$\mathcal{MAX}$–$\mathcal{MIN}$ Ant System for Quadratic Assignemnt Problems

Thomas Stützle

Forschungsbericht AIDA–97–04

Darmstadt

Fachgebiet Intellektik, Fachbereich Informatik
Technische Hochschule Darmstadt
Alexanderstraße 10
D–64283 Darmstadt
Germany

$\mathcal{MAX}$–$\mathcal{MIN}$ Ant System for Quadratic Assignment Problems

Thomas Stützle
Technical University of Darmstadt
Department of Computer Science – Intellectics Group
Alexanderstr. 10, 64283 Darmstadt
Tel.: +49-6151-166651
Fax: +49-6151-165326
stuetzle@informatik.th-darmstadt.de

Abstract

In this paper we present the application of $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System to the Quadratic Assignment Problem. $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System is an improvement over Ant System, an earlier proposed algorithm of Ant Colony Optimization. The computational results show that very high solutions quality can be obtained using our approach. As $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System is a hybrid algorithm combining solution construction with local search, the kind of local search procedure has significant influence on its final performance. In particular, we show that the best choice of a local search procedure depends strongly on the instance type.

1 Introduction

The Quadratic Assignment Problem (QAP) is an important problem in theory and practice. Many practical problems like backboard wiring [35], Campus and Hospital Layout [12, 17], Typewriter keyboard design [7], Scheduling [23] and many others [16, 26] can be formulated as QAPs. The QAP can best be described as the problem of assigning a set of facilities to a set of locations with given distances between the locations and given flows between the facilities. The goal then is to place the facilities on locations in such a way that the sum of the product between flows and distances is minimal.

More formally, given n facilities and n locations, two $n \times n$ matrices $A = (a_{ij})$ and $B = (b_{rs})$, where a_{ij} is the flow between facilities i and j and b_{rs} is the distance between locations r and s, the QAP can be stated as follows:

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

where $\Phi(n)$ is the set of all permutations (corresponding to the assignments) of the set of integers $\{1, \ldots, n\}$, and ϕ_i gives the location of unit i in the current solution $\phi \in \Phi(n)$. In the following we denote with $f(\phi)$ the objective function value of permutation ϕ.

The QAP is a $\mathcal{NP}$-hard [22] optimization problem. It is considered as one of the hardest optimization problems as general instances of size $n \geq 25$ cannot be solved exactly today.

Therefore, many heuristic algorithms have been proposed that try to find near optimal solutions to the Quadratic Assignment Problem. These approaches include local search algorithms like iterative improvement, Simulated Annealing [8, 10], Tabu Search [1, 34, 39], Genetic Algorithms [19, 31, 41], Evolution Strategies [32], GRASP [27], Ant System [29], and Scatter Search [11].

In this report we propose the application of $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System($\mathcal{MM}$AS) to the Quadratic Assignment Problem. Ant Colony Optimization (ACO) is a new search metaphor for the solution of combinatorial optimization problems inspired by the foraging behavior of real ant colonies. The first ACO algorithm is Ant System [15], that has also been applied to the QAP [29]. $\mathcal{MM}$AS is an improvement over Ant System that has been shown to yield significantly better solution quality than Ant System on the Traveling Salesman Problem [38, 37]. $\mathcal{MM}$AS is a hybrid algorithm that applies local search to improve solutions that are constructed by simple agents, called ants. The computational results show that $\mathcal{MM}$AS is one of the most efficient algorithms for the Quadratic Assignment Problem. For a hybrid algorithm of particular importance is which local search procedure should be chosen. We will investigate this issue in this paper in more detail. In particular we show that the local search procedure that should be used in an efficient hybrid algorithm depends strongly on the QAP instance type.

The Technical Report is structured as follows. In Section 2 we present the application of $\mathcal{MM}$AS to the QAP. In Section 3 we present the local search procedures considered for the application to the QAP and present the computational results in Section 4. We end by discussing recent related work in 5 and with some concluding remarks.

2 Applying $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System to the QAP

2.1 Introduction

Ant System [9, 13, 15] is a population based, cooperative search algorithm inspired by the foraging behavior of real ants. One of the basic ideas of Ant System is to use the counterpart of the *pheromone trail* used by real ants as a medium for cooperation and communication among a colony of (artificial) ants. In Ant System the pheromone trails are imitated by real-valued numbers, in the following referred to as trail, that are associated with solution attributes. In case of the QAP such a solution attribute is, for example, the assignment of an item i to a specific location j. In the following we will use this solution attribute. Then the trail τ_{ij} refers to the desire of setting item i to location j. The artificial ants are simple agents that have some basic capabilities. Typically, ants are used to construct solutions, influenced by the trails, and therefore have a limited memory in which they memorize partial solutions. Additionally, they may use some heuristic information, like the distance between cities when applied to the Traveling Salesman Problem (TSP). After having constructed the solutions the pheromone trails are updated. This is done by first decreasing them by some constant factor (corresponding to the evaporation of the pheromone) and then reinforcing the solution attributes of the constructed solutions taking into account the their solution quality.

When applied to the TSP, with Ant System only for rather small problems good results may be obtained. This is the main reason why several extensions of the basic algorithm have been proposed. Among those are Ant-Q [20], Ant Colony System (ACS) [21, 14], $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System [36, 38] and the rank-based version of Ant System [5]. Additionally, the performance of ACO algorithms can be improved significantly by adding a local search phase [29, 36, 37, 14], in which the solutions are improved by a local search procedure. Thus, the best performing

ACO algorithms are in fact hybrid algorithms consisting of a solution construction mechanism (influenced by the pheromone trails and possibly a heuristic information) and a subsequent local search phase.

ACO algorithms can be seen as an adaptive sampling algorithms — adaptive in the sense that they consider the experience gathered in earlier iterations of the algorithm in form of the trails. The pheromone trail is manipulated in such a way that solution attributes leading to better solutions receive a higher amount of pheromone and are selected with a higher probability in the following iterations of the algorithm. The pheromone trails act as an adaption mechanism that biases the sampling of new solutions – performed by a colony of ants – towards promising regions of the search space. In $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System additionally ants are allowed to improve their solutions by a local search. The trails are then used as a feedback mechanism to identify *attributes* of the local optimal solutions.

When applying an Ant Colony Optimization approach to combinatorial problems like the QAP, two basic steps have to be considered. One is a probabilistic construction step, in which solutions are constructed and a second step, in which the trails are updated. In the following sections we first detail these two steps. Before, we shortly discuss the previous application of Ant System to the QAP.

2.2 Previous ACO Approaches to the QAP

$\mathcal{MAX}$–$\mathcal{MIN}$ Ant System has features in common with Ant System[1], see also [38] for a more detailed discussion. Therefore, our approach also has some aspects in common with the earlier application of Ant System to the QAP [29]. This concerns the definition of the trails τ_{ij} and that also a local search has been applied in [29]. Apart from the differences concerning the algorithmic details of Ant System and $\mathcal{MM}$AS, the two approaches also differ in that $\mathcal{MM}$AS does not make use of heuristic information for the application to the QAP. In Ant System as heuristic values two vectors d and f have been used that represent the distance potentials and the flow potentials. The distance potentials for a location i is calculated as $\sum_{j=1}^{n} d_{ij}$ and the flow potential for an item i is computed as $\sum_{j=1}^{n} f_{ij}$. The reasoning behind potential values is that in good solutions intuitively items with a high flow potential will be placed on locations with a low distance potential. These two vectors are used in Ant System to guide, in addition to the pheromone trails, the solution construction, see [29] for more details.

2.3 Construction of Solutions

The (artificial) ants construct valid solutions to the QAP; they assign every item exactly to one location and no location is used by more than one item. The constructed solutions correspond to a permutation $\phi \in \Phi(n)$. The construction of a solution involves two steps. First, an item has to be chosen and next the item has to be assigned to some free location.[2] We choose the item which is placed next on a free location randomly among the not yet assigned items. For the second step the pheromone trails are used, τ_{ij} referring to the desire of setting item i on location j. The basic selection rule in Ant System would place an item i on some free location j with probability

[1]For an introduction to Ant System see [15].

[2]We call a location free if no item is assigned to it.

$$p_{ij} = \begin{cases} \frac{\tau_{ij}}{\sum_{k \ still \ free} \tau_{ik}} & if \text{ location } j \text{ is still free} \\ 0 & otherwise \end{cases} \tag{2}$$

The intuition behind this rule is to probabilistically prefer locations for item i that have been shown promising in previously found solutions; the higher τ_{ij}, the more likely it is to put item i on location j.

For ACS [21, 14] a modified selection rule, called *pseudo-random-proportional* rule, has been proposed. With a fixed probability p_0 the best possible decision (*exploitation*) is made, with probability $1 - p_0$ a decision according to (2) is made (*exploration*)

$$j = \begin{cases} arg\max_{k \ still \ free}\{\tau_{ik}\} & if\, p \leq p_0 \quad \text{(exploitation)} \\ S & otherwise \quad \text{(exploration)} \end{cases} \tag{3}$$

where p is a random number uniformly distributed in $[0, 1]$ and S is a random variable with probability distribution given by (2). The parameter p_0 controls the exploitation of the accumulated experience reflected in the trail matrix vs. the biased exploration of new solutions. For $\mathcal{MM}$AS we use the *pseudo-random-proportional* action choice rule. The main effect of using this rule, compared to using only the selection rule given by (2), is that good solutions are found faster.

2.4 Update of trails

After all ants have constructed a solution, the trails are modified by the ants. In $\mathcal{MM}$AS, like also in ACS, we allow only one ant after each iteration to update the trails. The trails are modified according to

$$\tau_{ij}^{new} = \rho \cdot \tau_{ij}^{old} + \Delta\tau_{ij}^{best} \tag{4}$$

where ρ, with $0 < \rho < 1$, is the persistence of the trail. The amount $\Delta\tau_{ij}$ is equal to $1/f(\phi_{best})$ if item i is put on location j in ϕ_{best}, otherwise 0. The solution that is taken to modify the trails may be the best solution found in the current iteration, the *iteration-best* solution ϕ_{best}^{iterat}, or the best solution found during the run of the algorithm, the *global-best* solution ϕ_{best}^{global}. Thus, if in the best solutions items are often put on specific locations, these solution attributes receive a high amount of pheromone and therefore items will be put preferably on these location in future iterations of the algorithm. For the application of $\mathcal{MM}$AS to the QAP we will generally choose the global best solution for the trail update, see Section 4.1 for details on the parameter settings. An exception is made after a reinitialization of the trails as described in the following section.

2.5 Maximum and Minimum trail limits

In $\mathcal{MM}$AS only one ant in each iteration is chosen for a trail update. As the chosen ant is typically the global-best or the iteration-best, we may be confronted with the problem of stagnation of the search. Stagnation of the search would occur if the τ_{ij} associated with the locations of the items in ϕ_{best}^{global} are very high and all the others very low. In such a case by 2 a solution identical or very similar to ϕ_{best}^{global} would be constructed. Thus, the exploration of new, possibly better solutions is very limited.

An important tool to provide a balance between exploration of new solutions and exploitation of the search experience in $\mathcal{MM}$AS is given by limiting the possible trail values between some maximum and minimum limits $[\tau_{min}, \tau_{max}]$. Thus, in $\mathcal{MM}$AS the trail strength associated with solution attributes has to obey that $\forall i, j \; \tau_{min} \leq \tau_{ij} \leq \tau_{max}$. When using the *pseudo-random-proportional* action choice rule in $\mathcal{MM}$AS, the higher p in (3) the tighter the trail limits have to be chosen, i.e., the lower the ratio τ_{max}/τ_{min} has to be. If τ_{max}/τ_{min} is too high, the influence of the trails may be such, that also by the choice according to (2) nearly always an item is put on a location with maximal τ_{ij} and again stagnation of the search would occur early.

For the application of $\mathcal{MM}$AS to the QAP we use an additional techniques to increase the diversification of the search. In case we encounter signs of search stagnation we reinitialize the trails to τ_{max}. After the re-initialization of the trails, only the iteration best solutions ϕ_{best}^{iterat} are used to update the trails for 5 iterations.[3] This additional technique was mainly necessary due to the use of the more aggressive *pseudo-random-proportional* action choice. Due to the tight maximum and minimum trail limits, the trails may rather fast reach the lower trail limit and stay at this level. This effect combined with the trail update using ϕ_{best}^{global} leads very fast to the fact that the trails τ_{ij} corresponding to the locations of the items in ϕ_{best}^{global} are near τ_{max} and almost all others near τ_{max}. Then, the construction process of the ants would basically introduce small random modifications to ϕ_{best}^{global}, similar to the kick-moves used in chained local optimization algorithms [30]. But, most of the search progress is made when the relative trail strengths change frequently when new, better solutions are found to update the trails.

2.6 Improving solutions by local search

Constructive algorithms often result in a poor solution quality compared to local search algorithms. On the other side, it has been noted that repeating local searches from randomly generated initial solutions still for most problems results in a considerable gap to the optimal solution. To find solutions of very high quality some hybrid method is needed. in fact, for many combinatorial optimization problems the highest quality solutions are found by hybrids methods combining some form of solution manipulation with subsequent local search. An often used idea is to combine a probabilistic, adaptive construction heuristic with local search heuristics [4, 18], where the construction heuristics provide good starting points for the local search procedures. ACO algorithms, and therefore also $\mathcal{MM}$AS, are such adaptive construction heuristics. They are used to identify solution attributes that lead to high quality solutions according to past experience reflected in the pheromone trails and in this way guide the local search towards promising regions of the search space. Another advantage of using $\mathcal{MM}$AS as a construction heuristic is that by generating good initial solutions, the subsequent local search needs much less iteration to reach a local optimum. Thus, for a given time limit many more local searches can be done than by starting from randomly generated solutions.

An algorithmic frame for such a hybrid algorithm is given in Figure 1. The loop consisting of *solution construction* according to (3), application of a local search procedure, and *trail update* according to (4) is simply repeated for a given number of iterations.[4]

[3]In case ϕ_{best}^{global} would be used, the solutions would resemble already one iteration after the reinitialization to ϕ_{best}^{global} due to the aggressive choice rule (3).

[4]We refer to the complete cycle of tour construction and trial update as one iteration, similar to generation in

```
procedure $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System with local search
(1) Initialize the pheromone trails and the parameters
(2) while ( termination condition not met) do
  for $k := 1$ to $m$ do
    construct a solution for ant k
  end for;
  Improve solutions by local search
  Update the pheromone trail, $\forall\tau_{ij}\ \tau_{\max} \geq \tau_{ij} \geq \tau_{\min}$
(3) return best solution found
end $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System;
```

Figure 1: Algorithmic frame for $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System with local search procedures.

3 Local Search for the QAP

For a hybrid algorithm like the one proposed here, it is important which kind of local search algorithm is used. Important for the choice of a local search algorithm is its speed and the solution quality it produces. We first discuss some basic issues of local search and local search for the QAP and then detail which local search algorithms have been chosen for the hybrid algorithm.

3.1 Preliminaries

Most heuristic solution methods for the QAP fall into the class of *local search algorithms*. The most basic local search algorithm is iterative improvement that starting from some initial solution iteratively replaces the current solution by a better solution in its neighborhood. Any local search method is based on a definition of a neighborhood structure $\mathcal{N}$. The neighborhood of a solution $s \in \mathcal{S}$ is the set of solutions $\mathcal{N}(s) \subset \mathcal{S}$ that can be reached from s in one step. In case of the QAP the neighborhood of a permutation ϕ is usually defined by set of permutations that can be obtained by exchanging two items, i.e. $\mathcal{N}(\phi) = \{\phi' \mid \phi'(i) = \phi(i)\ \forall i \neq r, s \ \wedge \ \phi'(r) = \phi(s),\ \phi'(s) = \phi(r)\}$. The objective function difference $\delta(\phi, r, s)$ of exchanging two units $\phi(s)$ and $\phi(r)$ can be computed in $O(n)$, using the following equation [40]:

$$\begin{aligned}\delta(\phi, r, s) = {} & a_{rr} \cdot (b_{\phi_s\phi_s} - b_{\phi_r\phi_r}) + a_{rs} \cdot (b_{\phi_s\phi_r} - b_{\phi_r\phi_s}) + \\ & a_{sr} \cdot (b_{\phi_r\phi_s} - b_{\phi_s\phi_r}) + a_{ss} \cdot (b_{\phi_r\phi_r} - b_{\phi_s\phi_s}) + \\ \sum_{k=1, k\neq r,s}^{n} & (a_{kr} \cdot (b_{\phi_k\phi_s} - b_{\phi_k\phi_r}) + a_{ks} \cdot (b_{\phi_k\phi_r} - b_{\phi_k\phi_s}) + \\ & a_{rk} \cdot (b_{\phi_s\phi_k} - b_{\phi_r\phi_k}) + a_{sk} \cdot (b_{\phi_r\phi_k} - b_{\phi_s\phi_k}))\end{aligned} \tag{5}$$

Note that (5) also holds for QAP instances with asymmetric matrices A and B. If both matrices are symmetric, the evaluation of $\delta(\phi, r, s)$ can be done faster using the simplified equation

genetic algorithms.

$$\delta(\phi, r, s) = 2 \cdot \sum_{k=1, k \neq r,s}^{n} (a_{kr} - a_{ks}) \cdot (b_{\phi_k \phi_s} - b_{\phi_k \phi_r}) \tag{6}$$

For the QAP the effect of a particular swap can be evaluated faster using information from preceding iterations. Let ϕ' be the solution that is obtained by exchanging units r and s in solution ϕ, then for swapping units u and v, with $(\{u, v\} \cap \{r, s\} = \emptyset)$ the following equation ([40]) can be used

$$\begin{aligned}\delta(\phi', u, v) = \delta(\phi, r, s) + (a_{ru} - a_{rv} + a_{sv} - a_{su}) \cdot (b_{\phi_s \phi_u} - b_{\phi_s \phi_v} + b_{\phi_r \phi_v} - b_{\phi_r \phi_u}) \\ (a_{ur} - a_{vr} + a_{vs} - a_{us}) \cdot (b_{\phi_u \phi_s} - b_{\phi_v \phi_s} + b_{\phi_v \phi_r} - b_{\phi_u \phi_r})\end{aligned} \tag{7}$$

Again, for symmetric QAP instances this formula can be simplified using the fact that $a_{ij} = a_{ji}$, $\forall i, j$ and $b_{\phi_i \phi_j} = b_{\phi_j \phi_i}$, $\forall i, j$.

3.2 Local search for the hybrid algorithm

For the hybrid algorithm combining we considered two possibilities. One is to use a simple iterative improvement local search, in the following referred to as *2-opt*. The other is the use of short runs of Tabu Search, referred to as *TS*. Iterative improvement has the disadvantage that it stops at the first local minimum encountered, whereas Tabu Search allows to escape from local minima and generally finds much better solutions. But, iterative improvement, compared to Tabu Search, runs considerably faster and can be applied much more often. Additionally it benefits from the good solutions generated by $\mathcal{MM}$AS and in later stages of the search only few improvement steps are necessary to reach a local minimum.

Iterative Improvement can be implemented using *first-improvement*, i.e., the first improving move found is accepted, or as *best-improvement*, i.e., the whole neighborhood is examined and a move that gives the best improvement is chosen. With first-improvement usually more moves have to be performed to reach a local minimum, but an examination of the whole neighborhood that may be rather expensive is avoided. Additionally, different local optima may be obtained by scanning the neighborhood in a random order. On the other side, best-improvement local search often gives a slightly better solution quality and in the case of the QAP it benefits from the fact that the effect of moves can be calculated using the information of previous iterations, see (7). We examined the addition of both, a first-improvement local search[5] and a best-improvement local search algorithm. We found that with the best-improvement local search heuristic a slightly better solution quality is obtained for most instances and therefore we present in Section 4 only results using the best-improvement local search.

For many problem instances contained in QAPLIB [6] the best known solutions have been found by Tabu Search algorithms. Thus, one gets the impression that Tabu Search is one of the most effective local search approaches to the QAP. Therefore, we also considered the addition of short Tabu Searches as a local search operator, similar to what was done in an earlier hybrid

[5]It is probably interesting to note that the first-improvement local search was sped up using the *don't look bits* originally used for the TSP [2]. This technique can be easily adapted to the QAP to significantly reduces the run-time of the first-improvement algorithm without loosing much solution quality. Apparently, this technique is also used in [31] although the authors give no details on this.

genetic algorithm for the QAP [19]. As the Tabu Search implementation we use short runs of the Robust Tabu Search (Ro-TS) [39]. We re-implemented Robust Tabu Search in C, corresponding to the Pascal implementation available at http://www.idsia.ch/~eric/. The length of the Ro-TS runs was limited to $4 \cdot n$ iterations as this was also done for the reimplementation of the genetic hybrid algorithm in [40]. This has the advantage that the computational results are comparable across the papers. Nevertheless, other choices for the Tabu Search length may give better performance depending on the QAP instance.

4 Experimental Results

In this section we report on the experimental results obtained with $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System on some QAP instances of QAPLIB. In [40] it has been argued that the type of problem instance has a strong influence on the performance of the different algorithmic approaches proposed for solving the QAP. We first introduce the most important aspects of the different problem types following the description in [40]. In hybrid methods like $\mathcal{MM}$AS with additional local search, the operators chosen may have a significant influence on the final performance. We noted, that for the QAP, in particular, the kind of local search that should be chosen for hybridization has a strong influence on the final performance. We will illustrate this issue before presenting the final computational results comparing $\mathcal{MM}$AS with previously proposed Tabu Search algorithms that are considered to be among the best algorithms for the QAP.

Before presenting the computational results of $\mathcal{MM}$AS applied to the QAP, we detail on the parameter settings used in the final experiments.

4.1 Parameter Settings

Suitable parameter settings were determined in some preliminary experiments. For the experimental evaluation of the approach we use the following parameter settings, suitable parameter settings were determined in some preliminary experiments. We use $m = 5$ ants and $\rho = 0.8$. All 5 ants are allowed to apply a local search. The low number of ants is motivated by the fact that local search for larger QAP instances computationally rather demanding. To still perform a reasonable amount of iterations, the number of ants is chosen rather low. The most important aspect concerning the possible values of the trail strength is that they have to be in some reasonable interval around the expected update through (4). Therefore, the interval for the allowed values of the trail is determined as follows. The (theoretically) maximal possible value for the trail strength in the limit is

$$\tau_{max} = \frac{1}{1-\rho} \cdot \frac{1}{\phi_{opt}} \tag{8}$$

where ϕ_{opt} is the optimal solution value. For $\mathcal{MM}$AS we set τ_{max} according to (8), substituting ϕ_{best}^{global} for ϕ_{opt}. The lower trail limit is chosen as $\tau_{max}/5$. As we use a fixed range for the possible trail limits, we modify the parameter value for p_0 according to the instance size. For the results presented in the following section we chose p_0 in such a way that on average 15 items are chosen according to (2), i.e., $p_0 = (n - 15)/n$. In general we found that the above proposed parameter values yield a rather robust performance. On an instance by instance basis other parameter values may give better performance. In case 2-opt local search is applied, the trail update is always done with ϕ_{best}^{global}, the best found solution during the run of the algorithm, except after the reinitialization of the trails as discussed in Section refss:mmtl. When applying Tabu Search, preliminary experiments suggested that alternatingly choosing ϕ_{best}^{global} and ϕ_{best}^{iterat} for the trail update gives slightly better results.

4.2 Types of QAP instances

According to [40] the instances of QAPLIB can be classified in four classes. (*i*) Instances with the distance and flow matrix entries generated randomly according to an uniform distribution (among those are instances **taixxa** used in Section 4.4). These instances are among the hardest to solve exactly. Nevertheless most iterative search methods find solutions within $1-2\%$ from the best known solutions relatively fast [40]. This observation can also be backed up theoretically [33]. It has been shown that for this type of instances in the limit of large instance sizes, the difference between the upper and lower bounds on the optimal objective function value converges to zero. (*ii*) Another class of instances has the particularity that the distance matrix stems from a $n_1 \times n_2$ grid and the distances are defined as the Manhattan distance between grid points. These instances have multiple global optima (at least 4 in case $n_1 \neq n_2$ and at least 8 in case $n_1 = n_2$) due to the definition of the distance matrices [40] (among those are instances **nugxx** and **skoxx** in Section 4.4). (*iii*) Other instances contained in QAPLIB are "real-life" instances from practical applications of the QAP. Among those are the instances of Steinberg [35] (instances **ste36x**), layout problem for a hospital [17, 25] (instances **kra30x**), instances corresponding to the layout of typewriter keyboards [7] (instances **bur26x**), and a new type of instances proposed in [40]. The real-life problems have in common that the flow matrices have many zero entries and the remaining entries are clearly not uniformly distributed. The matrix entries show structure and it is in this main aspect that real-life instances differ from the randomly generated instances described above. In real-life problems often the situation is given that there is a rather high interaction among a part of the items. Intuitively, in good solutions items with a high interaction with other items should be placed at locations with small inter-location distances. (*iv*) Because the real-life instances in QAPLIB are mainly of rather small size, a type of randomly generated problems has proposed in [40] (instances **taixxb**). These instances are generated in such a way that the matrix entries resemble a distribution found in real-life problems.

In order to differentiate among the classes of QAP instances the flow dominance *fd* may be used. It is defined as the coefficient of variation of the flow matrix entries multiplied by 100.

$$\mu = \frac{1}{n^2} \cdot \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}$$

$$\sigma = \sqrt{\frac{1}{n^2-1} \cdot \sum_{i=1}^{n}\sum_{j=1}^{n}(a_{ij} - \mu)}$$

$$fd(A) = 100 \cdot \frac{\sigma}{\mu}$$

In case few items comprise a large part of the overall flow, this is indicated by a high flow dominance. Thus, randomly generated problems according to an uniform distribution, will have rather low flow dominance, whereas real-life problems, in general, have a rather high flow dominance. A disadvantage of the flow dominance is that it captures only the structure of one matrix, neglecting the distance matrix. Therefore, similar to the flow dominance, also a

Table 1: Dominance values for the QAPLIB instances used in the experimental evaluation of $\mathcal{MM}$AS. The dominance values are given for the first A and the second B matrix as given in QAPLIB. The problem instances are ordered according to the 4 classes described in Section 4.2. Obviously, real-life (like) instances have significantly higher dominance values for at least one of the two matrices.

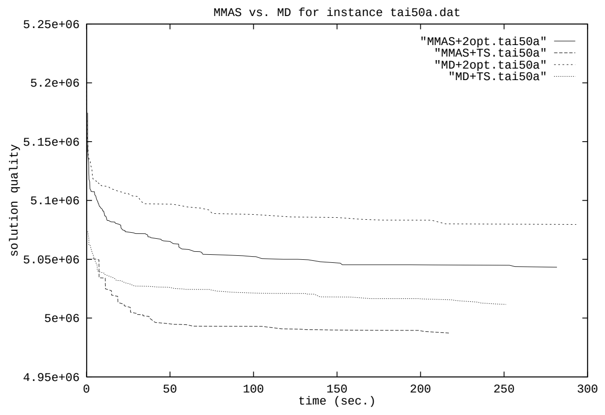
Problem instance	$fd(A)$	$fd(B)$
unstructured, randomly generated (i)		
`tai20a`	67.02	64.90
`tai25a`	61.81	64.29
`tai30a`	58.00	63.21
`tai35a`	61.64	61.57
`tai40a`	63.10	60.23
`tai50a`	60.75	62.24
`tai60a`	61.41	60.86
`tai80a`	59.22	60.38
`tai100a`	59.34	60.31
unstructured, grid-distances (ii)		
`nug20`	54.17	103.78
`nug30`	52.75	112.48
`sko42`	51.96	108.48
`sko49`	51.55	109.38
`sko56`	51.46	110.53
`sko64`	51.18	108.38
`sko72`	51.14	107.13
`sko81`	50.93	106.61
`sko90`	50.91	108.34
`sko100a`	50.75	106.64

Problem instance	$fd(A)$	$fd(B)$
real-life instances (iii)		
`bur26a`	15.09	274.95
`bur26b`	15.91	274.95
`bur26c`	15.09	228.40
`bur26d`	15.91	228.40
`bur26e`	15.09	254.00
`bur26f`	15.91	254.00
`bur26g`	15.09	279.89
`bur26h`	15.91	279.89
`kra30a`	49.22	149.98
`kra30b`	49.99	149.98
`ste36a`	55.65	400.30
`ste36b`	100.79	400.30
real-life like instances (iv)		
`tai20b`	128.25	333.23
`tai25b`	87.02	310.40
`tai30b`	85.20	323.91
`tai35b`	78.66	309.62
`tai40b`	66.75	317.22
`tai50b`	73.44	313.91
`tai60b`	76.83	317.82
`tai80b`	64.05	323.17
`tai100b`	80.42	321.34

distance dominance (dd) can be defined. In the appendix in Table 1 the dominance values for both matrices of some instances are given (ordered according to the four classes of instances described above). From this table it is clear that the real-life instances and the randomly generated instances close to real-life instances have considerably higher dominance values for at least one of the matrices.

4.3 Which local search?

$\mathcal{MM}$AS is designed as a hybrid method. For the application to the QAP we noted that especially the local search algorithm has a high influence on the final performance. To illustrate the performance difference depending on the type of local search and the instance type, we present computational results for instance `tai50a` (randomly generated matrix entries according to an uniform distribution) and for instance `tai50b` (randomly generated matrix entries that resemble characteristics of real life problems). Both instances show largely different flow dominance (fd = 62.24 in case of `tai50a` and fd = 313.91 in case of `tai50b`. We run $\mathcal{MM}$AS with 2-opt ($\mathcal{MM}$AS+2-opt) and $\mathcal{MM}$AS with short Tabu searches of length $4n$ ($\mathcal{MM}$AS+TS). Additionally we present also results for a multiple descent approach (MD) that restarts the local search from randomly generated initial solutions. All algorithms are given the same computation



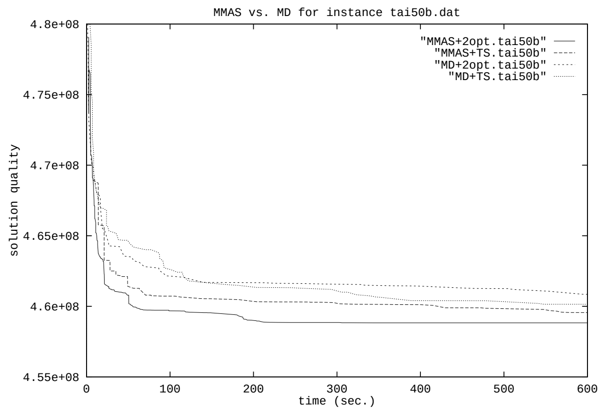


Figure 2: Solution quality development on **tai50a** (left side) and **tai50b** (right side) for $\mathcal{MM}$AS with 2-opt local search (MMAS-2opt), $\mathcal{MM}$AS with short Tabu Searches (MMAS-TS), multiple descent with 2-opt (MD-2opt), and multiple descent with short Tabu Searches (MD-TS). Results based on 10 independent runs for each algorithm. See text for more details

time.[6]

In Figure 2 we give the development of the best solutions found by each of the four algorithms averaged over 10 independent runs. Two important observations can be made from there. One is, that $\mathcal{MM}$AS is able to guide the local search towards better solutions, as for both, 2-opt and Tabu Search, the results with $\mathcal{MM}$AS are better than that of the multiple descent approach. The more important observation is that the kind of local search algorithm which should be chosen for a hybrid algorithm with $\mathcal{MM}$AS strongly depends on the instance type. For instance **tai50a** the addition of Tabu Search gives significantly better solution quality than adding 2-opt. On the contrary, for instance **tai50b** using the simple 2-opt descent local search suffices to outperform the hybrid algorithm using Tabu Search. This is the case although the solution quality obtained from random initial solutions is slightly better for the Tabu Search for longer run-times, for short run times it seems to be better to apply more often the 2-opt local search. In case of the structured instance **tai50b** it pays off to use more often a rather simple local search procedure to identify the preferred locations of the items instead of the more run-time intensive Tabu Search procedure that yields slightly better local optima. The computational results of the next section confirm this observation.

4.4 Computational Results

We now present computational results for the application of $\mathcal{MM}$AS to a wide range of QAP instances taken from QAPLIB. We only applied the algorithms to instances with $n \geq 20$, as most smaller instances are easily solved. For the application of $\mathcal{MM}$AS to the QAP we especially investigate the issue which local search procedure should be added to $\mathcal{MM}$AS. We present computational results using the two different local search procedures as discussed in the previous section. We allow for a total of 250 applications of Tabu Search. For 2-opt a total of 1000 local search applications is allowed, resulting in computation times comparable or lower to 250 short

[6]All the implemented algorithms use the same data structures and the some code as far as possible. Therefore it is fair to compare the algorithms based on CPU-time.

Table 2: Experimental results for $\mathcal{MM}$AS with different types of local search procedures, results for unstructured instances. We give the average solution quality found over 10 independent runs of the algorithms. The computational results are compared to Robust Tabu Search (Ro-TS) and Reactive Tabu Search (RTS), see text for more details.

Problem instance	**Ro-TS**	**RTSs**	$\mathcal{MM}$AS + 250 TS	$\mathcal{MM}$AS + 2-opt-b
random flows on grids				
nug20	0.0	0.7626	0.0	0.0
nug30	0.013	0.8916	0.0196	0.0588
sko42	0.025	0.6830	0.0038	0.1303
sko49	0.076	0.5901	0.0393	0.1557
sko56	0.088	0.6559	0.0720	0.1881
sko64	0.071	0.654	0.0359	0.0668
sko72	0.146	0.558	0.1026	0.2575
sko81	0.136	0.406	0.0776	0.2154
sko90	0.128	0.530	0.1281	0.3545
sko100a	0.108	0.429	0.1322	0.2582
random problems with entries uniformly distributed				
tai20a	0.108	0.1686	0.0665	0.6330
tai25a	0.274	0.365	0.507	1.2966
tai30a	0.426	0.3009	0.259	1.3756
tai35a	0.589	0.457	0.610	2.213
tai40a	0.990	0.569	0.7821	1.8996
tai50a	1.125	0.832	1.1471	1.9602
tai60a	1.203	0.822	0.9806	2.2118
tai80a	0.900	0.497	0.7409	1.5934
tai100a	0.894	0.361	0.6847	1.66

Tabu Searches.[7]

We compare the performance of $\mathcal{MM}$AS to two of the most efficient local search algorithm for the QAP. One is the Robust Tabu Search (Ro-TS) algorithm used also used for the hybrid with $\mathcal{MM}$AS, the other the Reactive Tabu Search (RTS) algorithm [1]. Because the original implementation runs only on symmetric instances, we patched the code to allow also the application to asymmetric instances.[8] Both Tabu Search algorithms, Ro-TS and RTS, are allowed $1000 \cdot n$ iterations, resulting in similar run-times to $\mathcal{MM}$AS+TS.

The computational results presented in Table **??** give the average solution quality obtained in 10 runs of the algorithms. The algorithm that gives the best average performance on each instance is indicated in **bold-face**. In general, the computational results show a significant dependence on the problem type. For the unstructured instances, RTS is the best algorithm(of those compared) for instances **taixxa**, but it performs relatively poor on the unstructured instances with distances taken from a grid (**skoxx** and **nugxx**). Both, $\mathcal{MM}$AS+TS and Ro-TS perform similar on the unstructured instances, but $\mathcal{MM}$AS+TS shows for most of these a slightly better average performance. This indicates the usefulness of $\mathcal{MM}$AS as a diversification mechanism for Tabu Search. The hybrid $\mathcal{MM}$AS+2-opt performs significantly worse than Ro-TS and $\mathcal{MM}$AS+TS on these instances.

[7]The larger the instance size, the smaller is the computation time needed by the 1000 local search applications relative to the 250 Tabu Searches.

[8]The RTS code can be found at http://rtm.science.unitn.it/~battiti/archive/code/rts_qap/README.html. We only changed the evaluation of the moves according to (7).

Table 3: Experimental results for $\mathcal{MM}$AS with different types of local search procedures, results for unstructured instances. We give the average solution quality found over 10 independent runs of the algorithms. The computational results are compared to Robust Tabu Search (Ro-TS) and Reactive Tabu Search (RTS), see text for more details.

Problem instance	**Ro-TS**	**RTSs**	$\mathcal{MM}$AS + 250 TS	$\mathcal{MM}$AS + 2-opt-b
		real life instances		
bur26a-h	0.002	0.092	**0.0**	**0.0**
kra30a	0.268	1.048	**0.135**	0.418
kra30b	0.023	0.128	**0.014**	0.117
ste36a	0.155	2.043	**0.061**	0.184
ste36b	0.081	0.081	0.014	**0.0**
		randomly generated real-life like instances		
tai20b	**0.0**	**0.0**	**0.0**	**0.0**
tai25b	**0.0**	4.427	**0.0**	**0.0**
tai30b	0.107	0.357	**0.0**	**0.0**
tai35b	0.064	0.576	**0.024**	0.094
tai40b	0.531	0.401	0.201	**0.0**
tai50b	0.342	0.426	0.125	**0.029**
tai60b	0.417	0.364	0.043	**0.014**
tai80b	0.591	1.033	0.725	**0.318**
tai100b	0.369	0.555	0.336	**0.142**

On the real-life instances the performance characteristics of the algorithms are different. Here, $\mathcal{MM}$AS+2opt shows a significantly improved performance and is the best algorithm for instances **taixxb** and **bur26x**. For example, for all instances **bur26x** in every run the best known solution value is found on average in 2.5 seconds on a SUN UltraSparc I processor (167Mhz). $\mathcal{MM}$AS+TS took on average 7.4 seconds to solve these instances and Ro-TS could not solve the instances to optimality in all runs and the best solutions in the runs were only found on average after 18.5 seconds. For the instances **taixxb** the Tabu Search algorithms, Ro-TS and RTS, perform, apart from the smallest instances, significantly worse than the $\mathcal{MM}$AS hybrids. Only on the two **kra30x** and the two **ste36x** instances Ro-TS can catch up with the $\mathcal{MM}$AS hybrids. The very good performance of $\mathcal{MM}$AS+2-opt is also backed up by the fact that we were able to find for the largest QAPLIB instance **tai256c** a new best solution with an objective function value of 44759294 in some longer runs of our algorithm [24]. Interestingly, using a simple local search procedure is sufficient to yield very high quality solutions on these structured instances. Hence, for these instances it seems to be better to apply more often a local search procedure in order to identify promising regions of the search space. Once such a promising region is found it is rather easy to find very high quality solutions.[9]

5 Discussion and Related Work

One might think that the flow (distance) dominance could be used to identify which algorithm should be used on a particular instance. In case the flow and distance dominance are low, the best seems to be to use Tabu Search algorithms like Ro-TS or $\mathcal{MM}$AS+TS to get good

[9]For the instances **taixxb** for $n \leq 60$ in almost every run the best-known solutions – conjectured to be optimal – are found.

performance. In case the flow and/or distance dominance is high, the best would be to apply $\mathcal{MM}$AS+2-opt as this algorithm performs best on the structured instances. Although such a simple rule works reasonably well, some exceptions occur. For example for the instance `ste36a` having the highest flow dominance among the real-life instances, $\mathcal{MM}$AS+TS and even Ro-TS give a slightly better average performance than $\mathcal{MM}$AS+2-opt. Thus, obviously there are exceptions to this rule and using the flow dominance alone is not a reliable predictor regarding to which algorithm performs best.

Recently, another Ant Colony Optimization approach to the QAP, called HAS-QAP, has been proposed in [28]. Their approach differs significantly from $\mathcal{MM}$AS presented here. In particular, trails are not anymore used to construct solutions but modify an ants' current solution, similar in spirit to the kick-moves in Chained Local Optimization [30]. For the local search the authors propose a first-improvement local search with maximally two complete neighborhood scans. The computational times of both algorithms, $\mathcal{MM}$AS and HAS-QAP, should be roughly comparable, as [28] report that their algorithm takes somewhat less time as their reimplementation of the genetic hybrids using 250 short Tabu Search runs of length $4 \cdot n$ like $\mathcal{MM}$AS+TS. Comparing the computational results of $\mathcal{MM}$AS to those presented in [28], the behavior of both ACO algorithms is similar. On structured instances, both perform better than the Tabu Search algorithms (Ro-TS and RTS) (and also significantly better than the best Simulated Annealing algorithm [10] as shown in [28]). For the unstructured instances both, HAS-QAP and $\mathcal{MM}$AS+2-opt perform significantly worse than the other algorithms. Nevertheless, the performance of $\mathcal{MM}$AS+2opt seems to be slightly better for the larger structured instances than that of HAS-QAP as well as on the unstructured instances of `taixxa`. As shown before, on these instances $\mathcal{MM}$AS+TS performs at least as good as Ro-TS. Interestingly, the performance of $\mathcal{MM}$AS+TS is comparable to the results of the genetic hybrid algorithm with respect to solution quality. In general both algorithm, $\mathcal{MM}$AS and the genetic hybrids are able to guide the local search effectively on structured problems. The better performance of $\mathcal{MM}$AS+2-opt on many of these instances suggests that the most important part taken by adaptive algorithms in structured domains is the ability to extract the structure and to find promising regions of the search space.

For QAP apart from HAS-QAP [28] two other very promising approaches have been proposed recently. In [31] a genetic local search algorithm for QAPs is presented, extending earlier work of the authors on the TSP. Their approach is in the same spirit as our application of $\mathcal{MM}$AS to QAPs as they adapt their successful genetic algorithm for the TSP to the QAP using only minor modifications. A new feature they present is a fast local search that restricts the size of the neighborhood dynamically based on the distance between individuals. Especially for larger problems they obtain very good results with rather low computation times. Another new approach to the QAP is based on Scatter Search. This algorithm also uses Tabu Searches of various lengths (therefore the results are not directly comparable) as the local search operator [11]. They obtain very high solution quality but apparently at the cost of rather high run times.

6 Conclusions

In this paper we presented computational results for the application of $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System to the Quadratic Assignment Problem. For the QAP the local search method that is combined with $\mathcal{MM}$AS in a hybrid algorithm has a strong impact on the performance. We considered two possibilities, one is the application of short Tabu Search runs, the other is the use of a simple 2-opt iterative improvement local search. For unstructured QAP instances the use of short Tabu Search runs has shown to be particularly effective, whereas with 2-opt the performance was significantly worse on these instances. But, for structured, real-life instances with the more frequent application of a simple 2-opt algorithm a significantly better performance than with short Tabu Searches could be achieved. This result suggests that the shape that an hybrid algorithm should take, in particular which local search algorithm should be chosen for solution improvements, depends strongly on the QAP instance type. Compared to other solution approaches, in general, we can conclude that for unstructured instances $\mathcal{MM}$AS acts as an effective diversification procedure and $\mathcal{MM}$AS is among the best algorithms for structured QAP instances. The good performance of the algorithm is also remarkable as it is a straightforward extension of an earlier application of $\mathcal{MM}$AS to the Traveling Salesman Problem.

For the future work we plan to further improve the performance of $\mathcal{MM}$AS especially on large, structured instances. One possibility could be the use of a faster local search to identify promising regions of the search space. Another important issue is to identify the problem classes which can be solved efficiently using ACO approaches. For many problems it has been shown that their search space provides a structure in the way that locally optimal solutions are clustered in a small part of the search space and the global optimal solution is somehow central to the best local minima [4, 3]. On problems which are known to have such a structure – like the Traveling Salesman Problem – current Ant Colony Optimization approaches work specifically well. We conjecture that also the structured QAP instances that are well solved by $\mathcal{MM}$AS have somehow a similar search space structure. Thus, an analysis of the QAP search space should give additional insights into the performance of $\mathcal{MM}$AS and Ant Colony Optimization algorithms in general.

References

[1] R. Battiti and G. Tecchiolli. The Reactive Tabu Search. *ORSA Journal on Computing*, 6(2):126–140, 1994.

[2] J.L. Bentley. Fast Algorithms for Geometric Traveling Salesman Problems. *ORSA Journal on Computing*, 4(4):387–411, 1992.

[3] K.D. Boese. *Models for Iterative Global Optimization.* PhD thesis, University of California, Computer Science Department, Los Angeles, 1996.

[4] K.D. Boese, A.B. Kahng, and S. Muddu. A New Adaptive Multi-Start Technique for Combinatorial Global Optimization. *Operations Research Letters*, 16:101–113, 1994.

[5] B. Bullnheimer, R.F. Hartl, and C. Strauss. A New Rank Based Version of the Ant System — A Computational Study. Technical report, University of Viena, Institute of Management Science, 1997.

[6] R.E. Burkard, S. Karisch, and F. Rendl. QAPLIB - A Quadratic Assignment Problem Library. *European Journal on Operational Research*, 55:115–119, 1991.

[7] R.E. Burkard and J. Offermann. Entwurf von Schreibmaschinentastaturen mittels quadratischer Zuordnungsprobleme. *Zeitschrift für Operations Research*, 21:B121–B132, 1977.

[8] R.E. Burkard and F. Rendl. A Thermodynamically Motivated Simulation Procedure for Combinatorial Optimization Problems. *European Journal of Operational Research*, 17:169–174, 1984.

[9] A. Colorni, M. Dorigo, and V. Maniezzo. Distributed Optimization by Ant Colonies. In *Proceedings of ECAL91 - European Conference on Artificial Life*, pages 134–142. Elsevier Publishing, 1991.

[10] D.T. Connolly. An Improved Annealing Scheme for the QAP. *European Journal of Operational Research*, 46:93–100, 1990.

[11] V.-D. Cung, T. Mautor, P. Michelon, and A. Tavares. A Scatter Search Based Approach for the Quadratic Assignment Problem. In T. Baeck, Z. Michalewicz, and X. Yao, editors, *Proceedings of ICEC'97*, pages 165–170. IEEE Press, 1997.

[12] J.W. Dickey and J.W. Hopkins. Campus Building Arrangement Using TOPAZ. *Transportation Science*, 6:59–68, 1972.

[13] M. Dorigo. *Optimization, Learning, and Natural Algorithms.* PhD thesis, Politecnico di Milano, 1992.

[14] M. Dorigo and L.M. Gambardella. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. *to appear in IEEE Transactions on Evolutionary Computation*, 1(1), 1997.

[15] M. Dorigo, V. Maniezzo, and A. Colorni. The Ant System: Optimization by a Colony of Cooperating Agents. *IEEE Transactions on Systems, Man, and Cybernetics – Part B*, 26(1):29–41, 1996.

[16] H.A. Eiselt and G. Laporte. A Combinatorial Optimization Problem Arising in Dartboard Design. *Journal of the Operational Research Society*, 42:113–118, 1991.

[17] A.N. Elshafei. Hospital Layout as a Quadratic Assignment Problem. *Operations Research Quarterly*, 28:167–179, 1977.

[18] T.A. Feo and M.G.C. Resende. Greedy Randomized Adaptive Search Procedures. *Journal of Global Optimization*, 6:109–133, 1995.

[19] C. Fleurent and J.A. Ferland. Genetic Hybrids for the Quadratic Assignment Problem. In P.M. Pardalos and H. Wolkowicz, editors, *Quadratic assignment and related problems*, volume 16 of *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, pages 173–187. American Mathematical Society, 1994.

[20] L.M. Gambardella and M. Dorigo. Ant-Q: A Reinforcement Learning Approach to the Traveling Salesman Problem. In *Proceedings of the Twelfth Iternational Conference on Machine Learning*, pages 252–260. Morgan Kaufmann, 1995.

[21] L.M. Gambardella and M. Dorigo. Solving Symmetric and Asymmetric TSPs by Ant Colonies. In *IEEE Conference on Evolutionary Computation (ICEC'96).* IEEE Press, 1996.

[22] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, San Francisco, CA, 1979.

[23] A.M. Geoffrion and G.W. Graves. Scheduling Parallel Production Lines with Changeover Costs: Practical Applications of a Quadratic Assignment/LP Approach. *Operations Research*, 24:595–610, 1976.

[24] Stefan Karisch. personal communication, 1997.

[25] J. Krarup and P.M. Pruzan. Computer-aided Layout Design. *Mathematical Programming Study*, 9:75–94, 1978.

[26] G. Laporte and H. Mercure. Balancing Hydraulic Turbine Runners: A Quadratic Assignment Problem. *European Journal of Operational Research*, 35:378–381, 1988.

[27] Y. Li, P.M. Pardalos, and M.G.C. Resende. A Greedy Randomized Adaptive Search Procedure for the Quadratic Assignment Problem. In P.M. Pardalos and H. Wolkowicz, editors, *Quadratic assignment and related problems*, volume 16 of *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, pages 237–261. American Mathematical Society, 1994.

[28] É.D. Taillard L.M. Gambardella and M. Dorigo. Ant Colonies for the QAP. Technical Report IDSIA-4-97, IDSIA, 1997.

[29] V. Maniezzo, M. Dorigo, and A. Colorni. The Ant System Applied to the Quadratic Assignment Problem. Technical Report IRIDIA/94-28, Universit Libre de Bruxelles, Belgium, 1994.

[30] O. Martin and S.W. Otto. Combining Simulated Annealing with Local Search Heuristics. *Annals of Operations Research*, 63:57–75, 1996.

[31] P. Merz and B. Freisleben. A Genetic Local Search Approach to the Quadratic Assignment Problem. In *to appear in: Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA'97)*, 1997.

[32] V. Nissen. Solving the Quadratic Assignment Problem with Clues from Nature. *IEEE Transactions on Neural Networks*, 5(1):66–72, 1994.

[33] W.T. Rhee. A Note on Asymptotic Properties of the Quadratic Assignment Problem. *Operations Research Letters*, 7(4):197–200, 1988.

[34] J. Skorin-Kapov. Tabu Search Applied to the Quadratic Assignment Problem. *ORSA Journal on Computing*, 2:33–45, 1990.

[35] L. Steinberg. The Backboard Wiring Problem: A Placement Algorithm. *SIAM Review*, 3:37–50, 1961.

[36] T. Stützle and H. Hoos. Improving the Ant-System: A Detailed Report on the $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System. Technical Report AIDA–96–12, FG Intellektik, TH Darmstadt, August 1996.

[37] T. Stützle and H. Hoos. The $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System and Local Search for the Traveling Salesman Problem. In *Proceedings 1997 IEEE International Conference on Evolutionary Computation (ICEC'97)*, pages 309–314, April 13–16, Indianapolis, Indiana, USA, 1997.

[38] T. Stützle and H. Hoos. Improvements on the Ant System: Introducing $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System. In *Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms*, April 1–4, Norwich, UK, 1997. Springer Verlag, Wien.

[39] É.D. Taillard. Robust Taboo Search for the Quadratic Assignment Problem. *Parallel Computing*, 17:443–455, 1991.

[40] E.D. Taillard. Comparison of Iterative Searches for the Quadratic Assignment Problem. *Location Science*, 3:87–105, 1995.

[41] D.M. Tate and A.E. Smith. A Genetic Approach to the Quadratic Assignment Problem. *Computers & Operations Research*, 22(1):73–83, 1995.