GRASP and GRASP+Path Relinking for the Maximum Diversity Problem

José Ramón Mena[a], Pau Part[b]

[a]*Degree in Mathematics, Universidad de Valencia, Valencia, Spain*
[b]*Degree in Mathematics, Universidad de Valencia, Valencia, Spain*

Abstract

In this paper, we develop heuristic procedures for solving the maximum diversity problem (MDP). The proposed algorithm is based on the GRASP + Path Relinking methodology. We will compare the solutions obtained with this algorithm with solutions obtained with a previous algorithm based on GRASP.

These procedures combine GRASP and Path Relinking to explore the solution space efficiently, achieving significant results in terms of computational time and solution quality.

Keywords: Maximum diversity problem, GRASP, Path Relinking, Heuristics

1. Introduction

The diversity problem involves selecting a subset of elements from a larger set, maximizing the differences between the selected elements according to a specific metric or criterion. This problem has applications in numerous fields, such as recommendation systems, territorial planning, bioinformatics, and experimental design, where it is important for the chosen options to reflect a wide variety or cover different relevant aspects.

Depending on the objective, diversity can be defined in several ways. Some formulations aim to maximize the sum of the distances between the selected elements, while others focus on the minimum distance between them. These definitions ensure that the chosen elements are not redundant and capture distinct aspects of the original set.

Solving this problem is complex due to its combinatorial nature, especially when the initial set is large. To address it, both exact methods, such as integer programming, and heuristic or metaheuristic approaches have been developed, providing efficient solutions in real-world problems. This has made the diversity problem a relevant topic, both for its theoretical interest and its practical impact across multiple disciplines.

Now, we will discuss the methods that we will implement and compare in this article. Path Relinking is a meta-heuristic method that, starting from a set of solutions (often named as "elite set") that are found with another method (in this case, GRASP) tries to improve those solutions. A pair of solutions is chosen from the set, one will be the initiating solution and the other one the guiding solution. The goal is to go from the first to the last by changing the nodes that are different. In each node change, the criterion is to choose the one that provides the highest objective value. Finally, among all the solutions in the path between the initiating and guiding solution, local search can be applied to some of them in search for a better solution.

GRASP (Greedy Randomized Adaptive Search Procedure) serves as the base of our approach. The procedure begins by constructing an initial solution iteratively. During this phase, good nodes are selected based on a combination of randomness and their contribution to the objective function. In each step, we evaluate the potential addition of a node, mixing the selection of high-quality nodes with random ones (this will be the paper of the alpha parameter) to balance exploration and exploitation. The alpha parameter takes part in

Once the initial solution is built, GRASP applies local search to improve it. This process systematically explores neighbours solutions by changing the nodes, often leading to significant enhancements in the solution value.

To conclude this introduction, throughout our experimentation, we will adjust different parameters so that our method provides the best possible solutions and, of course, to ensure a fair comparison between both methods. In order to do this, we will make adjustments to the alpha parameter, the parameter to set the diversity among the solutions of the elite set and time, which we will mention later.

2. Methodology

2.1. Experiment explanation

The experiment consists on comparing the quality of solutions obtained with GRASP and GRASP+Path Relinking, running the algorithms on some data instances. In particular, there are six instances of 100 different nodes where we will look for solutions with 10 nodes, and 9 instances of 500 different nodes where we will look for solutions with 50 nodes.

To begin with, we will adjust the alpha parameter of the GRASP, which is the main responsible of obtaining a great initial set of solutions that will later be improved with local search. This alpha will be used in both methods, in Grasp alone and in the Grasp part of Grasp+Path Relinking. Then, we will also adjust the difference threshold parameter for the Path Relinking that we will later explain.

2.2. Program Structure for GRASP

The heuristic implemented for the GRASP method basically follows what we advanced in the introduction. Once the input data (such as the distance matrix between elements) is provided, the program executes the following steps:

1. Initialization of Parameters: Key variables are set, such as the maxim time the program will run, the size of the solution set, and the alpha, that is the parameter to control randomness in the construction phase.

2. Construction Phase: At this stage, the program generates initial solutions using a strategy based on a Restricted Candidate List (RCL). Nodes are evaluated based on their contribution to the objective function and are selected using a mix of random and greedy criteria. This is where the $\alpha \in [0, 1]$ parameter takes part: the closer to zero, the greediest the procedure is whilst when it happens the contrary, the procedure is more random-based.

3. Local Search: After constructing a solution, the program applies an iterative improvement process. This step explores neighboring solutions to identify changes that increase the objective function value. The local search stops when no further improvements can be made.

4. Updating the Best Solution: If the solution found with local search is better than the best solution so far, it is saved as the new best solution.

5. Iteration: The program repeats the previous steps until the time limit is met.

6. Output of Results: At the end, the program outputs the best solution found, its associated objective function value, and additional statistics such as computational time. However, this last step is written with '#' in the code, because printing all the solutions in the Python console was unnecessary for the purpose of our experiment. Instead of this, we print the objective function value of the best solution in a .txt.

It is important to mention that in the

program some lines of code to automatize the comparison of the methods are there. By doing this, the program runs all the instances two times each just by clicking run.

2.3. New Method: GRASP + Path Relinking

The method proposed combines the strengths of GRASP and Path Relinking to enhance the quality of solutions for the Maximum Diversity Problem (MDP). While GRASP serves as the foundation by providing a good battery of solutions to start from, Path Relinking takes this process further by exploring intermediate solutions between high-quality candidates. This combination allows a more exhaustive exploration of the solution space, potentially discovering solutions that GRASP could have missed.

The main idea of the new method is to introduce a secondary phase after the standard GRASP process: the Path Relinking phase. This phase relies on an elite set of solutions, previously mentioned. By connecting pairs of solutions within this elite set, we can identify intermediate solutions that improve our initial set of solutions obtained with GRASP.

2.4. Elite Set Construction

The elite set plays a central role in the Path Relinking phase. During the GRASP iterations, the program identifies high-quality solutions that have a high objective function value but it doesn't takes into account the level of diversity from one another(% of different nodes). This is where the previously mentioned parameter, difference threshold, takes part. This parameter, that from now on we will call diff_thres, varies between 0 and 1 and it represents the percentage of different nodes that any solution of the elite set must fulfill, pairwise.

Then, the selection process for the elite set follows two criteria: 1. High Objective Value
2. Diversity (Pairwise)

2.5. Elite Set Construction with Pseudocode

To construct the elite set, we take all the solutions generated by GRASP and sort them from best to worst based on their objective value. The best solution from GRASP is the first to enter.

From there, we evaluate the second-best solution to determine if it is sufficiently distant from the best solution. If it meets the diversity criteria, it is added to the elite set. Here, it is crucial to carefully adjust the distance parameter, as the concept of "distance" between solutions can vary significantly depending on the problem size.

Once multiple solutions are included in the elite set, the program becomes stricter about adding new solutions. That's because as the elite set grows larger, it already contains diverse representatives with different nodes, so new solutions must meet a higher threshold to be included. This process is implemented as shown in the pseudocode below:

2.6. Pseudocode

```
sorted(all_grasp_solutions)
p=len(particular_solution)
EliteSet = []
  For candidate in sorted:
    distance = True
    For elite_sol in EliteSet:
      int= candidate∩elitesol
      if p-int < diff*p:
        distance = False
        break
    If distance:
      EliteSet.append(candidate)
    If len(Elite_Set) == max_size:
      break
```

As we have seen, the most important parameter is the difference parameter, as it dictates which solutions enter or not.

However, we include a parameter called max_size, which in our particular case will not be of high importance since we set a fixed time for the execution of the Path Relinking method. Therefore, we will set an extremely large vale for max_size so we don't restrict any possible solution to enter the elite set.

The main reason of having this parameter in our algorithm is that it is and interesting one if we are not setting a maximum running time for our method. Sometimes, mostly when the diff_thres is small, the elite set is very big, and doing all the paths among the solutions inside it can be too much, so limiting the size of the elite set it is interesting in that case.

2.7. Path Relinking Process

Once the elite set is constructed, Path Relinking is applied to pairs of solutions. For each pair, one solution is designated as the "initiating solution" and the other as the "guiding solution." The method gradually transforms the initiating solution into the guiding solution by changing nodes step by step. At each step, the change that results in the highest improvement to the objective function between all the possible, is selected. This process continues until the initiating solution matches the guiding solution.

In this case, we will work by transforming the best solutions into the worst solutions, moving in that direction. After several explorations and tests, we observed that Path Relinking provides better solutions when iterating this way. Contrary to what one might think, which would be going from the worst solutions to the best ones (as it seems like the path of "improvement"), we have observed that in that case, the method does not give as good solutions as expected. On the contrary, while moving in the opposite direction, we may discover unexplored solutions and environments where we can find better node exchanges that contribute more to the objective function, resulting in better solutions.

Finally, the last step is the application of local search to some of the intermediate solutions generated during Path Relinking. In our particular case, just to the best of each step. This ensures that even solutions obtained with Path Relinking are refined, further improving the overall performance of the method.

2.8. Program Structure for Path Relinking

The heuristic implemented for the Path Relinking method is basically what we advanced. Given the initial and final solutions to iterate between, the program executes the following steps:

1. <u>Distinct Nodes:</u> The program starts by identifying the nodes that are present in the initial solution but not in the final solution, and vice versa. These are saved as the nodes to be replaced and the nodes to be included. This step prepares the data needed for the iterative process.
2. <u>Perform the First Exchange:</u> The program explores all possible exchanges of one node for another, evaluating their impact on the solution. It selects the best exchange and updates the solution. The program then treats this new solution as the "initial solution" for the next iteration.
3. <u>Perform All Exchanges Until The Last Exchange:</u> The program continues iterating until we reach the guiding solution or time limit is reached.
4. <u>Apply Local Search to the Best Intermediate Solution:</u> After identifying the best intermediate solution found with Path Relinking, the program applies local search to it. This step refines the solution by exploring neighboring solutions and finding improvements to maximize the objective function value.
5. <u>Comparison of Solutions:</u> Finally, the solution obtained after applying local search is compared to the initial solution (which is the best solution, as we were moving from the best solutions to others with lower objective value).

The implementation is designed to minimize computational cost while exploring a diverse range of solutions. This ensures that the Path Relinking method is capable of producing high-quality solutions in an efficient manner.

Important to mention that in our code, we implement as before a method to automatize running the instances.

2.9. Key Considerations for Implementation

In summary, for the correct implementation of the methods and their comparison, we will adjust the parameters of each method through several preliminary tests and experiments, which we will show below. We will also execute both methods in similar time frames to ensure the comparison is as fair as possible. The time set for GRASP is 60 seconds, while the time set for the GRASP+Path Relinking are 20 seconds for GRASP and 40 seconds for Path Relinking. Finally, we will execute each instance twice, and we will compare the results obtained (for instances of $n = 100$, 2*6 = 12 solutions for each method and for instances of $n = 500$, 2*9 = 18 solutions for each method).

3. Results

First of all, we will adjust our parameters. In table 1 and 2, we compare some different values for parameters with the average percentage of the relative error for each type of instance. This basically represent the deviation of the solutions obtained with an specific parameter compared with the highest obtained. As said before, we have two different type of instances, ones with 500 nodes and others with 100.

As it can be seen, $\alpha = 0.75$ is the chosen one, as it gives a lower value. Following with the diff_thres parameter, we see that the chosen one is 0.2, even logic wil tell us to choose the highest value so our solutions are more diverse.

Table 1: Preliminary experiment with GRASP

α	Type I ($n = 100$), %	Type II ($n = 500$), %
0.05	0	0.2480×10^{-3}
0.25	0	0.4791×10^{-3}
0.5	0	0.4274×10^{-3}
0.75	0	0.2325×10^{-3}
0.95	0	0.4774×10^{-3}

Table 2: Preliminary experiment with GRASP+Path Relinking

diff_thres	Type I ($n = 100$), %	Type II ($n = 500$), %
0.2	0.4×10^{-3}	0.4×10^{-3}
0.4	0.16×10^{-2}	0.6×10^{-3}
0.6	0.48×10^{-2}	0.11×10^{-2}
0.8	0.137×10^{-1}	0.11×10^{-2}

In in the tables below, we see the final comparison between the two method. The deviation is the same as in the preliminary experiment, while #Best means how many time the method obtained the best solution for an instance(considering best solution as

the best obtained by both methods). So, we have to remember that, as mentioned before, for Type I there will be 12 possible solutions per each method, and for Type II, 18 possible solutions.
The results are quite equitative. For Type I instances, GRASP+PR absolutely outstands GRASP, obtaining 7 out of 12 best solution, while GRASP obtains none, and apart from that, a lower deviation. However, when looking on the results of Type II instances, GRASP obtains better solutions (7 out of 12), even though GRASP+PR obtains 2 of the best and stays really close on the deviation.

Table 3: Best constructive methods – Type I instances(n=100, m=10)

	GRASP	GRASP+PR
Deviation (%)	0.0033	0.0010
#Best	0	7

Table 4: Best constructive methods – Type II instances(n=500, m=50)

	GRASP	GRASP+PR
Deviation (%)	0.0001	0.0002
#Best	7	2

4. Conclusion

This experiment really brought into the table some interesting methodologies. GRASP+PR can be such an interesting option in some specific problems, as in the Type I instances really outstanded GRASP. However, this advantage over GRASP is not universal, as in other problems, may not be the best option to use.