The Effect of the Heuristic Information on a Hybrid $\mathcal{MAX}-\mathcal{MIN}$ Ant System for the Quadratic Assignment Problem

Augusto Lopez Dantas[1] and Aurora Trinidad Ramirez Pozo[1]

Department of Computer Science – Federal University of Paraná (UFPR)
augusto.dantas@ufpr.br, aurora@inf.ufpr.br

Abstract. The $\mathcal{MAX}-\mathcal{MIN}$ Ant System is a bio-inspired metaheuristic algorithm that has presented some of the best results for combinatorial optimization problems, specially the Quadratic Assignment Problem. But, when applied with local search, it traditionally does not use any heuristic information, since both the heuristic and the local search act as an intensification component. In this work, we compared the results obtained with and without heuristic, also analysing which pheromone update choice rule is more suitable for each case. It was possible to observe that the algorithm benefits from the use of the heuristic regarding the expended CPU running time to find good solutions.

Keywords: Ant Colony Optimization, Quadratic Assignment Problem, Swarm Intelligence

1 Introduction

Ant Colony Optimization (ACO) is a class of metaheuristic algorithms that simulate the behavior of ants on finding the shortest path between the nest and the food source [5]. This is done through the exploration of the trail of pheromone (a chemical substance) deposited by the ants during their walking.

ACOs are constructive and population-based approaches that use the learned information throughout its execution (the pheromone trail), along with a problem-dependent information, called heuristic, to guide the construction of better solutions [4]. Therefore, each ant can be seen as an agent that moves on a graph and select one edge at a time based on its desirability until the path is complete.

The first implemented ACO algorithm was the Ant System (AS) [5], which was initially applied to the Traveling Salesman Problem (TSP). Since then, several modifications and extensions to it were proposed, compounding the group of ACO metaheuristic. These algorithms have presented some of the best performances for many combinatorial optimization problems (problems with discrete solutions and a finite search space) such as the quadratic assignment, the vehicle and network routing, the sequential ordering, the graph coloring and others [4].

Among these variations there is the $\mathcal{MAX}-\mathcal{MIN}$ Ant System (MMAS), whose main differences from the original AS is that it defines upper and lower

limits to the pheromone values and that only one ant is allowed to update the learned information at each iteration. This ant could be either the iteration best or the best found so far since the beginning of the algorithm [14].

Recently, there have been some studies attempting to further improve the MMAS, like adding a reactive heuristic that provides better tour constructions in case of pheromone reinitialization [11], or simulating a colony caste system by having ants with different parameters, which helps the algorithm to adapt to a particular problem instance [9].

One of the reasons for the MMAS to become relevant is its outstanding performance on the Quadratic Assignment Problem (QAP), as shown in [14]. The QAP is one of the most challenging problems and was first proposed as a mathematical model in 1957 [8]. It consists on assigning a set of facilities into a set of locations in a way to obtain the minimal possible flux between the facilities and the minimal distance between the locations.

Several others ACOs variants were proposed to solve the QAP [13], such as the more recent Population-Based Ant Colony Optimization, that maintains an archive of solutions that are used to update the pheromone information [10], and the cunning Ant System, that uses pre-existing partial solutions in the construction process [16]. Apart from ACO algorithms, different metaheuristic approaches have already been successfully applied to the QAP, like the Simulated Annealing, Tabu Search, Genetic Algorithm, GRASP, among others [3].

According to [1], a good metaheuristic algorithm should have a balance of intensification and diversification components. The former consists in extensively scanning the local regions that contain good solutions, whereas the later is about how the algorithm moves to unexplored regions. This requirement of having both components often results in the elaboration of hybrid algorithms, that is, the combination of intensification and diversification based approaches.

In fact, most of the ACO variants proposed for the QAP make use of a local search mechanism to improve the constructed solutions [13]. Regarding the MMAS, when combined with local search, it is traditionally applied without the use of any heuristic information. The reason behind this is that the heuristic information has the role of guiding the ants at the beginning of the algorithm to avoid it to become a completely random search and to avoid the construction of initially bad paths. It is concluded then, that this behavior is overlapped by the local search, making the heuristic unnecessary [13].

Although this affirmation may be correct, the experiments to evaluate it were only made on the TSP [13, 14]. Hence, in this work we analyse the performances of MMAS with local search to the QAP (MMAS-QAP) when using or not the heuristic information. The following sections are divided like this: in Sect. 2 the QAP is formally explained. Sections 3 and 4 explain in details how the MMAS and local search algorithms work for QAP. Following, in Sect. 5 and 6 we present the performed experiments and its results, ending with a conclusion in Sect. 7.

2 Quadratic Assignment Problem

Given two sets of n activities and n locations, the goal in QAP is to assign each facility into one unique location in order to minimize the total flux and distance between the associations. The problem can be formally defined as

$$\min_{\phi \in S_n} \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} d_{\phi(i)\phi(j)} \tag{1}$$

where S_n represents all possible permutations of a set $N = \{1, 2, \ldots n\}$, f_{ij} and d_{ij} are the correspondent values in the flux and distance matrices, respectively, and the product $f_{ij} d_{\phi(i)\phi(j)}$ is the singular cost of assigning the activity i to the location $\phi(i)$ and the activity j to the location $\phi(j)$.

In the permutation $\{2, 5, 1, 4, 3\}$, for example, the activity 3 is assigned to the location 5 and the activity 1 to the location 3. Therefore, the singular assignment cost between them is given by the product $f_{31} * d_{53}$. Thus, the objective is to find a permutation ϕ that minimizes the sum of all the products.

The QAP can be applied to several real world problems, such as the typewriter keyboard design, the location of the hospital departments, the backboard wiring, and many others [3].

Because it is an NP-hard problem [12], the use of exact methods is unfeasible for instances with large sizes. Until now, the largest instance without any special property optimally solved has the size of 36 [2]. Instances with sizes of 64 and 128 were solved in [6], but benefiting on their extremely symmetric structures.

3 $\mathcal{MAX}-\mathcal{MIN}$ Ant System for QAP

In MMAS, the solutions represented by permutations, described in Sect. 2, are initially empty and one facility is selected and assigned to one location at each time according to the desirability of this association [5]. Let m represent the number of ants, then this construction process is done simultaneously by each ant_k, with $k = [1, 2, \ldots, m]$.

The assignment desirability is generally based on two informations: the pheromone trail (τ) and the heuristic (η). Each of them is represented as a matrix of dimensions $n \times n$, with n being the instance size. In this way, τ_{ij} and η_{ij} are respectively the pheromone trail and the heuristic information of assigning the facility i to the location j. So, the probabilistic choice of this association is given by [5]

$$p_{ij}^{k} = \begin{cases} \dfrac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum\limits_{l \in \text{free}_k} [\tau_{il}]^{\alpha}[\eta_{il}]^{\beta}} & \text{if } j \in \text{free}_k \ . \\ 0 & \text{otherwise} \ . \end{cases} \tag{2}$$

where free_k is the set of all free locations available for an ant k and α and β are parameters that control the influence of the pheromone and heuristic values.

To achieve the rule for the MMAS-QAP described in [14], in which no heuristic information is used, these parameters are set to $\alpha = 1$ and $\beta = 0$.

Since our goal is to observe the effect of the heuristic information in MMAS for QAP, we use the same heuristic described in [4]. In this definition, first two vectors are generated by the sum of the rows in the distance (D) and flux (F) matrices (see Sect. 2). They are called the potential vectors P_d and P_f, respectively, and represent the importance of the correspondent locations and facilities. An example of this process can be seen in (3).

$$D = \begin{bmatrix} 0 & 5 & 2 & 4 \\ 5 & 0 & 1 & 3 \\ 2 & 1 & 0 & 1 \\ 4 & 3 & 1 & 0 \end{bmatrix} \Rightarrow P_d = \begin{bmatrix} 11 \\ 9 \\ 4 \\ 8 \end{bmatrix} \qquad F = \begin{bmatrix} 0 & 50 & 60 & 94 \\ 50 & 0 & 22 & 50 \\ 60 & 22 & 0 & 44 \\ 94 & 50 & 44 & 0 \end{bmatrix} \Rightarrow P_f = \begin{bmatrix} 204 \\ 122 \\ 126 \\ 188 \end{bmatrix} \tag{3}$$

With these potential vectors we can then obtain a potential matrix H by multiplying P_d to the transpose of P_f. Following the example, this multiplication results in (4). Each column of H represents a facility and each value represents the potential cost of assigning that facility to the row location. Finally, the heuristic information is given by the inverse of the potential cost, meaning that $\eta_{ij} = 1/H_{ji}$. For now on, when the MMAS is applied with this heuristic, we refer to it as MMAS$_h$.

$$H = P_d * P_f' = \begin{bmatrix} 2244 & 1342 & 1386 & 2068 \\ 1836 & 1098 & 1134 & 1692 \\ 816 & 488 & 504 & 752 \\ 1632 & 976 & 1008 & 1504 \end{bmatrix} \tag{4}$$

The process described so far is shown in Alg. 1, in which the order of the values returned by `GetUnassignedFacility()` is random if $\beta = 0$, as in [14], and it is in descending order from P_f otherwise, according to [5]. The location returned from `GetFreeLocation()` is probabilistic according to (2).

Algorithm 1: Construct Solution

```
Function ConstructSolution()
    ant ← []
    freeLocations ← GetAllLocations()
    repeat
        facility ← GetUnassignedFacility()
        location ← GetFreeLocation(facility, freeLocations)
        ant[l] ← f
        freeLocations ← PopItem(location)
    until ant is complete
    return ant
end
```

For representing the pheromone information, another matrix with a similar structure from the heuristic is used, called the pheromone matrix (τ). Thus, the pheromone trail on assigning facility i to location j is given by τ_{ij}. After all ants construct their solutions, this matrix is updated according to the equation

$$\tau_{ij} = (1 - \rho)\tau_{ij} + \Delta\tau_{ij}^{best} \tag{5}$$

where ρ represents the evaporation rate and $\Delta\tau_{ij}^{best}$ is given by

$$\Delta\tau_{ij}^{best} = \begin{cases} 1/J_{\phi^{best}} & \text{if facility i is assigned to j in } \phi^{best} \\ 0 & \text{otherwise} \end{cases}. \tag{6}$$

Thus, the amount of pheromone deposited at each iteration is the inverse of the cost function J (given in (1) for the QAP) of the solution ϕ^{best}. This solution can be either the best of the current iteration (ϕ^{ib}) or the best-so-far since the beginning of the algorithm execution (ϕ^{gb}).

The main difference introduced by MMAS is the maximum (τ^{max}) and minimum (τ^{min}) limits imposed to the allowed values of pheromone. This is intended to prevent search stagnation, where all ants produce the same solution due to a high predominance of that path in the pheromone matrix [14]. The calculation for them are often given by

$$\tau^{max} = \frac{1}{1-\rho}\frac{1}{J_{\phi^{gb}}} \text{ and } \tau^{min} = \frac{\tau^{max}}{2 * n} \tag{7}$$

with n being the instance size, and they are updated every time a new global best solution is found. Therefore, after applying (5) to update the pheromone matrix, τ_{ij} will be set to τ^{max} if $\tau_{ij} > \tau^{max}$ and to τ^{min} if $\tau_{ij} < \tau^{min}$.

Finally, with the construction and pheromone update methods defined, the whole MMAS algorithm is summarized in Alg. 2. Regarding the initial values for the pheromone matrix, it is better to set them at an arbitrary high value so that after the first iteration they will be set to τ^{max}, which increases the initial exploration of the algorithm [14].

The method `SelectBest()` is responsible for choosing which of the best ant will be used to update the pheromone trails. This could be a fixed choice or could be according to some schedule of variation between both of them [14].

Algorithm 2: $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System

```
m ← number of ants
global-best ← ∞
pheromone-matrix ← InitializePheromoneMatrix()
repeat
    k ← 0
    iteration-best ← ∞
    while k < m do
        ant ← ConstructSolution()
        ant ← LocalSearch(ant)
        if Cost(ant) < Cost(iteration-best) then
            iteration-best ← ant
        end
        k ← k + 1
    end
    if Cost(iteration-best) < Cost(global-best) then
        global-best ← iteration-best
        UpdateMaxMinValues()
    end
    best-ant ← SelectBest(global-best, iteration-best)
    UpdatePheromoneMatrix(best-ant)
until stopping criteria is not met
return global-best
```

The stopping criteria may be defined as a maximum number of iterations or computational time. It could also be the acceptance of the current global best solution or some convergence detection mechanism [13]. The `LocalSearch()` method is described in the following section.

4 Local Search for QAP

The most basic local search mechanism is the Iterative Improvement, which only allows movements that improve the current state until getting stuck in a local optimum solution. Nevertheless, it is still a suitable approach to be used as an intensification component in a hybrid approach, due to its simplicity and because the good quality of the constructed solutions means that less local search improvement is necessary, specially in later stages of the ACO algorithm [13].

The behavior of the algorithm depends on the movement operator and on the pivot rule, which respectively define the neighborhood and how it is scanned. For the later, the two main approaches are the First Improvement (FI), where the movement happens as soon as a better solution is found, and the Best Improvement (BI), in which the whole neighborhood (or a truncated one) is first scanned and then it moves to the best solution among them [3].

As for the movement operator, it can be easily achieved by exchanging two elements from randomly selected positions i and j (with $i \neq j$). For example, with $i = 1$ and $j = 6$, the permutation $\{2, 4, 8, 1, 3, 5, 6, 7\}$ becomes $\{2, 5, 8, 1, 3, 4, 6, 7\}$. In this operator, the positions pairs (i, j) and (j, i) result in the same permutation, so, for an instance of size n, the amount of possible neighbors is given by $\left(\frac{n^2-n}{2}\right)$.

In case of the QAP, the computational performance of this algorithm using the described operator can be enhanced by, instead of calculating the $O(n^2)$ cost function for each neighbor, we calculate just the cost $\delta(\phi, i, j)$ of swapping the elements from positions i and j in permutation ϕ, with the linear equation [15]

$$\begin{aligned}\delta(\phi,i,j) = {} & d_{ii} * (f_{\phi(j)\phi(j)} - f_{\phi(i)\phi(i)}) + d_{ij} * (f_{\phi(j)\phi(i)} - f_{\phi(i)\phi(j)}) + \\ & d_{ji} * (f_{\phi(i)\phi(j)} - f_{\phi(j)\phi(i)}) + d_{jj} * (f_{\phi(i)\phi(i)} - f_{\phi(j)\phi(j)}) + \\ & \sum_{k=1,k\neq i,j}^{n} (d_{ki} * (f_{\phi(k)\phi(j)} - f_{\phi(k)\phi(i)}) + d_{kj} * (f_{\phi(k)\phi(i)} - f_{\phi(k)\phi(j)}) + \\ & d_{ik} * (f_{\phi(j)\phi(k)} - f_{\phi(i)\phi(k)}) + d_{jk} * (f_{\phi(i)\phi(k)} - f_{\phi(j)\phi(k)}))\end{aligned} \tag{8}$$

This computational cost can be even further reduced by using information from preceding iterations. For the cases where the swapping indexes $\{u, v\}$ are different from the previous indexes $\{i, j\}$ that were used to generate the current permutation ϕ', such as that $(\{u, v\} \cap \{i, j\} = \emptyset)$, the movement cost can be calculated in constant time by [15]

$$\begin{aligned}\delta(\phi',u,v) = \delta(\phi,i,j) * {} & (d_{ru} - d_{rv} + d_{sv} - d_{su}) * (f_{\phi(j)\phi(u)} - f_{\phi(j)\phi(v)} + \\ & f_{\phi(i)\phi(v)} - f_{\phi(i)\phi(u)}) * \\ & (d_{ur} - d_{vr} + d_{vs} - d_{us}) * (f_{\phi(u)\phi(j)} - f_{\phi(v)\phi(j)} + \\ & f_{\phi(v)\phi(i)} - f_{\phi(u)\phi(i)})\end{aligned} \tag{9}$$

This justifies the use of the BI pivot rule, hence it is the one that benefits the most from the preceding neighborhood scans.

5 Experiments

We compared the performances of the standard MMAS-QAP ($\beta = 0$) to the MMAS-QAP using heuristic information ($\beta = 1$). In preliminary tests, higher values for β yielded bad results for most instances. The other parameters were set according to [13], being $\alpha = 1$, $\rho = 0.2$ and the number of ants $m = 5$. The low number of ants is justified by the employment of a complete local search with the best improvement pivot rule. As for the values of τ^{max} and τ^{min}, they depend on the instance size and are given by (7).

We also studied the three different choices for which ant is allowed to update the pheromone trails, they being: the global best; the iteration best; both of them in a specific schedule. For the later, we used the following schedule described in [13]: for every u^{gb} iteration, the global best is selected, then, for the first 11 iterations, u^{gb} is set to 3, decreasing to 2 until iteration 25 and to 1 from there on. The ideia behind this is to initially select the iteration best ant more often, and then, gradually increase the frequency of the global best ant. In this way, the algorithm starts with diversification and moves towards intensification.

In order to observe the dimension of the search space covered by the ants, we used the average λ-branching factor ($\overline{\lambda}$), proposed in [7]. Considering each column c of the pheromone matrix (whose values represent the pheromone trail on assigning the correspondent facility to each location by row), we set τ_c^{max} and τ_c^{min} to the greatest and the smallest values of that column, respectively. Thus, the λ-branching factor of c is given by the number of values that are greater than $\tau_c^{min} + \lambda * (\tau_c^{max} - \tau_c^{min})$, where λ is a control parameter ($0 \leq \lambda \leq 1$).

The average λ-branching factor is then obtained by computing the average of all columns factors and its value varies between $[1, n-1]$, with n being the instance size (for the first iteration, where all pheromone trails are equal, thus resulting in $\overline{\lambda} = 0$, we manually set it to 1). With this, we can infer the degree of exploration made by the algorithm, being that high values indicate high exploration, whereas values decreasing close to 1 indicate search stagnation. During the tests, we set $\lambda = 0.05$ and measured the $\overline{\lambda}$ every iteration.

All the experiments were applied on sixteen QAP instances under different classifications. According to [15], the instances can be classified in four groups: (i) Random and uniform distances and flows (`nug30`, `sko42`, `sko56`, `sko64`); (ii) Random flows on grid (`tai35a`, `tai40a`, `tai50a`, `tai60a`); (iii) Real-life problems (`kra30b`, `kra32`, `ste36a`, `ste36c`); (iv) Randomly generated real-life like (`tai35a`, `tai40b`, `tai50b`, `tai60b`). The instances were retrieved from the online repository QAPLIB and the numbers in the names represent their sizes.

Every configuration was applied 30 times during 10 minutes, where each 15 of them ran simultaneously on an Intel Xeon CPU with 16 threads (8 cores) and 2.4 GHz clock frequency. The results shown are the average of the 30 executions. We applied the Kruskal-Wallis H test and the Nemenyi post-hoc test to check if the results samples being compared are from different distributions. This is shown in the tables by marking the best results in boldface and marking with a gray background all that are equivalent to the best (*p-value* > 0.05).

6 Results

We initially observed the performance of the three choices for updating the pheromone matrix when not using the heuristic information. These results are shown in Table 1, in which the values are the average distances from the known optimum solutions for each instance.

Table 1: Comparison of the average distances from known optimum solutions for different update choices not using the heuristic information ($\beta = 0$)

Instance	Custom Schedule	Global Best	Iteration Best
Random flows on grid			
nug30	**0.146**	0.177	0.197
sko42	**0.207**	0.299	0.226
sko56	1.123	**1.023**	1.224
sko64	1.300	1.278	**1.238**
Random and uniform distances and flows			
tai35a	**1.537**	1.594	2.376
tai40a	1.907	**1.781**	2.694
tai50a	2.720	**2.623**	3.160
tai60a	3.291	**3.194**	3.257
Real-life problems			
kra30b	**0.087**	0.147	0.126
kra32	**0.509**	0.601	0.557
ste36a	**0.340**	0.597	0.379
ste36c	0.162	0.285	**0.046**
Randomly generated real-life like			
tai35b	0.088	0.095	**0.062**
tai40b	0.201	**0.182**	0.366
tai50b	**0.376**	0.401	0.512
tai60b	1.329	**1.202**	1.216
Total Average	0.95769	0.96744	1.10225

It can be seen that the global best and the custom schedule strategies choices presented best results, each outperforming the other more often depending on the instance class. The custom schedule had a clear advantage by absolute majority for the real-life problems whereas the global best had its best performance for the random and uniform instances.

But there was no unanimity in any of the classes, which means that there are instances whose MMAS behavior diverges from the others of the same class. For example, even though the iteration best was the least overall performing choice, it had the best result for the instance `ste63c`, even with statistical difference from the global best choice.

So, for comparison we selected the custom schedule as the best update choice for $\beta = 0$ because it outperformed the others more often and, in cases it did not, it was always statistically equivalent to the best one. Also, it presented the least overall distance, given by the average of all instances results.

As for the experiments using the heuristic information, the results are shown in Table 2, in which we can see that, again, the competition was between the custom schedule and global best choices, since they presented similar performances.

Although both of them were always statistically equivalent to the best one, only the global best remained its absolute majority for the random and uniform class. Also, it was the strategy that had the best total average and with the most

wins. The reason for why the global best update strategy is now outperforming the custom schedule is that the heuristic information allows better initial solution constructions, thus rewarding the use of a more intensifying strategy.

Table 2: Comparison of the average distances from known optimum solutions for different update choices using the heuristic information ($\beta = 1$)

Instance	Custom Schedule	Global Best	Iteration Best
Random flows on grid			
nug30	0.179	0.251	**0.150**
sko42	**0.168**	0.238	0.208
sko56	0.909	**0.761**	1.128
sko64	**1.179**	1.219	1.313
Random and uniform distances and flows			
tai35a	1.644	**1.520**	2.170
tai40a	**1.682**	1.710	2.587
tai50a	2.671	**2.546**	3.166
tai60a	3.232	**3.119**	3.305
Real-life problems			
kra30b	0.174	0.175	**0.112**
kra32	0.648	**0.383**	0.511
ste36a	0.516	0.451	**0.441**
ste36c	**0.043**	0.131	0.085
Randomly generated real-life like			
tai35b	**0.080**	0.136	0.082
tai40b	**0.269**	0.335	0.400
tai50b	0.374	**0.300**	0.466
tai60b	0.794	**0.651**	0.936
Total Average	0.91013	0.87038	1.06625

Finally, in Table 3 we then compared the results obtained by the MMAS without heuristic and using the custom schedule update choice rule (MMAS_{noh_cs}) to the MMAS with heuristic and using the global best update choice rule (MMAS_{h_gb}).

Table 3: Comparison of the average distances from known optimum solutions between MMAS_{noh_cs} and MMAS_{h_gb}

Instance	MMAS_{noh_cs}	MMAS_{h_gb}
Random flows on grid		
nug30	**0.146**	0.251
sko42	**0.207**	0.238
sko56	1.123	**0.761**
sko64	1.300	**1.219**
Random and uniform distances and flows		
tai35a	1.537	**1.520**
tai40a	1.907	**1.710**
tai50a	2.720	**2.546**
tai60a	3.291	**3.119**

Instance	MMAS_{noh_cs}	MMAS_{h_gb}
Real-life problems		
kra30b	**0.087**	0.175
kra32	0.509	**0.383**
ste36a	**0.340**	0.451
ste36c	0.162	**0.131**
Randomly generated real-life like		
tai35b	**0.088**	0.136
tai40b	**0.201**	0.335
tai50b	0.376	**0.300**
tai60b	1.329	**0.651**

	MMAS_{noh_cs}	MMAS_{h_gb}
Total Average	0.95769	0.87038

Apart from the random and uniform class, in which the MMAS_{h_gb} had better results for all instances, both strategies outperformed each other in equivalent times, with a slightly better overall performance for the MMAS_{h_gb}. However, it is noticeable that, for the larger instances, the MMAS_{h_gb} yielded statistically better results than MMAS_{noh_cs}. This, allied to the fact that the execution time

might not have been enough for those larger instances, indicates that the use of heuristic information allows a faster convergence.

To further analyse this statement, in Table 4 other informations from the previous experiments are given: the average iterations made by each strategy for all instances, the average time in seconds in which the best solution was found and the average value of $\overline{\lambda}$ captured at the end of the algorithm execution. It is possible to notice that the strategy that uses heuristic information completed more iterations, meaning that less time was required in the local search phase due to the better constructed solutions.

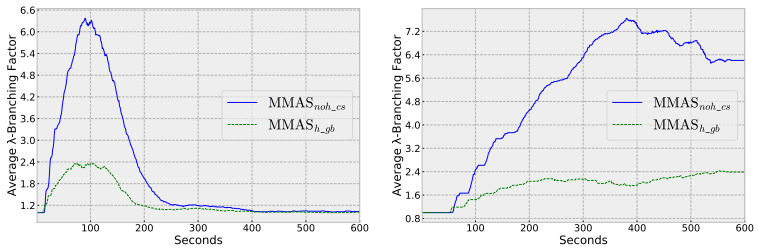
Table 4: Average of iterations, times in which the best results were found and the $\overline{\lambda}$ at the end of algorithm for all instances with both strategies

	MMAS_{noh_cs}			MMAS_{h_gb}		
instance	Iterations	Time (seconds)	Finishing $\overline{\lambda}$	Iterations	Time (seconds)	Finishing $\overline{\lambda}$
nug30	229	160	1.0244	271	153	1
sko42	66	453	1.0627	80	406	1.05
sko56	19	438	6.2065	23	529	2.3869
sko64	12	325	7.0651	14	360	2.1182
tai35a	161	354	1.0448	187	267	1.0095
tai40a	102	450	1.0758	118	380	1.0408
tai50a	46	481	1.3373	55	501	1.334
tai60a	24	387	5.3539	28	475	1.7733
kra30b	239	172	1	278	158	1
kra32	194	222	1.0198	198	184	1
ste36a	119	297	1.0037	150	272	1
ste36c	117	381	1.0315	150	287	1.012
tai35b	120	229	1	137	226	1
tai40b	71	318	1.0092	84	301	1.0092
tai50b	26	571	3.4593	32	556	1.8073
tai60b	13	406	7.0022	15	463	2.5667

Furthermore, the instances that managed to enter in a convergence state, i.e., with $\overline{\lambda}$ close to 1, achieved their best results faster with the MMAS_{h_gb} strategy. On the other hand, for the larger instances, which ended with a relatively high value of $\overline{\lambda}$, we can see that the MMAS_{noh_cs} strategy yielded its best results earlier. Yet, these premature results are worse than those obtained with the strategy using heuristic (see Table 3), besides, their average λ-branch factor at the end were also higher than those obtained by using heuristic. We can then conclude that, for larger instances, the MMAS_{noh_cs} finished while still in the beginning of a more exploratory phase, meaning that it required more execution time, whereas the MMAS_{h_gb} strategy ended closer to search stagnation.

Aside from requiring fewer iterative improvement appliances, the MMAS_{h_gb} strategy was also able to achieve faster convergence due to the benefit of using the global best update choice, which means that less exploration was necessary. This behavior is exemplified in Fig. 1, which illustrates the average λ-branching factor history by execution time in seconds. We can see that for the instance `ste36c` (Fig. 1a), both strategies started converging about the same time the best result was found, as shown in Table 4. It is also clear the difference of the

exploration made by them, which was mainly due to the different update choices. Meanwhile, Fig. 1b shows the exploration behavior for the instance `sko56`, which was one of the larger instances with premature algorithm termination.



(a) ste36c (b) sko56

Fig. 1: Average λ-branching factor history for different instances

Thus, this confirms what has been stated about the faster convergence time for the MMAS$_{h_gb}$. However, it is important to remark that, although the average λ-branching factor can represent the convergence for MMAS, the upper and lower pheromone trail limits avoid that all ants follow the same path. Therefore, even though it seems that the exploration made by MMAS$_{h_gb}$ is less than desired, the use of heuristic information made it possible to yield overall better and faster solutions than those obtained by employing more exploration.

7 Conclusion

The goal of this work was to do an analysis about the influence of heuristic information for the $\mathcal{MAX}-\mathcal{MIN}$ Ant System with local search when applied to the Quadratic Assignment Problem. Through the performed experiments, it was possible to observe that the use of it or not determines how much exploration is required by the algorithm.

This alters which pheromone update strategy is more suitable to be employed. We were able to conclude that, when using the heuristic, it is better to always use the best solution found so far. If no heuristic is present, the best approach is to vary between this global best ant and the iteration best, in order to make the algorithm spend more time diversifying the search.

We could notice that using the heuristic information results in faster convergence time, meaning that the best results are found earlier than if not using it. The explanation for this is that the solutions constructed by the ants require fewer iterations of local search, and also, the use of heuristic allows less exploration to be made. Since both strategies presented similar solution qualities, the required execution time was a decisive factor for the comparison.

However, this CPU running time performance enhancement was possible because the selected heuristic for the QAP was simple and straightforward. For

some other optimization problems, whose heuristic may be rather complex and add a meaningful overhead to the algorithm, the use of MMAS without heuristic is still a robust approach for them due to its problem independence nature.

For future work, it can be studied the effect of the heuristic when using some techniques such as pheromone smoothing and pheromone reinitialization. It is also possible to apply different and more robust local search strategies.

References

1. Blum, C., Roli, A.: Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Computing Surveys (CSUR) 35(3), 268–308 (2003)
2. Brixius, N.W., Anstreicher, K.M.: The steinberg wiring problem. SIAM (2001)
3. Burkard, R.E., Cela, E., Pardalos, P.M., Pitsoulis, L.S.: The quadratic assignment problem. In: Handbook of Combinatorial Optimization, pp. 1713–1809. Springer (1998)
4. Dorigo, M., Caro, G.D.: Ant colony optimization: a new meta-heuristic. In: Proceedings of the 1999 Congress on Evolutionary Computation. vol. 2, p. 1477 (1999)
5. Dorigo, M., Maniezzo, V., Colorni, A.: Ant system: optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 26(1), 29–41 (1996)
6. Fischetti, M., Monaci, M., Salvagnin, D.: Three ideas for the quadratic assignment problem. Operations Research 60(4), 954–964 (2012)
7. Gambardella, L.M., Dorigo, M.: Ant-q: A reinforcement learning approach to the traveling salesman problem. In: Proceedings of the Twelfth International Conference on International Conference on Machine Learning. pp. 252–260. Morgan Kaufmann (1995)
8. Koopmans, T.C., Beckmann, M.: Assignment problems and the location of economic activities. Econometrica: Journal of the Econometric Society pp. 53–76 (1957)
9. Kovářík, O., Skrbek, M.: Ant Colony Optimization with Castes. In: Artificial Neural Networks - ICANN 2008. pp. 435–442. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (Sep 2008)
10. Oliveira, S.M., Hussin, M.S., Stuetzle, T., Roli, A., Dorigo, M.: A Detailed Analysis of the Population-based Ant Colony Optimization Algorithm for the TSP and the QAP. In: Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation. pp. 13–14. GECCO '11, ACM, New York, USA (2011)
11. Sagban, R., Ku-Mahamud, K.R., Abu Bakar, M.S.: Reactive max-min ant system with recursive local search and its application to tsp and qap. Intelligent Automation & Soft Computing 23(1), 127–134 (2017)
12. Sahni, S., Gonzalez, T.: P-complete approximation problems. Journal of the ACM (JACM) 23(3), 555–565 (1976)
13. Stützle, T., Dorigo, M.: Aco algorithms for the quadratic assignment problem. New Ideas in Optimization (C50), 33 (1999)
14. Stützle, T., Hoos, H.H.: Max–min ant system. Future Generation Computer Systems 16(8), 889–914 (2000)
15. Taillard, E.D.: Comparison of iterative searches for the quadratic assignment problem. Location Science 3(2), 87–105 (1995)
16. Tsutsui, S., Fujimoto, N.: A Comparative Study for Efficient Synchronization of Parallel ACO on Multi-core Processors in Solving QAPs. In: 2015 IEEE Symposium Series on Computational Intelligence. pp. 1118–1125 (Dec 2015)