$\sum_{k \in K} y_{jk} \leq 1 \quad , \forall j \in J \quad (5)$$

$$x_{ij} \in \{0, 1\} \quad , \forall i \in I, \forall j \in J \quad (6)$$

$$y_{jk} \in \{0, 1\} \quad , \forall j \in J, \forall k \in K \quad (7)$$

$$z_k \in \{0, 1\} \quad , \forall k \in k \quad (8)$$

where:

(1) minimizes the sum of costs of establishing the concentrators at locations on the first and the second levels, plus the costs of assigning terminals to first-level concentrators and assigning first-level concentrators to second level concentrators;

(2,3) each terminal is allocated to exactly one concentrator located on the first level, which is further connected to a concentrator located on the second level;

(4,5) each first level concentrator is assigned to at most one second-level concentrator;

(6,7,8) the variables are binary.

1.1.1 Variant considered

As the objective of this project is to study and apply multi-objectives methods, we had to consider a second objective to the TSUFLP problem. Our choice ended up being an objective of the *p-center problem* (or *pull objective*): the objective where we seek to minimize the maximum distance between terminals and their nearest first-level concentrators.

We also chose to add another constraint to the problem and we selected the *multiple homing* constraint which mean that one terminal can be assigned to more than one concentrator, here we consider each terminal is assigned to exactly two concentrators, one for the primary coverage and the second for the secondary or backup coverage in case of disturbance on the primary coverage.

To take into account our variant we made the following changes to the previous model:

$$\min \; f_2(x,y,z) = \max_{i\in I, j\in J} c_{ij}x_{ij} \qquad (1b)$$

s.t.

$$\sum_{j\in J} x_{ij} = 2 \qquad ,\forall i \in I \qquad (2)$$

Here is an example of the TSUFLP problem provided in the project subject.

1.2 Example

Consider a network illustrated by Figure 1 with $|I| = 5$ terminal nodes $(t_1, t_2, t_3, t_4, t_5)$, $|J| = 3$ potential locations for first-level concentrators (p_1, p_2, p_3) and $|K| = 2$ potential locations for second-level concentrators (q_1, q_2).

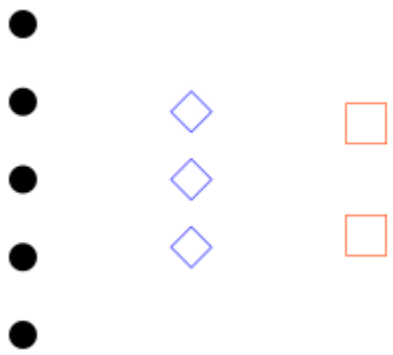


Figure 1: Illustration of an example with 5 terminal nodes, 3 potential locations for first-level concentrators, 2 potential locations for second-level concentrators.

The assignment costs of terminals to first level concentrators are given in Table 1, the costs for installing first-level concentrators and their assignments to second-level concentrators are available in Table 2, and finally the costs of establishing second-level concentrators are equal to $s_1 = 20$ and $s_2 = 16$ respectively.

	p_1	p_2	p_3
t_1	12	22	18
t_2	14	19	20
t_3	20	31	13
t_4	22	21	2
t_5	17	24	9

Table 1: Assignments costs of terminals to first level concentrators.

	q_1	q_2
p_1	29	12
p_2	28	31
p_3	21	13

Table 2: Costs for installing first-level concentrators and their assignments to second-level concentrators.

The optimal solution of this example is illustrated in Figure 2. Concentrators p_1 and p_3 are installed on the first level, while concentrator q_2 is established on the second level. Terminal nodes t_1 and t_2 are assigned to p_1, while t_3, t_4 and t_5 are assigned to p_3. Both first-level concentrators p_1 and p_3 are allocated to the second-level concentrator q_2. The objective function that corresponds to this optimal solution takes the values of 91.

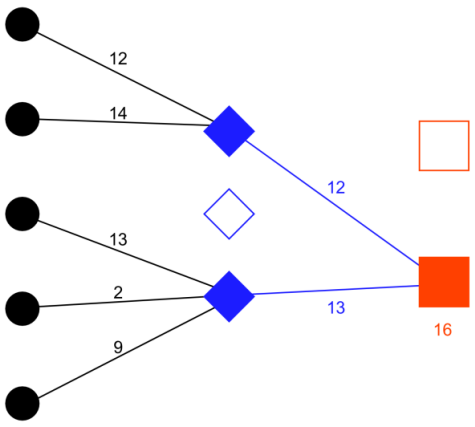


Figure 2: Illustration of an example with 5 terminal nodes, 3 potential locations for first-level concentrators, 2 potential locations for second-level concentrators.

1.3 Numerical Instances

The numerical instances used to test our scatter search implementation comes from Sanchez's paper [1]. They were generated with the code provided by Karatas and Yakici [5]. Table 3 summarizes the parameter values of the three sets of problem instances from [1]. In our case, we only considered Set 1 and Set 2 and we also discarded p and r since we use $4/5$ of the m concentrators from Sanchez instances as first level concentrators and the other $1/5$ of the m concentrators as second level concentrators. The terminals locations are used as is.

Other instances created using open-data available on the FTTH Orange (PONS) network on the downtown of Nantes[1] and Angers (GeoJSON instances). Although we tested our algorithm on these instances, no results for these instances will be presented here as they were inconclusive.

Parameter	Set 1 (small size)	Set 2 (medium size)	Set 3 (large size)
m	20	50	200
n	50	100	400
p	5	10	15
r	20	15	10

Table 3: Characteristics of problem instances. In our case, m is the number of concentrators, n the number of terminals. p and r are not used here.

To be more specific, we used 21 instances from Set 1 namely `small1` to `small20` and `verySmall1`. We also used 10 instances from Set 2 (`medium1` to `medium10`).

[1] See `https://github.com/xgandibleux/momhTSUFLP`

Chapter 2

Scatter Search

Scatter Search is a population-based metaheuristic originally proposed in [6] by F. Glover for mono-objective problems. The multi-objective variant we implemented for this project is the Scatter Search procedure proposed in [1]. Since the problem we consider is bi-objective we'll present the procedure for bi-objective problems. The multi-objective Scatter Search procedure is divided in several components:

- a construction phase to create a population of solutions. The Greedy Randomized Adaptative Search Procedure (GRASP) [7] algorithm is used to construct a good and diversified initial population. This phase is applied to each objective independently. As such, the population is divided in two sets : a set of initial solutions constructed with respect to the first objective and another set of initial solutions constructed with respect to the second objective function.
- an improvement phase is applied to the population. The Tabu Search (TS) [8] is used to improve the initial solutions, the two aforementioned sets are improved by TS.
- finally, a combination phase is used to combine the two sets of solutions and find compromise points with a good balance among the two objectives.

We'll present these components in this section.

2.1 GRASP

The GRASP component (see Algorithm 1) is used to create an initial population that is diversified enough and with good quality. It works by iteratively selecting level 1 concentrators based on a random construction procedure using a restricted candidate list (RCL). The selected level 1 concentrators are then connected to the terminals, and subsequently, each connected level 1 concentrator is connected to a level 2 concentrator. As suggested in [1], two functions g_1 and g_2 are defined. These functions compute a greedy function value for a given concentrator j with respect to the first and second objective functions, respectively:

$$g_1(j) = \frac{1}{\sum_{i=1}^{|I|} C_{i,j} + \sum_{k=1}^{|K|} B_{j,k}} \qquad g_2(j) = \frac{1}{\max_{i=1}^{|I|} C_{i,j}}$$

. These utility functions provide a measurement of the variation in objective value for the solution if concentrator j is selected. Our implementation of the GRASP method uses two parameters :

- α a parameter that controls the degree of randomness in the solution construction process. Specifically, it is used to determine the limit L for selecting candidates from the Restricted Candidate List (RCL). The RCL is a subset of concentrators from which the algorithm randomly selects the next concentrator to include in the solution. L is expressed as:

$$L = g_{\max} - \alpha \cdot (g_{\max} - g_{\min})$$

  where $g_{\max}$ is the maximum greedy function value (obtained with g_1 or g_2 depending on which objective is chosen for the construction of the current solution) among the remaining candidates, and $g_{\min}$ is the minimum greedy function value among the remaining candidates. A higher α encourages more exploration (more random), while a lower α leads to more exploitation of promising solutions (more greedy).

- p a parameter that represents the proportion of level 1 concentrators to locate during the solution construction process. The algorithm selects $p \times |J|$ level 1 concentrators in total. This parameter controls the number of concentrators selected in the solution.

Algorithm 1 GRASP procedure

procedure GRASP(g, α, p)
  $J_t \leftarrow J$
  $lvl1 \leftarrow \emptyset$
  $Eval_J \leftarrow \{g(j), \forall j \in J\}$
  $x_{ij} \leftarrow 0, \forall i \in I, \forall j \in J$
  $y_{jk} \leftarrow 0, \forall j \in J, \forall k \in K$
  $z_k \leftarrow 0, \forall k \in K$

  while $|lvl1| \neq p \times |J|$ **do**
    $L \leftarrow \max(Eval_{J_t}) - \alpha * (\max(Eval_{J_t}) - \min(Eval_{J_t}))$
    $RCL \leftarrow \{j \in J_t, g(j) \geqslant L\}$
    $j^* \leftarrow RandomSelect(RCL)$
    $lvl1 \leftarrow lvl1 \cup \{j^*\}$
    $J_t \leftarrow J_t \setminus \{j^*\}$
  end while

  for $i \in I$ **do**
    Let j_1 and j_2 be the closest level 1 concentrators to terminal i in $lvl1$
    Set $x_{ij_1} = 1$ and $x_{ij_2} = 1$
  end for

  for $j \in lvl1$ **do**
    Let k be the closest level 2 concentrator to level 1 concentrator j
    Set $y_{jk} = 1$ and $z_k = 1$
  end for

  return x_{ij}, y_{jk}, z_k
end procedure

2.2 Tabu Search

The Tabu Search (TS) component is used to improve the solutions from the initial population. The key idea behind TS is to iteratively explore the solution space by making moves from the current solution to neighboring solutions while keeping track of a short-term memory structure known as the "Tabu List" (TL). The tabu list is used to prevent the algorithm from revisiting recently explored solutions. The Tabu Search requires two parameters (see Algorithm **??**):

- TL the size of the Tabu List. A larger Tabu List can allow the algorithm to explore a broader solution space but might slow down the convergence. The optimal size depends on the characteristics of the problem. The tabu list prevents the algorithm from revisiting the same solutions within a certain number of iterations, helping to diversify the search and avoid cycling.
- k the maximum number of iterations allowed without finding an improved solution. A higher value of k allows for more exploration before restarting, but it may also increase the computational cost. This parameter introduces a mechanism to diversify the search and escape local optima by restarting the exploration when improvement stagnates.

The tabu list is used to represent forbidden states. In the literature, we can add solutions to the tabu list or the moves that lead to a given forbidden solution. Here, we chose to forbid the moves leading to a solution. Iteratively, a move generation mechanism is used to create a set of candidate moves from the current solution. These moves represent changes to the current solution (e.g. swapping the terminals of two level 1 concentrators, assigning a terminal assigned to a level 1 concentrator to another level 1 concentrator). Tabu moves are excluded from the set of possible moves while they are in the tabu list unless their inclusion leads to a solution that is better than the best solution visited during the current search (*aspiration criteria*).

Algorithm 2 Tabu Search

```
1: procedure TABU(TL, k)
2:     Create Tabu List of length TL
3:     Get current solution sol obtained with GRASP
4:     while the number of iteration without improvement ≠ k do
5:         Generate candidate moves from sol
6:         Evaluate objective function values for candidate moves
7:         Select a move based on some strategy
8:         if selected move is not tabu OR it improves the objective value then
9:             Apply the move to update sol
10:            Update Tabu List
11:            if new solution improves the objective then
12:                Update best solution
13:            end if
14:        end if
15:    end while
16: end procedure
```

The neighborhoods (moves) we consider are

- **Swap Move**: Swap terminals assigned to two level 1 concentrators.

- **Shift Move**: Reassign a terminal of a level 1 concentrator to a different level 1 concentrator.
- **Drop Move**: Remove terminals from a concentrator and reconnect them to their nearest level 1 concentrators.
- **Open Move**: Open a concentrator, reconnect terminals to their two closest level 1 concentrators and connect the level 1 concentrators to their closest level 2 concentrator.

2.3 Path Relinking

The path relinking [9] procedure aims to create new solutions that bridge the gap between the two halves of the population, exploring the solution space for potential efficient solutions. Its main goal in the Scatter Search method is to provided well-balanced compromise solutions for the two objectives. As aforementioned, the population is split into two halves (the solutions improved by tabu search with respect to the first objective and the solutions improved by TS with respect to the second objective). Each half is sorted based on the first and second objective values, respectively. To add some variation, we also randomly select 10 solutions from each half to perform the path relinking procedure between diverse pairs of solutions. The path relinking procedure works as follow:

- select a pair of solutions $(S', S")$ where S' belongs to the first half of the population (first objective) and $S"$ belongs to the second half of the population. The solution that dominates the other is called the *guiding solution* while the other one is the *initiating solution*.
- create two sets In (insertion set) and Out (removal set) where In contains the index of the first-level concentrators from the guiding solution that are not present in the initiating solution, while Out contains the index of the first-level concentrators from the initiating solution that are not present in the guiding solution. The goal is to transform the initiating solution into the guiding solution by iteratively adding concentrators from In and removing concentrators from Out.
- Each iteration consists of selecting a pair of concentrators, one from In and one from Out, and performing the corresponding move (insertion for the one from In, removal for the one from Out). The selected concentrators are then removed from In and Out. This process continues until both In and Out are empty. If the solution obtained after applying the moves is better than the initiating solution we use the new solution in the following iterations, we keep using the initiating solution otherwise. Whenever a new solution is found, it is tested for membership in the nondominated set (an external archive Y_{PN}).

2.4 Skip list

In our implementation, we maintain an external archive Y_{PN} to keep track of the nondominated points. The data structure we chose to represent that archive is the skip list. We use an implementation of skip lists for Julia[1].

[1] `https://github.com/kernelmethod/SkipLists.jl`

Chapter 3

Numerical experimentation

To present the results of our work we decided to focus on one instance on this paper. Know that if you are interested in the full results they are available on the GitHub repository for this project [1].

The instance selected is `medium3` as our Scatter Search algorithm results were more interesting for this instance.

3.1 Exact Method

First of all to be able to observe the performance of our algorithm we implemented an exact resolution using the Julia package MultiObjectiveAlgorithms and Gurobi.

Here we observed a first drawback to our problem variant. The objectives are very correlated as they both tend to minimize the distances between terminals and first level concentrators. As such, very few efficient solutions were returned (in most cases only one). Moreover the second objective (Pull objective) admits close objectives values, thus solutions diversification is more difficult. The latter issue also applies for the Scatter Search algorithm and, intrinsically, exploration and intensification methods are less efficient to get close to an optimal solution as very few efficient solution can be observed.

3.2 Components

The different components of SS gives results that seems adequate at first look.
Here is a graph that show the solutions returned for each component:

[1] `https://github.com/Anzury/MOMeta_M2_ORO`

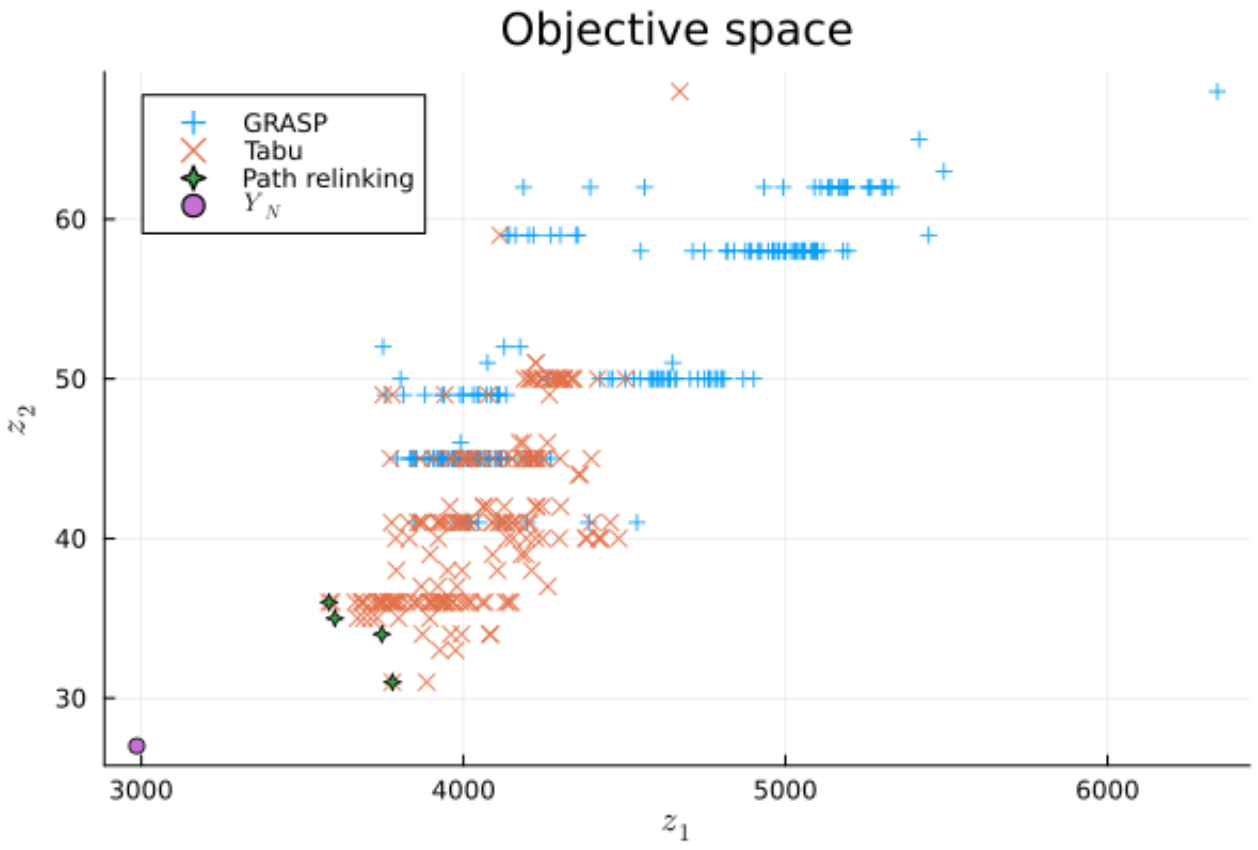


Figure 3: Solutions returned for *medium*3 by each component of SS

As we can see GRASP generate an original population with some diversity even if as discussed in the section before it can be challenging to improve it further.

Then TS improves those solutions. Solution find themselves gathered up from the two original population given by grasp which is not ideal for PR.

Finally PR have very few improving solutions from TS.

3.3 Scatter Search

The final results after using the *SkipList* data structure to keep only the efficient solutions give a graph with very few solutions:

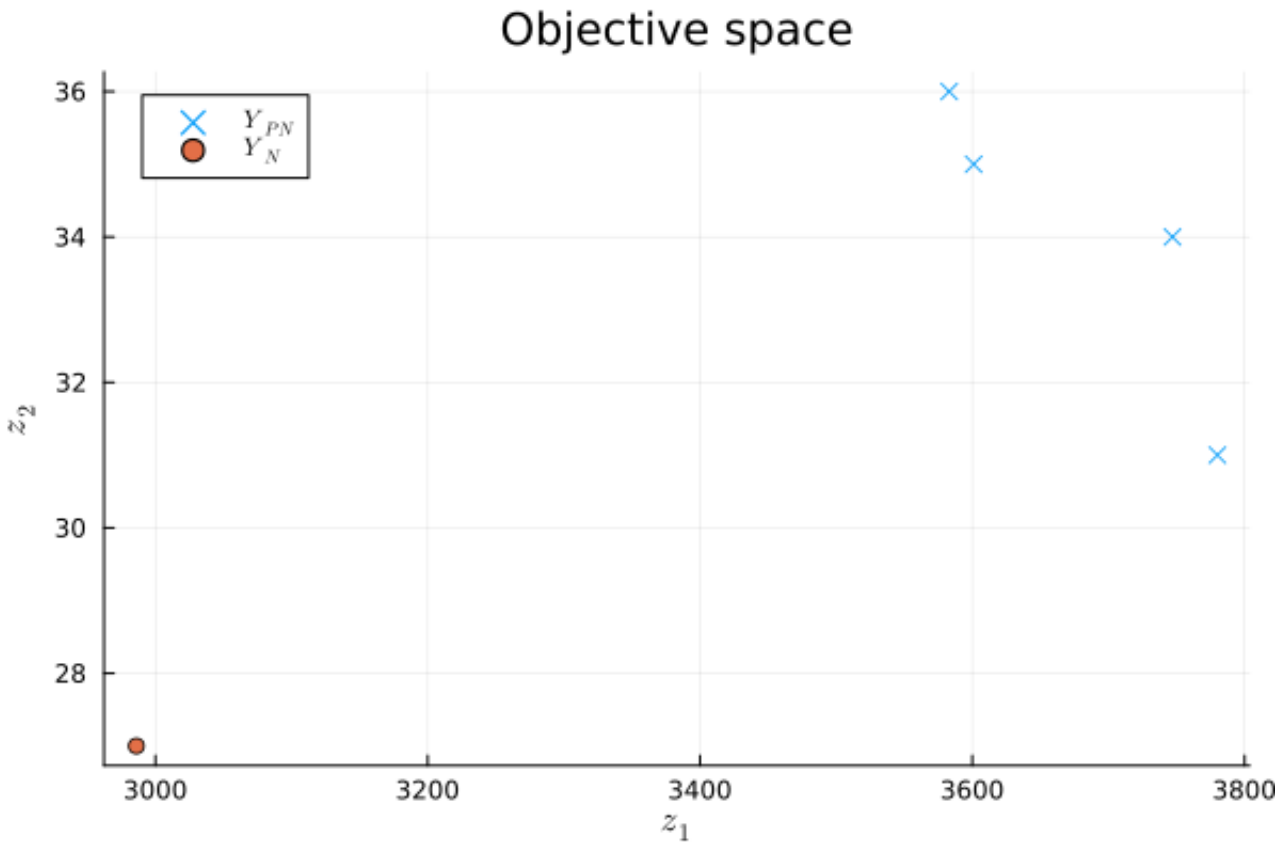


Figure 4: Final results for *medium*3 with Y_N the solutions returned by the exact method and Y_{PN} the solutions returned by SS

Further data about solutions returned can be found again on the repository, [2] Such as detailed run times and evaluation values for the results returned by SS.

3.4 Parameters Tuning

To improves the efficiency of our algorithm we ran a battery of test with different values of parameters for GRASP and TS to see if we can find values that globally return good solutions.

This study highlighted that GRASP gives good results with α set to 1 (GRASP fully random) and that the size of the tabu list seem to don't have a significant impact (outside of run times).

[2] `https://github.com/Anzury/MOMeta_M2_ORO`

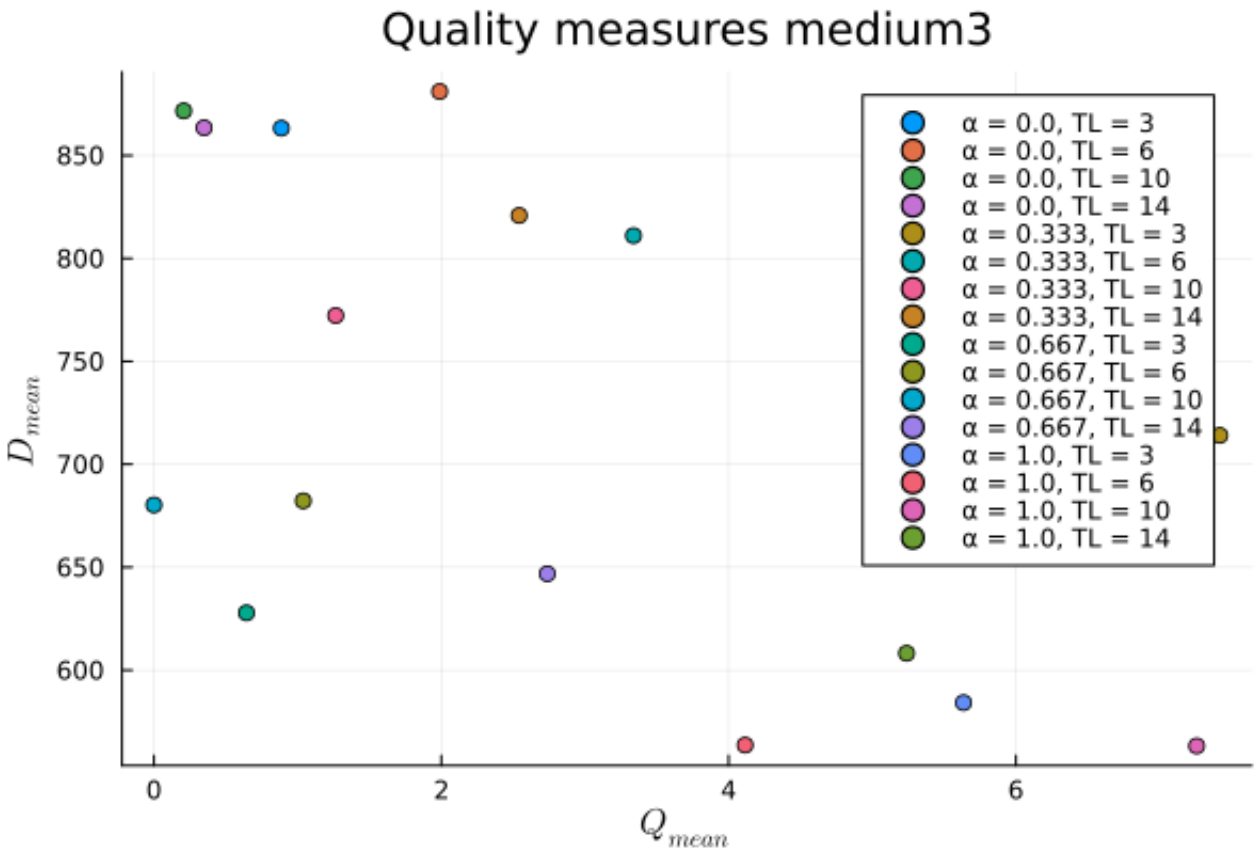


Figure 5: Let D be the distance between the ideal point of Y_N and the ideal point of Y_{PN}. Let Q be the ratio between the area defined by the ideal point of Y_N and the Y_{PN} points and, the area defined by the ideal point of Y_N and the nadir point of Y_{PN} minus the area between the area between the ideal point of Y_N and the points of Y_N. This figure show the quality measures for different values of SS parameters where D_{mean} and Q_{mean} are the mean values of quality measures D and Q obtained after several iterations.

Chapter 4

Conclusion

Study of meta-heuristics imply a consequent number of choices in the hope of achieving the best results possible. In this study we observed that in the case of multi-objectives problems the number of choices grow exponentially and that each of them need to be optimized. In this regard an algorithm such as Scatter Search which have multiple components is even more challenging to implement.
We find our implementation lacking in some area in regards to the results obtained even taking in consideration the difficulties linked to the problem structure. To improve those results some strategies could be reviewed, particularly in the GRASP and TS components to diversify the solutions returned and by doing so improving the efficiency of the PR component. The TS algorithm could also be optimized as it is by far the heaviest (in time complexity) component.
Even so this exercise proved to be interesting for several aspects such as the reflections needed to have a working algorithm on such a problem, and the link between multiple already efficient algorithms to have the best results possible.

Bibliography

[1] AD López-Sánchez, Jesús Sánchez-Oro, and Manuel Laguna. "A new scatter search design for multiobjective combinatorial optimization with an application to facility location". In: *INFORMS Journal on Computing* 33.2 (2021), pp. 629–642.

[2] Stefan Miškovic and Zorica Stanimirovic. "A memetic algorithm for solving two variants of the two-stage uncapacitated facility location problem". In: *Information technology and control* 42.2 (2013), pp. 178–190.

[3] Pierre Chardaire, J-L Lutton, and Alain Sutter. "Upper and lower bounds for the two-level simple plant location problem". In: *Annals of operations research* 86.0 (1999), pp. 117–140.

[4] Mercedes Landete and Alfredo Marín. "New facets for the two-stage uncapacitated facility location polytope". In: *Computational Optimization and Applications* 44.3 (2009), pp. 487–519.

[5] Mumtaz Karatas and Ertan Yakıcı. "An iterative solution approach to a multi-objective facility location problem". In: *Applied Soft Computing* 62 (2018), pp. 272–287.

[6] Fred Glover. "Heuristics for integer programming using surrogate constraints". In: *Decision sciences* 8.1 (1977), pp. 156–166.

[7] Thomas A Feo and Mauricio GC Resende. "Greedy randomized adaptive search procedures". In: *Journal of global optimization* 6.2 (1995), pp. 109–133.

[8] Fred Glover and Manuel Laguna. "Tabu search". In: *Handbook of combinatorial optimization*. Springer, 1998, pp. 2093–2229.

[9] Fred Glover. "A template for scatter search and path relinking". In: *European conference on artificial evolution*. Springer. 1997, pp. 1–51.