Using different meta-heuristics in the quadratic assignment problem

1st Garces Angel
Computer science department. Universidad Simon Bolivar
Universidad Simon Bolivar
Caracas, Venezuela
angelgarces1248@gmail.com

2nd Guerrero Leonel
Computer science department. Universidad Simon Bolivar
Universidad Simon Bolivar
Caracas, Venezuela
leonelisaacguerrero@gmail.com

***Abstract*—The Quadratic Assignment Problem (QAP) it is know for be a very computational expensive problem. Since QAP is NP-complete, it is very difficult to be solved by exact solution methods. This paper try different methods to see witch meta-heuristics give better results in the benchmark QAPLIB. This paper presents a study on the Quadratic Assignment Problem (QAP), a combinatorial optimization problem that has many practical applications. We compare different algorithms to solve the QAP, including Local Search (LS), Iterated Local Search (ILS), Tabu Search (TS), Genetic Algorithm (GA), Ant Colony Optimization (ACO), and a Memetic Algorithm (MA) that combines GA and LS. Additionally, we propose a multi-modal approach to the genetic algorithm that runs multiple independent genetic algorithms simultaneously, and we incorporate local search into the best multi-modal combination. We test these algorithms on a benchmark of 115 QAP instances from QAPLIB, and evaluate their performance based on time, percentage of error, and number of exact solutions found. The results show that the MA is the best algorithm, with an average error of 0.33% and an average time of 34.47 seconds, while the ACO is the worst algorithm, with an average error of 54.93% and an average time of 9.31 seconds. The multi-modal approach improves the results of the GA, and adding local search further improves the performance of the best combination.**

***Index Terms*—QAP, meta-heuristics, NP-complete, local search, iterative local search, genetic algorithms, memetic algorithms**

I. Introduction

In this article we test different methods in the QAP, like local search, iterative local search, genetic algorithms, memetic algorithms. We made different modifications in the hyper-parameters of each one of them to see the best way to tackle this problem. We evaluate our results in the QAPLIB, we use as metrics the time and the percentage of error in the different test of the data-set.

The Quadratic Assignment Problem (QAP) is a well-known NP-hard combinatorial optimization problem that arises in many real-world applications, such as facility layout design, scheduling, and network optimization. The QAP involves assigning a set of facilities to a set of locations, such that the total cost of assigning any pair of facilities to any pair of locations is minimized. Despite its importance, the QAP is known to be challenging to solve optimally, especially for large instances.

Identify applicable funding agency here. If none, delete this.

To tackle this challenging problem, various heuristics and meta-heuristics have been developed over the years. In this paper, we investigate the effectiveness of six different heuristics and meta-heuristics for the QAP: Local Search (LS), Iterated Local Search (ILS) [Stü06], Tabu Search (TS), Genetic Algorithm (GA), Memetic Algorithm (MA), and Ant Colony Optimization (ACO). Specifically, we compare their performance in terms of solution quality and computational efficiency on a set of benchmark instances from the QAPLIB library.

The LS, ILS, and TS are local search-based heuristics that explore the neighborhood of a given solution by iteratively modifying it. The GA and MA are population-based meta-heuristics that maintain a population of solutions and apply genetic operators to evolve it. The ACO is a swarm-based meta-heuristic that mimics the foraging behavior of ants to construct promising solutions.

Our experimental results show that each of these heuristics and meta-heuristics has its strengths and weaknesses for the QAP. While LS, ILS, and TS are fast and effective for small and medium-sized instances, they struggle to find high-quality solutions for larger instances. GA and MA, on the other hand, can handle larger instances and are more effective at escaping local optima, but they require more computational resources. ACO, can be used to found a solution in a constructive way, despite being slower than other heuristics, especially for instances with strong spatial correlation. Also we saw that adding a multi modal process to the genetic algorithm we were able found better solutions.

Overall, this study provides a comprehensive comparison of six different heuristics and meta-heuristics for the QAP, which can help practitioners choose the most appropriate approach for their specific problem instances.

II. Heuristics and Meta-heuristics

This paper investigates the results of six different heuristic and metaheuristic algorithms for solving the Quadratic Assignment Problem (QAP). In this section, we present and explain the implementation of the six heuristics and meta-heuristics

A. Local search

The first heuristic algorithm implemented in this study is Local Search (LS), a classic optimization technique that explores the neighborhood of the current solution iteratively to improve it.

In the LS algorithm implemented, a random initial solution is generated, and then the best neighbor of this solution is found iteratively until a local minimum is reached, the algorithm explores the neighborhood of a given permutation to find the best neighbor. The neighborhood is defined as all the permutations that can be obtained by exchanging the position of two facilities in the current permutation. The cost of each solution is evaluated by the default cost function of QAP.

The implementation of the LS algorithm can be found in the following link: Local Search Implementation

B. Iterative local search

To improve the results obtained by local search, we decide to use Iterative local search, this algorithm is composed of 3 sections. First we run local search in a random perturbation, to find a possible solution, after that we make a perturbation on the found solution to run the local search algorithm again in the previous solution. To be more precise we use as:

Perturbation: We decide to use 3-opt lo make perturbation in the founded solution, we use this because is different way to found neighbor not used in the local search algorithm.

Acceptance criteria: We will stop the algorithm if we not a better solution after the perturbation .

The implementation of the iterative local search can be found in the following link: Iterative local search implementation

C. Tabu search

The second heuristic algorithm implemented is Tabu Search (TS), a metaheuristic that uses a local search algorithm to explore the search space while avoiding getting stuck in local optima by keeping track of forbidden or "tabu" moves.

The neighborhood used for the TS algorithm search heuristic is the same as that defined for LS. In the TS algorithm, the tabu list is implemented as a circular list where the perturbations are stored. Under the mentioned neighborhood, the tabu list stores the exchange made. The size of this list is fixed and its value is equal to the square root of the instance size.

The general implementation of the TS algorithm is divided into three main stages: the search process, an intensification process, and a diversification process. Each step involves handling a different type of memory, short-term, medium-term, and long-term, respectively, with the aim of escaping the local minimum found. If a better solution is obtained, the algorithm iterates again, and if the three phases give the same solution, the algorithm's execution is stopped. At the beginning of the algorithm, a random solution is generated, and from that point, the algorithm starts to run.

In the search stage, non-optimal perturbations are performed on the best solution found so far. If the new solution is in the acceptable neighborhood, which is defined as solutions that are better than the best solution found so far or the perturbation performed is not in the tabu list, it is accepted as valid. If the solution is in the acceptable neighborhood, the perturbation is stored in the tabu list, and the algorithm iterates up to three times the size of the tabu list to renew the short-term memory.

Once the neighbors in the acceptable neighborhood are iterated over, a modified local search process is performed where the n best neighbor is found. The n is reduced until it represents the best neighbor search. The first part of the process tries to escape the local minimum, and the second part follows the strategy of escaping the minimum to find another local minimum. This two-stage process of the search stage is performed three times, and in each time, the final tabu list is stored in the medium-term memory to be used in the second stage. If a better solution than the best one found is found, the short-term memory is deleted, and the algorithm is iterated over again.

In the intensification stage, if the attempt to get out of the local minimum has not found a better solution, we proceed to perform the intensification process. The three tabu lists stored in the medium-term memory are used to find the most common perturbations and generate from the best solution found so far a new solution applying these common perturbations to find a better solution using the most common features of the best solutions. If a better solution is found, the short and medium-term memory is deleted, and the algorithm is iterated over again. If no better solution is found, we proceed to the intensification stage.

In the diversification stage, we make use of the long-term memory which contains all the tabu lists generated so far. We apply all the features that are not common among all the tabu lists to find a different solution and one that is far from the search space.

The implementation of the TS algorithm can be found in the following link: Tabu Search Implementation

D. Genetic algorithm

For our implementation of genetic algorithms we use as a genotype a permutation and as genotype the QAP cost (the value that we are minimizing). Also we decided to use a population size of 100 individuals and 60 generations. The first population is generated randomly. The individuals have always a 50% chance of being a father. After each generation only the 100 individuals with lower QAP cost will survive the generation.

For the mutation of the individuals we chose a 10% chance of being muted for each child of every two possible parents, if the individual is chosen to be muted, he have an uniform probability to be muted in each of the following methods:

- two-swap: This mutation operator consists of swap randomly two positions in the permutation

- insertion: He make a cyclical swap between two points in the permutation
- random-mess: We pick two random positions in the permutation and we reorder the positions randomly
- reverse: This operation consists of taking 2 positions in the permutation and revert the order of the subsegment.

Also the amount of time we make the selected mutation operation will increase by 1 each 4 generations. For we crossover between two fathers we also have 4 options chosen randomly:

- random: The i-th gen will be taken from any of the two parent if it's not in the child
- random_P1_P2: The pick some positions in the gens of P1 and then complete with the gens of P2 in the same order that they are.
- two-points: We select a random subsegment of P1 and then we complete with the gens of P2.
- partially-mapped-crossover: We start mapping each position from the genes of both parents, Then we randomly select some subsequence from the first parent, we grab it and copy it in the child. After that we fill the child with the not used genes in the same subsegment but in the parent 2, Finally we fill the rest of position with the parent two.

The implementation for the genetic algotihm can be found in the following link: Genetic algorithm implementation

E. ant colony optimization

We decide to represent the ants as a permutation, and as a neighborhood the ants have all possible assignments of the cities in the graph of nodes in the QAP. We make some tests with the amount of ants in each iteration of the algorithm and the amount of time we reset the population.

The QAP cost was used as heuristic to assign the pheromone trail, if currently the ant has a particle cost x, then she will leave 1/X pheromone when we assign a city to a specific node. We represent all possible assignments of pheromones as a matrix where the position i,j represent the assignation of the i-th city to the j-th node, also we choose as evaporation ratio 50%.

We chose where a specific ant will move in a specific moment of time, with a discrete distribution probability based in the pheromone matrix

F. Memetic algorithm

This algorithm was performed by transforming the genetic algorithm previously explained, where many of its characteristics were maintained, the changes that were made were that the population was changed from 100 individuals to 120, because it was required that the size of the population was a multiple of 4, however the number of generations was kept at 60, likewise the way to generate the initial population was maintained (randomly), the probability that each individual of the population has of being a parent and the way to obtain the new generation. The way in which the algorithm was transformed from genetic to memetic was to perform a recombination and improvement of the solutions, against the position of the genetic crossover and mutation.

The recombination was implemented by taking four parents and by means of two recombination operators, the set of parents was partitioned into groups of four, where the two recombination operators were applied to each group of four. However, it should be noted that although they were implemented with that intention, they were also implemented so that the recombination would produce a valid solution, therefore, the implemented operators do not always fulfill their feature, for example, in case the element chosen by the operator for the solution has already been chosen, a different one is taken so that the solution found is a valid permutation.

- Respectful operator: this takes four parents and produces a descendant, where the most common characteristics are chosen from among all parents, that is, iterating through each index of the parents and looking for the most common number for each index and this will be the one that will appear in the descendant, however, if the most common number has already been chosen previously, another number is chosen to have a valid permutation.
- Assorted operator: this takes four parents and produces three descendants, where some of the characteristics are chosen among all the parents, that is to say, iterating through each index of the parents and choosing some number for each index and this will be the one that will appear in the descendant, however, if the selected number is already chosen previously, another number is chosen to have a valid permutation.

Apart from the recombination, an improvement process is applied to the offspring, first, 50% of the offspring are arbitrarily chosen and to this percentage the improvement algorithm is applied, the algorithm chosen was the LS algorithm previously explained. After applying this process, we proceed to obtain the new generation by obtaining the cost of each individual in the population, ordering them by this cost and keeping the size of the initial population.

The implementation of the Memetic algorithm can be found in the following link: Memetic Algorithm Implementation

G. Multi modal process for the genetic algorithm

To improve the results that we found in the genetic algorithm we try to have multiple independent genetic algorithms running at the same time, they are connected as if they were in a circle. We test multiple population sizes for the genetic algorithms and the number of iterations to allow communication between the different populations.

After some test we reach to improve the solutions found by the genetic algorithm, but after the improvements found in the memetic algorithm we thought that adding local search for the top 5 individuals with better QAP cost in the population will be good idea, so we did this in the best multi modal combination found (changing population size and steps to communication).

The implementation of this algorithm can be found in the following link: Multi modal implementation

H. Teacher student

For this meta heuristic we take inspiration in the classic student teacher relationship, where the student assists multiple courses with different teachers, and in each of these classrooms the student should learn something.

Each student and each teacher will be a permutation (a possible solution for the QAP), we decide to have 100 iterations where in each one of this we generate a random set of students (100 students in our case) and these students will see a class with all the teachers, and the students will try to learn for each one of the teachers.

The learning method that we use consist of taking a position p in the teacher and assign permutation[p] of the teacher to the permutation[p] of the student, since we need to have a permutation, we swap the position p with the position the permutation[p] of the teacher appear.

The implementation can be found in the following link teacher-student

III. Results

Each of algorithms described in the previous section were implemented in c++, to be tested in the benchmark (he QAPLIB), for each tested we register three main metrics, the time for each algorithm, the percentage of error with the right answered and the number of times the solver found the exact solution.

The test consist of 115 problems with solutions in the benchmark, and because of the size of the input we chose the set as a time limit for each algorithm of 20 minutes. Also all the time was measured in milliseconds.

TABLE I
Results of the benchmark

Algorithm	Time	Error	Exact
LS	17474.87 ± 44098.41	12.09 ± 21.47	8
ILS	27711.92 ± 78783.05	7.09 ± 11.21	13
TS	14193.46 ± 54927.79	5.83 ± 9.86	16
GA	31559.30 ± 57144.79	5.87 ± 9.36	22
GA + MMP	165501.99 ± 138652.55	2.52 ± 5.36	39
GA + MMP + LS	441890.78 ± 436405.13	1.07 ± 3.67	66
ACO	9305.31 ± 41159.78	54.93 ± 82.27	1
MA	34470.64 ± 95598.80	0.33 ± 1.19	69
TSS	1026.36 ± 207.89	58.70 ± 294.54	2

As we can see in Table I, the best algorithm is the Memetic Algorithm, with an average error of 0.33% and an average time of 34470.64 milliseconds. The worst algorithm is the Ant Colony, with an average error of 54.93% and an average time of 9305.31 milliseconds. The worst algorithm in time is the GA + MMP + LS for all the calculation that are needed to give an answer.

We can note that adding the multi-modal process (MMP) to the genetic algorithm improves the results, and also making an hybrid algorithm with the MMP and the local search (MMP+LS) improves the results even more.

IV. Conclusions

In this paper we describes the development of several metaheuristic algorithms for the Quadratic Assignment Problem (QAP), including Local Search (LS), Iterated Local Search (ILS), Tabu Search (TS), Genetic Algorithm (GA), Ant Colony Optimization (ACO), and Memetic Algorithm (MA). We also propose a Multi-Modal Process (MMP) to improve the results of the GA and a hybrid algorithm that combines the MMP with Local Search (MMP+LS).

The algorithms were tested on a benchmark of 115 QAP instances from QAPLIB, and the results were evaluated based on three metrics: time, percentage of error with the right answer, and the number of times the solver found the exact solution. The Memetic Algorithm was found to be the best-performing algorithm, with an average error of 0.33% and an average time of 34470.64 milliseconds, while the Ant Colony algorithm was the worst-performing, with an average error of 54.93% and an average time of 9305.31 milliseconds.

In conclusion the proposed metaheuristic algorithms are effective in solving the QAP, and that the Multi-Modal Process and hybrid algorithms improve the performance of the Genetic Algorithm.

References

[Stü06] Thomas Stützle. “Iterated local search for the quadratic assignment problem”. In: *European Journal of Operational Research* 174.3 (2006), pp. 1519–1539. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2005.01.066. URL: https://www.sciencedirect.com/science/article/pii/S0377221705002651.