ACO variants applied to the quadratic assignment problem

Aurélien van Delft

Abstract—**We applied two variants of the Quadratic Assignment Problem. We tuned their parameters and run different experiments on their results. We found that both Max-Min Ant System and Best-Worst Ant System display similar results. We introduced a local search procedure, Robust Tabu Search to Max-Min Ant System. We found that the results were significantly better.**

I. INTRODUCTION

One of the best studied NP-Hard algorithm is the *Travelling Salesman Problem*. The *Travelling Salesman Problem*, or TSP for short, is expressed as the following: given a complete graph G = (V,E) and a cost matrix A such that a cost $A_{i,j}$ is associated to each edge pair (i,j), find a Hamiltonian path of minimum cost. There are many possible applications to the TSP, one of such is *Integrated Circuit insertion*, for which the problem of inserting chips on a board is formulated as a TSP [14]. While being NP-Hard, TSP is actually a fairly simple problem to solve, and there are many heuristics that can provide really good solutions for large instances [11].

Many other problems can be seen as an extension of the *Travelling Salesman Problem*. Here, we focus on one of them, the *Quadratic Assignment Problem*. QAP is essentially defined the same way as TSP, only there is an additional flow matrix. It can be interpreted as follow: given a complete graph G = (V,E), a set of locations of size n, a distance (cost) matrix A and a flow matrix B, find an assignment mapping locations to nodes of minimum cost (we define the QAP more formally in the next section).

QAP has multiple applications, the most obvious being the assignment of facilities to locations. Another, less obvious one, is to find an "efficient" assignment of keys on a computer keyboard. But the real interest of QAP lies somewhere else. In contrast to TSP, QAP is very hard to solve. While we can easily find TSP instances of size bigger than 1,000, QAP instances are usually smaller than 100[1]. This computational complexity makes QAP very interesting for assessing metaheuristics.

In the following sections, we use two metaheuristic for solving QAP instances. These two metaheuristics are variants of the *Ant Colony Optimization* developed by M. Dorigo [15]. *Ant Colony Optimization* is a mathematical representation of the behaviors of ants. If one observes an ant colony, one can see that when the ants are foraging for food, they leave a pheromone trail to guide the search of the other ants. When the system reaches an equilibrium, a majority of ants follow the same path, further reinforcing the pheromone trail. ACO uses the same ideas replacing real ants with artificial ones. Multiple variants of ACO have been developed. Here, we focus on *Max-Min Ant System* [6] and *Best-Worst Ant System* [4].

In addition to the two metaheuristic, we also study the addition of a local search strategy to ACO variants. A local search strategy is a family of metaheuristics that guide their search process using only local information. Again, multiple local search algorithms exist.

[1]See http://anjos.mgi.polymtl.ca/qaplib/inst.html and http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html for examples of QAP and TSP instances

We chose *Robust Tabu Search* [7] for reasons we give later on.

II. Quadratic Assignment Problem

A. Formulation

The *Quadratic Assignment Problem* is defined as finding an assignment of facilities to locations with minimal cost. More formally, given n locations, n facilities, a distance matrix A such that a distance is associated to each $(location_i, location_j)$ pair and a flow matrix B such that a flow is associated to each $(facility_r, facility_s)$ pair, find an assignment of facilities to locations such that the sum of the products between flows and distance is minimal [6]. It can be mathematically formulated as

$$f(\phi) = \sum_{i=1}^{n} \sum_{j=1}^{n} b_{\phi_i \phi_j} a_{ij} \tag{1}$$

With $b_{\phi_i\phi_j}$ the flow between facility ϕ_i and facility ϕ_j, a_{ij} the distance between location i and location j, ϕ a permutation over the set of integers $\{1, \ldots, n\}$ (i.e. an assignment of facilities to locations) and ϕ_i gives the facility at location i [6].

Intuitively, the objective of the metaheuristics will be to find a permutation ϕ such that $f(\phi)$ is minimal, or at least very close to the best solution found so far.

B. Delta evaluation

What we present in this section is only used by the local search procedure. However, we introduce it here since it is closely related to the formulation of QAP.

Upon seeing the objective function for QAP (Eq.1), one can check that the full evaluation of a permutation is in $O(n^2)$. This is not an issue for *MMAS* or *BWAS* since they are constructive heuristics, but this is not the case for perturbative heuristics, such as *Robust Tabu Search.* A perturbative heuristic starts from an initial solution and carries out a sequence of perturbative steps in order to find a better solution (usually reaching a local optimum). Each perturbative step requires the evaluation of a permutation so using Eq. 1 for this purpose is quite inefficient [9].

In practice, it would be beneficial to only compute the delta cost associated to each perturbative step. Starting from the initial solution, the final cost of the heuristic's procedure can be retrieved by summing the delta costs with the initial cost.

The way the delta costs are computed depends on the structure of the distance and flow matrices. If both are symmetric, then the delta costs are easier to compute. The same goes if they have zero diagonal. Here, however, we present the most general case, as we did not make any assumption on the structure of the QAP instances. These formulas are taken from [7], [9] and [10].

The delta cost associated to the permutation of facilities r and s, starting from permutation ϕ can be computed in $O(n)$ as follow

$$\begin{aligned}
\Delta(\phi, r, s) = {} & (a_{ss} - a_{rr})(b_{\phi_r\phi_r} - b_{\phi_s\phi_s}) \\
& + (a_{sr} - a_{rs})(b_{\phi_r\phi_s} - b_{\phi_s\phi_r}) \\
& + \sum_{k=1, k\neq r,s}^{n} ((a_{sk} - a_{rk})(b_{\phi_r\phi_k} - b_{\phi_s\phi_k}) \\
& + (a_{ks} - a_{kr})(b_{\phi_k\phi_r} - b_{\phi_k\phi_s}))
\end{aligned}$$

Let π be the permutation obtained after swapping facility r and s, starting from permutation ϕ. If one takes care of storing the delta costs, and for u, v different from r, s. Then, the delta cost associated to swapping u and v can be computed in constant time as

$$\begin{aligned}
\Delta(\pi, u, v) = {} & \Delta(\phi, u, v) \\
& + ((b_{\phi_r\phi_u} - b_{\phi_r\phi_v} + b_{\phi_s\phi_v} - b_{\phi_s\phi_u}) \\
& \times (a_{su} - a_{sv} + a_{rv} - a_{ru})) \\
& + ((b_{\phi_u\phi_r} - b_{\phi_v\phi_r} + b_{\phi_v\phi_s} - b_{\phi_u\phi_s}) \\
& \times (a_{us} - a_{vs} + a_{vr} - a_{ur}))
\end{aligned}$$

Using this two formulas, the initial delta costs for all neighboring solutions can be computed in $O(n^3)$ while further delta costs are computed in $O(n^2)$.

III. Landscape analysis

An interesting analysis related to the hardness of a given problem is the *Landscape analysis*. The *landscape* is defined as a tuple containing the search space, the evaluation function and a neighborhood relation. Details about *landscape analysis* can be found in [11].

In this section, we focus on two features of the landscape of QAP, the *Fitness Distance Correlation* and the *Landscape Ruggedness*. *Fitness Distance Correlation* can be seen as a measure of the guidance provided by the evaluation function w.r.t. better solutions. *Landscape Ruggedness* is a way of defining the variability in the evaluation function for neighboring positions in the search space. Here, we do not go into the details of the analysis, as measuring *FDC* and *Ruggedness* typically induces empirical analysis. We simply report the resulting conclusions of the *Landscape Analysis*. Interested readers are redirected to [2], [5], [6] and [16] for in-depths analyses.

For the *FDC*, it has been reported that the correlation coeffecients for QAP are close to 0. This indicates that the evaluation function provides low guidance towards better solution, at least lower guidance than for the TSP [6]. The *Ruggedness* depends, as the *FDC*, on the instances of QAP. It has been reported that for a particular class of instances, QAP instances are very rugged and hard to search [5].

In general, the search space of a QAP instance is harder to search than the search space of a TSP instance. This seems to agree with the experimental results where TSP instances of size +1000 can be solved efficiently.

IV. Implemented Heuristics

Ant Colony Optimization variants are based on the same algorithmic design. A colony is a set of artificial ants. Each ant will construct a solution to the target problem. Then, ants will deposit pheromones on some solution components. This procedure iterates for a given number of rounds.

In the case of QAP, a solution is a permutation ϕ of the set $\{1, \ldots, n\}$. The indices of this permutation correspond to the location ids and the values ϕ_i correspond to the facility ids.

Ants construct a solution using a probability rule that defines the probability of assigning a facility k to a location i. In the original version of *Ant System*, this probability rule both depends on the pheromones and on a heuristic information.

The pheromones are updated using a two-steps process. The first step is that each (or a subset) of ants will deposit a number of pheromones on their solution components. The second step is an evaporation process to avoid too much exploitation.

The heuristic information used here is taken from [1]. Intuitively, if one looks at the definition of the QAP, one can see that it should be better to assign two facilities with a high flow between them to close locations and to assign two facilities with a low flow to far away locations. The heuristic information captures these ideas. The first computation step is to compute two vectors f and d. The element d_j represents the sum of the distances from location j to all other locations. The element f_i represents the sum of the flows from facility i to all other facilities. Then a matrix E is computed as $E = f.d^T$, where $e_{ij} = f_i.d_j$. The heuristic value of assigning facility i to location j is then $n_{ij} = \frac{1}{e_{ij}}$ [1].

The probability rule for both heuristics is defined the same way. The next unassigned facility i is taken randomly from the set of unassigned facilities. Then, this facility is assigned to a free location j with a probability given by [1]

$$p_{ij} = \frac{[\tau_{ij}(t)]^{\alpha} \cdot [n_{ij}]^{\beta}}{\sum_{l \in N_i^k} [\tau_{il}(t)]^{\alpha} \cdot [n_{il}]^{\beta}} \tag{2}$$

with $\tau_{ij}(t)$ the pheromone trail at iteration t, α and β two parameters defining the weight of the pheromone trails and the heuristic informations and N_i^k the set of free locations neighboring node i.

The pheromone evaporation is also defined the same way as

$$\tau_{ij}(t+1) = (1-\rho).\tau_{ij}(t) \tag{3}$$

where ρ is a parameter defining the strongness of the evaporation.

A. *Max-Min Ant System*

Max-Min Ant System is a variant of QAP introduced by Stützle et Hoos in 2000. MMAS differs from standard AS by four principles. The first change is that the pheromone deposit process becomes elitist. Indeed, in MMAS, only the best ant can deposit pheromones on its solution components. This increases the exploitation of past information. The best ant can be understood in two ways, either it is the iteration best ant or the global best ant. According to [6], iteration best provides better exploration so we chose to let the iteration best ant deposits pheromones.

Another change is that the pheromone trails are limited to a range $[\tau_{min}, \tau_{max}]$. This avoids the stagnation of the search. The two limits can be computed in different ways. Following [6], we let $\tau_{max} = \frac{1}{\rho} \cdot \frac{1}{bestScore}$, where bestScore is the score of the best solution found so far and $\tau_{min} = \frac{\tau_{max}}{2n}$.

The third change is the initialization of the pheromones trails. The trails are initialized at τ_{max}. The idea is to perform more exploration at the start of the algorithm. However, given the way we defined τ_{max}, we can see that there are no BestScore at the start of the algorithm. So, we initialized the pheromones to $\rho_0 = 1$.

The last change is a restart procedure. Once the search process stagnates for a given number of rounds, the procedure is restarted by setting the pheromones to τ_{max}. Since all the changes are related to the pheromones, and since the three last are about pheromones limits, the only "real" change is about the pheromone deposit rule, which is defined MMAS as

$$\tau_{ij}(t+1) = \text{evaporatePheromone} + \Delta\tau ij^{best} \tag{4}$$

with $\Delta\tau_{ij}^{best}$ defined as

$$\Delta\tau_{ij}^{best} = \begin{cases} \frac{1}{f(\phi^{best})} & \phi^{best}(j) = i \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

MMAS, as it is defined in [6] and [1], usually removes the heuristic information from the probability rule. This is because, if a local search procedure is added to MMAS, this heuristic information is not necessary anymore. However, since one of our purpose is to study the impact of introducing local search to MMAS, our basic implementation of MMAS does not use a local search step. Because of that, we did not change the probability rule defined by Eq.2.

B. *Best-worst Ant System*

Best-worst Ant System is another variant of ACO. It was developed by Cordón et al. in 2000[4]. BWAS shares some common ideas with MMAS, and takes also ideas from the *Population Based Incremental Learning*[17] algorithm. The common ideas with MMAS are the restart procedure when the algorithm stagnates and the elitist pheromone deposit.

The deposit rule for BWAS is the same as for MMAS (Eq.4), i.e. only the iteration-best ant can deposit pheromones on its solution components. The only change in the pheromones update procedure is that BWAS records the iteration-worst ant, the ant having constructed the worst solution for the current iteration. Then, the pheromones of the worst ant's solution components are evaporated two times instead of one, except if they are also in the iteration best solution. Also, BWAS does not use pheromones trails limits, in contrast with MMAS.

The PBIL algorithm is a kind of genetic algorithm. As such, it defines a mutation operator, as well as other components. This is what inspired BWAS' creators to introduce a mutation operator to ACO. Each pheromone has a probability μ to mutate. This introduces more exploration from the ants. Formally, this mutation operator is defined as [3]

$$\tau_{ij}^{'} = \begin{cases} \tau_{ij} + \text{mut(it, threshold)} & , if a = 0 \\ \tau_{ij} - \text{mut(it, threshold)} & , if a = 1 \end{cases} \quad (6)$$

with a a value in $\{0, 1\}$, it the current iteration number and $threshold$ being the average pheromone value of the iteration best solution components. The mut(.) operator is defined as mut(it, threshold) $= \frac{iter - iter_R}{N_{iter} - iter_R}.\sigma.threshold$, with iter$_R$ the last iteration number where a restart occurred, N$_{iter}$ the maximum number of iterations and σ a parameter.

Except from this mutation procedure and the worst ant update, every other component of BWAS is the same as the component of MMAS.

C. Robust Tabu Search

As reported multiple times ([1], [6] for instance), it is beneficial to constructive meta-heuristics to add a local search procedure to the construction process. The performances of an ACO procedure can increases if each ant performs a subsequent local search after its solution has been constructed.

There are many local search procedure (e.g. *first-improvement*, *Variable Neighborhood Search*, *Simulated Annealing*, *Tabu Search*, etc.). The choice of a local search procedure is guided by multiple criteria. First, since this procedure is supposed to be run by each ant, this procedure should be relatively efficient. Also, given the complexity of QAP instances and the associated landscapes (see section III), too simple local search procedures, such as first and best improvement, might not explore the search space sufficiently.

According to [18], *Tabu Search* heuristics produces good results for the *Quadratic Assignment Problem*. Also, *Tabu Search* can be implemented efficiently [8]. This led us to choose a *Tabu Search* variant as the local search procedure. Before explaining *Robust Tabu Search*, we give a brief review of the concepts of *Tabu Search*.

Tabu Search firstly defines a neighborhood relation to produce a set of adjacent solutions. Usually, this neighborhood relation is a 2-exchange neighborhood, which is what we chose here. A 2-exchange move swaps two facilities (or two locations) of a given permutation ϕ. Then, it evaluates the neighbors of the current solution, choosing the best-improving neighbor as the next solution. If there are no improving neighbors, it choses the one that least degrade the solution [7].

To avoid stagnating in a local minimum, *Tabu Search* forbids a move if it has been chosen in the last tt iterations. This tt parameter is called the *tabu tenure*. However, since some moves might always lead to good solution, an aspiration criterion is introduced to allow a tabu move to be chosen.

We did not choose the original *Tabu Search* because its results depend on the value of tt, and also because there are other, more performing variants [18]. One of them is *Robust Tabu Search*. RoTS was developed by Taillard in 1991 [7].

RoTS defines the tabu moves using a 2-dimensional array T. An element (i,j) of this array indicates the last iteration where facility i was assigned to location j. Swapping the facility r with facility s is tabu if $iter - T(r, loc_r) < tt$ and $iter - T(s, loc_s) < tt$, with iter being the current iteration number. Using an array allows to check if a move is tabu in constant time.

In the original paper, RoTS uses a aspiration criterion slightly different than TS. In TS, a move passes the Tabu condition if the resulting solution leads to the best solution so far. In RoTS, an additional criterion is set. If a move has not been chosen in the last t iterations, with t relatively large, then this move passes the Tabu condition. However, here we implemented TS aspiration criterion. The reason for that is that t is set to relatively large value [7] as RoTS is expected to be run for a lot of iterations when it is the main heuristic. In our case, RoTS is used by ants as a subsequent procedure. Thus, we decided to limit the number of iterations performed by RoTS to accelerate the search process. In this case, this number of iterations was too small for t to have an impact, so we removed it. Another reason is that it reduces the number of parameters to tune.

The last change introduces by RoTS is a random, dynamic tabu tenure tt. The tabu tenure, defining the number of iterations before a move becomes authorized, is bounded in RoTS by $[tt_{min}, tt_{max}]$. During the run, this tabu tenure is changed every $2.tt_{max}$ [7]. Following Taillard, we set $tt_{min} = \lfloor 0.9n \rfloor$ and $tt_{max} = \lceil 0.9n \rceil$. Thanks to this definition of the tabu tenure, using simplified RoTS does not increase the number of parameters to tune.

D. *Parameters tuning*

Each metaheuristic algorithm has a number of parameters. The original ACO had the number of ants, the initial pheromone value, the value of α, the value of β and obviously the number of iterations. Finding a good set of parameters is crucial to have a very good algorithm. One way of finding this set is to rely on automatic tuning. Automatic tuners takes an algorithm and a set of tunable parameters and returns a (good) set of parameter values. These tuners can based on heuristics or machine learning algorithms. Among the existing tuners, we chose the irace package [12], [13]. We don't go into the details of this package. We simply give the fixed and tunable parameters as well as the best parameters for each heuristic.

The fixed parameters were the maximum number of tours (10000), the initial pheromone (1.0) and the restart threshold (restart after 5 iterations). The number of iterations of the Robust Tabu Search was fixed and set to 500. For MMAS and MMAS+RoTS, the tunable parameters can be seen in table I. We allowed the tuner to perform 1000 experiments.

Name	Type	Domain
numAnts	categorical	(5,10,15,20,25,30)
β	real	(0.2, 3.0)
α	real	(0.2, 3.0)
ρ	real	(0.1,1.0)

Table I: Tunable parameters for MMAS and MMAS+RoTS

For BWAS, we allowed the tuner to perform 1500 experiments. We allowed more experiments for BWAS since it has two additional parameters, as can be seen in table II. As a recall, μ is the pheromone mutation probability and σ a weight for the mut(.) operator.

The best parameters for MMAS were $numAnts = 25$, $\beta = 1.753$, $\alpha = 1.4872$, $\rho = 0.5819$. For MMAS+RoTS, $numAnts = 30$, $\beta = 2.5555$, $\alpha = 1.9819$, $\rho = 0.7382$. And for BWAS, $numAnts = 25$, $\beta = 1.1459$, $\alpha = 0.315$, $\rho = 0.4545$, $\sigma = 6.0198$, $\mu = 0.1593$.

Name	Type	Domain
numAnts	categorical	(5,10,15,20,25,30)
β	real	(0.2, 3.0)
α	real	(0.2, 3.0)
ρ	real	(0.1,1.0)
σ	real	(0.5,8)
μ	real	(0.05,0.25)

Table II: Tunable parameters for BWAS

V. Results

A. *MMAS vs. BWAS*

To assess the performances of both heuristics (MMAS and BWAS), we performed 10 runs for each instances for each algorithm. We assigned a specific seed to each one of these 10 runs. We used python to compute these seeds [2].

For a given instance, we saved the best solution score, worst solution score, average solution score and standard deviation of the solution score. We allowed the ants to build 10000 solutions for each run.

The reported standard deviation is the square root of the corrected estimation of the variance of the solutions. We do not report the average deviation w.r.t. the opt value, as the input instances did not contain information about their optimal values. However, these values can be found on http://anjos.mgi.polymtl.ca/qaplib/inst.html.

The algorithms were run on an i7-6700K with 4.0 Ghz, 4 physical cores and 8 virtual cores (only for information as we did not use any thread in our implementation). The computer possessed 16gb of DDR4, 2133 Mhz. The results can be seen on Table III.

We can see that the results of MMAS and BWAS are close. The best solutions of BWAS are systematically better, except for tai17a and tai35a. Also, the worst solutions of BWAS are also always worse, except for chr15b and wil50. As it can be verified by observing the estimated standard deviations, it seems that BWAS displays more variance in its solutions than MMAS. This is not the case only for bur26g. This might be because BWAS has more parameters. We did not reported it here, but both algorithm run in a similar time.

[2]The seeds used for testing can be found in seeds.txt

Overall, both algorithms managed to find the optimal solutions for esc16d and had14. This seems to indicate that these two instances are the simplest ones.

To check whether or not there is a significant difference between both algorithms, we carried out a Wilcoxon-signed rank test in R using the means reported in Table III. The resulting p-value was 0.1933594. So we cannot conclude that there is a significant different between the two algorithms.

Since both algorithms take roughly the same time to run, both are suitable for solving the *Quadratic Assignment Problem*. BWAS seems to produce slightly better solution but at the expense of a bigger variance and two additional parameters. Given this fact, we would recommend to primarily use MMAS. Then, if one is not satisfied with MMAS performances for a specific instance, BWAS can be used as its best solutions are usually better than MMAS' ones.

To compare the convergence of both algorithms we selected the tai35 instance, as it is a fairly hard one. Next, we let both algorithm run for 2000 solutions and we reported the score every 25 solutions. This score is then average over the 10 runs. We can see on Figure 1 that after tour 100, BWAS's solutions are already better than MMAS ones. Also, the evolution of both algorithms is quite steady until tour 1500. At tour = 1500, their is a clear score gap between MMAS and BWAS as BWAS's scores drops abruptly. Upon seeing that, it seems that the solution improvements of MMAS are more smooth than BWAS. So, the computation effort to get to the best solution for MMAS seems lower than for BWAS, as longer runs are required to reach BWAS's best scores.

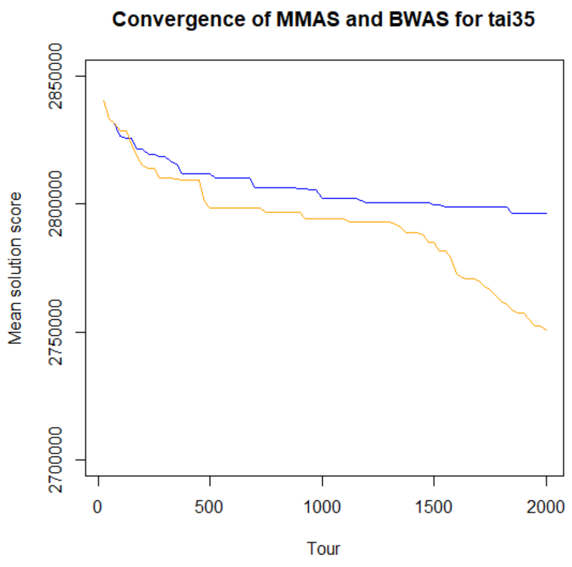

Figure 1: Convergence for tai35a. Blue line is MMAS, orange line is BWAS

B. *MMAS vs. MMAS+RoTS*

As the results of MMAS are slightly worse than BWAS, we decided to add the local search procedure to MMAS. Also, as stated in [1], the heuristic information is not really useful anymore if a local search procedure is added. This has the advantage of removing the parameter β from the heuristic set of tunable parameters. Here, however, since we wanted to solely study the impact of RoTS, we left the implementation of MMAS unchanged. So β still needed to be tuned (cf. section III.D).

Since we presented RoTS in section III.C, we directly move onto the results. We can see in Table IV that the performances of MMAS+RoTS clearly surpasses those of MMAS (and even BWAS). Not only that, but MMAS+RoTS found the optimal solution for bur26g, esc16d, had14, had20 and scr12. Also, the standard deviation of MMAS+RoTS is systematically better, except for tai35a. This seems to indicate that MMAS+RoTS produces very good solution more steadily.

We performed again a Wilcoxon-signed rank test using the mean values. The resulting p-value was 0.001953125. This indicates a significant difference between both algorithms results as the p-value is $<$ than 0.05.

A possible explanation to these performance improvements is the fact that ACO is a constructive heuristic. Constructive heuristic alone are known to not always produce good solutions [6], [11]. This is because constructive heuristics, at least ACO variants, do not use any information related to the search space. The construction procedure for ACO is completely guided by prior results + random exploration. By introducing a local procedure, the heuristic makes use of additional information regarding the search space (i.e. the neighboring solution). In the end, after the local search procedure, their is a high probability that the resulting solution is better than the one the ant initially constructed.

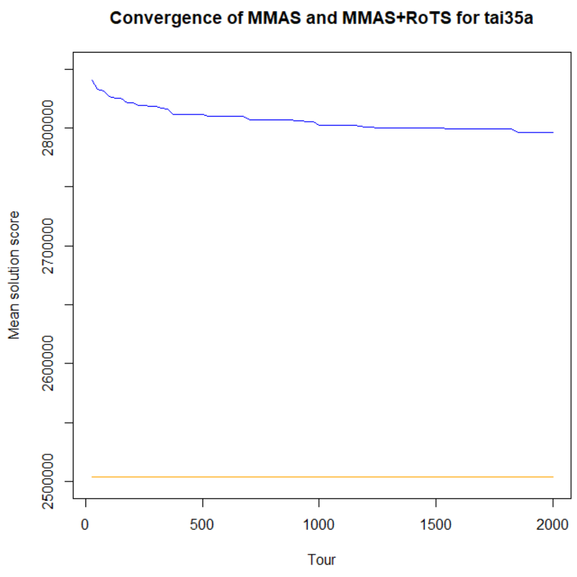

Figure 2: Convergence for tai35a. Blue line is MMAS, orange line is MMAS+RoTS

	MMAS				BWAS			
	best	worst	mean	sd	best	worst	mean	sd
bur26g	1.01278e+07	1.02201e+07	1.01642e+07	30929.5	1.01268e+07	1.02238e+07	1.01681e+07	28253.7
chr15b	12478	18504	16269	2093.24	9902	16236	13674.6	2147.71
esc16d	16	18	16.4	0.843274	16	18	16.8	1.0328
had14	2724	2748	2732.8	7.72873	2724	2772	2741	16.8721
had20	6954	7088	7007.2	38.8982	6944	7178	7000.4	69.3192
nug28	5480	5600	5558.6	34.0855	5392	5686	5584.2	88.3853
scr12	32292	34216	33083.6	564.016	32236	35532	33555	1174.46
tai17a	513648	524812	517994	4111.86	520938	539602	531411	5540.24
tai35a	2.60187e+06	2.63437e+06	2.61803e+06	8194.42	2.6085e+06	2.65658e+06	2.62814e+06	17348.6
wil50	50898	51590	51128.2	191.403	50512	51322	51023	256.427

Table III: Results for MMAS and BWAS

Obviously, MMAS+RoTS is much slower than MMAS, since each ant runs a Robust Tabu Search on its local solution. Studying the convergence for tai35a in Figure 2, we can see that MMAS+RoTS almost immediately converges to its optimum solution. Thus, the computational burden of computing one tour is much bigger but MMAS+RoTS reaches its optimum faster than MMAS.

VI. Conclusion

We presented here two ACO variants for the *Quadratic Assignment Problem*. The QAP is a very hard problem and no exact solution exists for instances bigger than 20. For these reasons, QAP is interesting for assessing metaheuristics.

The two heuristics that we implemented were *Max-Min Ant System* and *Best-Worst Ant System*. We tested both on a test set of 10 instances. The results were very close. Overall, BWAS produces slightly better results, at the expense of a bigger variance and a less smooth convergence.

Next, we introduced a local search procedure, *Robust Tabu Search* to MMAS. The results were significantly better, MMAS+RoTS finding optimal solutions for 5 instances out of 10. Also, the standard deviations of MMAS+RoTS are better and it convergences very quickly. However, one MMAS+RoTS iteration takes way more time than for MMAS.

In the end, our results suggest that it is beneficial for constructive heuristics, at least ACO variants in our case, to introduce a local search procedure in order to improve the quality of the produced solutions.

	MMAS				MMAS+RoTS			
	best	worst	mean	sd	best	worst	mean	sd
bur26g	1.01278e+07	1.02201e+07	1.01642e+07	30929.5	1.01172e+07	1.0119e+07	1.01183e+07	602.937
chr15b	12478	18504	16269	2093.24	8228	10358	9435.8	566.632
esc16d	16	18	16.4	0.843274	16	16	16	0
had14	2724	2748	2732.8	7.72873	2724	2724	2724	0
had20	6954	7088	7007.2	38.8982	6922	6930	6924.6	3.13404
nug28	5480	5600	5558.6	34.0855	5224	5310	5270.6	31.7497
scr12	32292	34216	33083.6	564.016	31410	32236	31613.4	338.623
tai17a	513648	524812	517994	4111.86	493662	506212	503034	3760.77
tai35a	2.60187e+06	2.63437e+06	2.61803e+06	8194.42	2.48579e+06	2.52492e+06	2.50404e+06	11820.3
wil50	50898	51590	51128.2	191.403	49068	49334	49188.4	81.7845

Table IV: Results for MMAS and MMAS+RoTS

REFERENCES

[1] Stützle, Thomas, and Marco Dorigo. ”ACO algorithms for the quadratic assignment problem.” New ideas in optimization C50 (1999): 33.

[2] Angel, Eric, and Vassilis Zissimopoulos. ”On the landscape ruggedness of the quadratic assignment problem.” Theoretical computer science 263.1-2 (2001): 159-172.

[3] Cordón, Oscar, Iñaki Fernández de Viana, and Francisco Herrera. ”Analysis of the best-worst ant system and its variants on the QAP.” International Workshop on Ant Algorithms. Springer, Berlin, Heidelberg, 2002.

[4] Cordon, Oscar, et al. ”A new ACO model integrating evolutionary computation concepts: The best-worst Ant System.” (2000).

[5] Merz, Peter, and Bernd Freisleben. ”Fitness landscape analysis and memetic algorithms for the quadratic assignment problem.” IEEE transactions on evolutionary computation 4.4 (2000): 337-352.

[6] Stützle, Thomas, and Holger H. Hoos. ”MAX–MIN ant system.” Future generation computer systems 16.8 (2000): 889-914.

[7] Taillard, Éric. ”Robust taboo search for the quadratic assignment problem.” Parallel computing 17.4-5 (1991): 443-455.

[8] Paul, Gerald. ”An efficient implementation of the robust tabu search heuristic for sparse quadratic assignment problems.” European Journal of Operational Research 209.3 (2011): 215-218.

[9] Stützle, Thomas, and Marco Dorigo. ”Local search and metaheuristics for the quadratic assignment problem.” Technical Report (2001).

[10] Burkard, Rainer E., and Franz Rendl. ”A thermodynamically motivated simulation procedure for combinatorial optimization problems.” European Journal of Operational Research 17.2 (1984): 169-174.

[11] Hoos, Holger H., and Thomas Stützle. Stochastic local search: Foundations and applications. Elsevier, 2004.

[12] López-Ibáñez, Manuel, et al. ”The irace package: User guide.” IRIDIA, Université Libre de Bruxelles, Belgium, Tech. Rep. TR/IRIDIA/2016-004 (2016).

[13] López-Ibáñez, Manuel, et al. ”The irace package: Iterated racing for automatic algorithm configuration.” Operations Research Perspectives 3 (2016): 43-58.

[14] CHAN, DONALD, and Daniel Mercier. ”IC insertion: an application of the travelling salesman problem.” The International Journal of Production Research 27.10 (1989): 1837-1841.

[15] Dorigo, Marco, Vittorio Maniezzo, and Alberto Colorni. ”Ant system: optimization by a colony of cooperating agents.” IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 26.1 (1996): 29-41.

[16] Weinberger, Edward. ”Correlated and uncorrelated fitness landscapes and how to tell the difference.” Biological cybernetics 63.5 (1990): 325-336.

[17] Baluja, Shumeet, and Rich Caruana. ”Removing the genetics from the standard genetic algorithm.” Machine Learning Proceedings 1995. 1995. 38-46.

[18] Taillard, Eric D. ”Comparison of iterative searches for the quadratic assignment problem.” Location science 3.2 (1995): 87-105.