ACO Algorithms for the Quadratic Assignment Problem

Thomas Stützle[1] and Marco Dorigo
IRIDIA
Université Libre de Bruxelles
{tstutzle,mdorigo}@ulb.ac.be

[1] Currently on leave from FG Intellektik, TU Darmstadt.

Chapter 3

ACO Algorithms for the Quadratic Assignment Problem

3.1 Introduction

The quadratic assignment problem (QAP) is an important problem in theory and practice. Many practical problems like backboard wiring [33], campus and hospital layout [8, 14], typewriter keyboard design [4], scheduling [18] and many others [13, 22] 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 distance between locations i and j and b_{rs} is the flow between facilities r and s, the QAP can be stated as follows:

$$\min_{\psi \in S(n)} \sum_{i=1}^{n} \sum_{j=1}^{n} b_{ij} a_{\psi_i \psi_j} \tag{3.1}$$

where $S(n)$ is the set of all permutations (corresponding to the assignments) of the set of integers $\{1, \ldots, n\}$, and ψ_i gives the location of facility i in the current solution $\psi \in S(n)$. Here $b_{ij} a_{\psi_i \psi_j}$ describes the cost contribution of simultaneously assigning facility i to location ψ_i and facility j to location ψ_j. In the following we denote with J_ψ the objective function value of permutation ψ.

The term *quadratic* stems from the formulation of the QAP as an integer optimization problem with a quadratic objective function. Let x_{ij} be a binary variable which takes value 1 if facility i is assigned to location j and 0 otherwise. Then the problem can be formulated as:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{l=1}^{n} \sum_{k=1}^{n} a_{ij} b_{kl} x_{ik} x_{jl} \tag{3.2}$$

subject to the constraints $\sum_{i=1}^{n} x_{ij} = 1$, $\sum_{j=1}^{n} x_{ij} = 1$, and $x \in \{0, 1\}$.

The QAP is a $\mathcal{NP}$-hard optimization problem [31]; even finding a solution within a factor of $1 + \epsilon$ of the optimal one remains $\mathcal{NP}$-hard. It is considered as one of the hardest optimization problems as general instances of size $n \geq 20$ cannot be solved to optimality. Therefore, to practically solve the QAP one has to apply heuristic algorithms which find very high quality solutions in short computation time. Several such heuristic algorithms have been proposed which include algorithms like iterative improvement, simulated annealing [5, 6], tabu search [1, 32, 39], genetic algorithms [15, 29, 42], evolution strategies [30], GRASP [24], ant algorithms [17, 25, 26, 27, 34, 38], and scatter search [7].

In particular, for structured, real-life instances, Ant Colony Optimization (ACO) algorithms are currently one of the best performing algorithms. Since the first application of Ant System to the QAP [27], several improved applications have been proposed. These include direct improvements over this first application [26], the application of improved ACO algorithms [25, 34, 38, 41], and an ant based algorithm [17], called HAS-QAP. Except for HAS-QAP, the Ant System application to the QAP and all the proposed improvements on it [25, 26, 27, 34, 38] fit into the framework of the ACO meta-heuristic (see Chapter 2 for a detailed outline of the ACO meta-heuristic). HAS-QAP has the particularity of using pheromone trail information to *modify* complete solutions instead of constructing new solutions from scratch.

In this chapter we give an overview of the existing applications of the ACO meta-heuristic to the QAP and present computational results which confirm that ACO algorithms are among the best performing algorithms for this problem. The chapter is structured as follows. We first outline how the ACO meta-heuristic can be applied to solve QAPs. Then we present the implementations of the ant algorithms for the QAP and discuss the common features of these approaches and their differences. To exemplify the computational results obtained with ACO algorithms, we compare the performance of $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System, one of the improved ACO algorithms, to that obtained using HAS-QAP and other algorithms proposed for the QAP.

3.2 Applying the ACO meta-heuristic to the QAP

To apply the ACO meta-heuristic to a combinatorial optimization problem, it is convenient to represent the problem by a graph $G = (C, L)$, where C are the components and L is the set of connections (see Chapter 2). To represent the QAP as such a graph, the set of components C is given by the facilities and locations of the problem description, and the set of connections L may fully connect the components. The construction of an assignment of facilities to locations can — in the most general case — be interpreted as an ant's walk on the graph representation of the QAP, guided by pheromone trail information (associated to the transitions) and possibly by locally available heuristic information. The ant's walk additionally has to obey certain constraints to ensure that a feasible solution is generated. In the case of QAP, such a feasible solution assigns every facility to exactly one location and every location is assigned exactly one facility. Hence, a feasible solution ψ to the QAP consists of n couplings (i, j) of facilities and locations.

For the practical application of the ACO meta-heuristic to the QAP it is convenient to use a fixed assignment direction, for example, to assign facilities to locations (or locations to facilities). Then, facilities are assigned by the ants in some order, which we refer to as the *assignment order*, to locations. An ant iteratively applies the following two steps until it has constructed a complete solution. In a first step the ant chooses a facility which should be assigned next. In a second step the chosen facility is assigned to some location. Pheromone trails and heuristic information may be associated with both steps. With respect to the first step, the pheromone trails and the heuristic information may be used to learn an appropriate assignment order (obviously, the assignment order may influence the performance of the algorithm). Concerning the second step, the pheromone trail τ_{ij} and the heuristic information η_{ij} associated with the couplings between facility i and location j determines the desirability of putting facility i on location j. After all ants have generated a feasible solution and possibly improved it by applying a local search, ants are allowed to deposit pheromone, taking into consideration the quality of the solutions.

3.3 Available ACO algorithms for the QAP

Since the first application of Ant System [11, 9, 12], the seminal work on ACO, to the QAP [27], several improved ACO applications to this problem have been proposed by various authors. Despite the differences among these algorithms, they share at least two important common aspects.

One aspect concerns solution construction. All the proposed ant algorithms for the QAP associate pheromone trails τ_{ij} only to couplings of the form $\psi_i = j$, hence, τ_{ij} can be interpreted as the desirability of assigning facility i to location j. Concerning the assignment order, all the proposed algorithms suppose that it is either fixed throughout the run of the algorithm (possibly chosen according to some heuristic information) [27, 26, 25] or it is chosen randomly according to a uniform distribution [34, 38, 41].

The second aspect is that all the proposed algorithms improve the ants' solutions using a local search algorithm (see Section 3.4 for details on the local search algorithms applied). Hence, these algorithms are hybrid algorithms combining solution construction by (artificial) ants with local search algorithms. Such a combination may be very interesting. 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 results for most problems in a considerable gap to the optimal solution [20]. Yet, past experience has shown that the combination of a probabilistic, adaptive construction heuristic with local search may yield significantly improved solutions [3, 10, 38]. ACO algorithms are such adaptive construction heuristics, in the sense that a colony of ants modifies the solution representation assigning higher pheromone trail strength to connections (here, corresponding to couplings of facilities to locations), which are contained in better solutions. During the solution construction ants preferably select couplings which have a high pheromone strength and by combining such couplings they generate promising starting solutions for the local search algorithm. An additional advantage of using ant algorithms is that, by generating good initial solutions, the subsequent local search needs far fewer iterations to reach a local optimum. Thus, for a given time limit many more local searches can be run than by starting from randomly generated solutions.

Additionally, all ACO algorithms applied to the QAP apply the procedures of the ACO meta-heuristic in a specific order (see Figure 2.3 in Chapter 2). They first construct solutions, then improve the generated solutions by a local search algorithm, and finally update the pheromone trails. In fact, most of the best performing ACO algorithms for $\mathcal{NP}$-hard combinatorial optimization problems follow this specific scheme and apply local search, which is an optional feature of the ACO meta-heuristic, to improve solutions [10, 16, 38, 35]. An exception are the current applications to network routing and the application to the shortest common supersequence problem, presented in Chapter 4 of this book.

In the following we present the available ant algorithms for the QAP in more details. We first discuss those which are in fact instantiations of the ACO meta-heuristic defined in Chapter 2 and then outline concisely the HAS-QAP algorithm, which does not use solution construction. The details of the local search algorithms applied by each of the approaches are given in Section 3.4.

3.3.1 Ant System for the QAP

Ant System (AS) is the first ACO algorithms which has been applied to the QAP (see Chapter 2 for an outline of its application to the traveling salesman problem); we refer to it the following as AS-QAP [27, 12]. In AS-QAP the assignment order is determined by a pre-ordering of the facilities, as explained below. Then, at each step a facility i is assigned probabilistically to some location j preferring locations with a high pheromone trail τ_{ij} and promising heuristic information η_{ij}.

Heuristic information The heuristic information on the potential goodness of an assignment is determined in AS-QAP as follows. Two vectors d and f are calculated in which the ith component represents respectively the sum of the distances from location i to all other locations and the sum of the flows from facility i to all other facilities. The lower d_i, the distance potential of location i, the more central is considered the location, the higher f_i, the flow potential of facility i, the more important is the facility. Next a coupling matrix $\mathbf{E} = f \cdot d^T$ is calculated, where $e_{ij} = f_i \cdot d_j$. Then, the heuristic desirability of assigning facility i to location j is given by $\eta_{ij} = 1/e_{ij}$. The motivation for using this type of heuristic information is that, intuitively, good solutions will place facilities with high flow potential on locations with low distance potential.

Solution construction A solution is constructed as follows. In AS-QAP facilities are sorted in non increasing order of the flow potentials. At each construction step an ant k assigns the next still unassigned facility i to a free location j with a probability given by:

$$p_{ij}^k(t) = \frac{[\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{l \in \mathcal{N}_i^k} [\tau_{il}(t)]^\alpha \cdot [\eta_{il}]^\beta} \qquad \text{if } j \in \mathcal{N}_i^k \tag{3.3}$$

where $\tau_{ij}(t)$ is the pheromone trail at iteration t, α and β are parameters which determine the relative influence of the pheromone strength and the heuristic information, and $\mathcal{N}_i^k$ is the feasible neighborhood of node i, that is, only those locations that are still free (note that $\sum_{j \in \mathcal{N}_i^k} p_{ij}(t) = 1$). The single steps are repeated until a complete assignment is constructed.

Pheromone update The pheromone trail update applied to all couplings is done according to the following equation:

$$\tau_{ij}(t+1) = \rho \cdot \tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^k \tag{3.4}$$

where ρ, with $0 < \rho < 1$, is the persistence of the pheromone trail, so that $(1-\rho)$ represents the evaporation. The parameter ρ is used to avoid unlimited accumulation of the pheromone trails and allows the algorithm to forget previously done bad choices. $\Delta\tau_{ij}^k$ is the amount of pheromone ant k puts on the coupling (i, j); it is given by

$$\Delta\tau_{ij}^k = \begin{cases} Q/J_\psi^k & \text{if facility } i \text{ is assigned to location } j \text{ in the solution of ant } k \\ 0 & \text{otherwise} \end{cases} \tag{3.5}$$

where ψ^k is the kth ant solution, J_ψ^k its objective function value, and Q is the amount of pheromone deposited by an ant.

A first improvement of AS-QAP is presented in [26] and we refer to it as AS2-QAP. AS2-QAP differs mainly in the way the heuristic information is calculated and in a different way of calculating the probability of assigning facilities to locations. Yet, since a further improvement over AS2-QAP, called ANTS-QAP and presented in the next section, performs generally better, we do not present the details of AS2-QAP. For the sake of completeness, we will indicate the common features between AS2-QAP and the algorithm presented in the next section.

3.3.2 The ANTS algorithm applied to the QAP

In [25] a substantially improved ACO algorithm, called ANTS[1], is introduced presenting its application to the QAP, in the following referred to as ANTS-QAP. The ANTS framework introduces several modifications with respect to AS. The most significant modification is the

[1] According to [25] the term ANTS derives from the fact that the proposed algorithm can be interpreted as an **A**pproximate **N**ondeterministic **T**ree **S**earch since it shares several elements with an approximated Branch & Bound procedure. In fact, in [25] the ANTS algorithm is extended to an exact algorithm; we refer the interested reader to the original reference for details, here we only present the heuristic algorithm.

use of lower bounds on the solution cost of the completion of a partial solution to compute dynamically changing heuristic values η_{ij}. Further modifications are the use of a different action choice rule and a modified trail update rule.

Use of lower bounds In ANTS-QAP lower bounds on the completion cost of a partial solution are used to give heuristic information on the attractiveness of a adding a specific coupling (i, j). This is achieved by tentatively adding the coupling to the current partial solution and by estimating the cost of a complete solution containing that coupling by means of a lower bound. This estimate is then used to influence the probabilistic decisions taken by the ant during the solution construction: the lower the estimate the more attractive is the addition of a specific coupling. Using lower bounds for the heuristic information presents several advantages like the elimination of possible moves if the cost estimation is larger than the so far best found solution. Yet, the lower bound is calculated at each construction step of an ant and, hence, the lower bounds should be efficiently computable.

In ANTS-QAP the Gilmore-Lawler lower bound (GLB) [19, 23] is computed at the start of the algorithm. Along with the lower bound computation one gets the values of the dual variables corresponding to the constraints when formulating the QAP as an integer programming problem (see Equation 3.2). A disadvantage of the GLB is the relatively high computational complexity, which is of $\mathcal{O}(n^3)$. To decrease the computational effort associated with the lower bound computations, in ANTS-QAP a weaker lower bound than GLB is proposed, called LBD. This lower bound has the advantage of having a computational complexity of $\mathcal{O}(n)$.[2] The computational results presented in [25] show that using this weaker lower bound is sufficient to guide the ants' solution construction.

Solution construction In ANTS-QAP a pre-ordering of the locations is given by the values of the dual variables which are obtained when computing the GLB for an instance. Given a location j, ant k decides to assign facility i to this location with the following probability:[3]

$$p_{ij}^k(t) = \frac{\alpha \cdot \tau_{ij}(t) + (1-\alpha) \cdot \eta_{ij}}{\sum_{l \in \mathcal{N}_j^k} \alpha \cdot \tau_{lj}(t) + (1-\alpha) \cdot \eta_{lj}} \qquad \text{if } i \in \mathcal{N}_j^k \tag{3.6}$$

An advantage of Equation 3.6 is that, compared to Equation 3.3, one parameter is eliminated. Additionally, simpler operations which are faster to compute, like multiplications instead of exponentiations, are applied. $\mathcal{N}_j^k$ is the feasible neighborhood; it is defined by the facilities which are not yet assigned to any location. Also note that here $\sum_{i \in \mathcal{N}_j^k} p_{ij}(t) = 1$. In fact, in this formulation the pre-ordering is done on locations instead of facilities. Obviously, it is also possible to use a pre-ordering of the facilities like in AS-QAP.

[2]For details on the lower bound computation we refer to [25]. In AS2-QAP lower bounds are used in the same way as in ANTS, but in AS2-QAP the GLB is used in each construction step.

[3]The same action choice rule is also used in AS2-QAP.

Pheromone update The pheromone update in ANTS-QAP does not use pheromone evaporation, that is, we have $\tau_{ij}(t+1) = \tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^k$. Here, $\Delta\tau_{ij}^k$ is defined as:

$$\Delta\tau_{ij}^k = \begin{cases} \tau_0 \cdot (1 - \frac{J_\psi^k - \mathrm{LB}}{J_{\mathrm{avg}} - \mathrm{LB}}) & \text{if facility } i \text{ is assigned to location } j \text{ in the solution of ant } k \\ 0 & \text{otherwise} \end{cases} \tag{3.7}$$

where J_{avg} is the moving average of the last l solutions provided by the ants and LB is the value of the GLB which is calculated at the start of the algorithm. If an ant's solution is worse than the current moving average, the pheromone trail strength of the couplings constructed by the ants is decreased; if the ant's solution is better, the pheromone trail strength is increased. The additional effect of using Equation 3.7 is a dynamic scaling of the objective function differences which may be advantageous if in later stages of the search the absolute difference between the ant's solution qualities become smaller.

3.3.3 $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System for the QAP

$\mathcal{MAX}$–$\mathcal{MIN}$ Ant System is an improvement over AS, first proposed applying it to the TSP [37, 36] and subsequently to the QAP; referred to in the following as $\mathcal{MM}$AS-QAP. The modifications introduced by $\mathcal{MM}$AS with respect to AS are the following. First, to exploit the best solutions found in an iteration or during the run of the algorithm, after each iteration only one ant is allowed to add pheromone. This ant may be the *iteration-best* or the *global-best* ant. Second, to avoid search stagnation,[4] the allowed range of the pheromone trail strengths is limited to the interval $[\tau_{\min}, \tau_{\max}]$, that is, $\forall \tau_{ij}\ \tau_{\min} \leq \tau_{ij} \leq \tau_{\max}$. Last, the pheromone trails are initialized to the upper trail limit, which causes a higher exploration at the start of the algorithm [35].

Construction of solutions In $\mathcal{MM}$AS-QAP, at each construction step, ant k first randomly chooses a facility i among those not yet assigned, and then places it on a free location j with a probability given by:

$$p_{ij}^k(t) = \frac{\tau_{ij}(t)}{\sum_{l \in \mathcal{N}_i^k} \tau_{il}(t)} \qquad \text{if } j \in \mathcal{N}_i^k \tag{3.8}$$

It is noteworthy that $\mathcal{MM}$AS does not make use of any heuristic information. As shown in the $\mathcal{MM}$AS application to the TSP [38], the heuristic information is not necessary to achieve high quality solutions when solutions are improved by local search. Not using heuristic information additionally eliminates one parameter and, thus, reduces the effort for parameter tuning.[5]

[4]Search stagnation is defined in [12] as the situation where all the ants follow the same path, that is, they all construct the same solution. Yet, with an appropriate choice of the trail limits such a situation will not occur in $\mathcal{MM}$AS. Hence, stagnation in $\mathcal{MM}$AS refers to a situation in which the solutions constructed by the ants are rather similar.

[5]In a first application of $\mathcal{MM}$AS to the QAP the pseudo-random-proportional action choice rule [10]

Update of pheromone trails After all ants have constructed a solution, the pheromone trails are updated according to

$$\tau_{ij}(t+1) = \rho \cdot \tau_{ij}(t) + \Delta\tau_{ij}^{best} \tag{3.9}$$

Here, $\Delta\tau_{ij}^{best}$ is defined as

$$\Delta\tau_{ij}^{best} = \begin{cases} 1/J_{\psi}^{best} & \text{if facility } i \text{ is assigned to location } j \text{ in solution } \psi^{best} \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

where J_{ψ}^{best} is the objective function value of ψ^{best}. ψ^{best} may be either the *iteration-best* solution ψ^{ib}, or the best solution found during the run of the algorithm, the *global-best* solution ψ^{gb}. Hence, if in the best solutions facilities are often put on specific locations, these couplings will have a high amount of pheromone. A judicious choice in the use of either ψ^{ib} or ψ^{gb} for the pheromone trail update may easily help to improve the algorithm's performance. In general, best results are obtained if the frequency of choosing ψ^{gb} increases during the run of the algorithm [35]. Additionally one has to ensure that the pheromone trail strength respects the limits. If after the pheromone update we have $\tau_{ij} > \tau_{\max}$, we set $\tau_{ij} = \tau_{\max}$; analogously, if $\tau_{ij} < \tau_{\min}$, we set $\tau_{ij} = \tau_{\min}$.

Additional diversification features For the $\mathcal{MM}$AS application to the QAP an additional technique to increase the diversification of the search is used. In particular, if the search progress is only very small, the pheromone trails are reinitialized to $\tau_{\max}$.

3.3.4 FANT

In [41] another ACO algorithm is proposed for the QAP, which is called Fast Ant System (FANT). In FANT solutions are constructed in the same way as in $\mathcal{MM}$AS-QAP using the action choice rule given in Equation 3.8, that is, also no heuristic information is used. FANT differs from the other approaches presented so far in two main aspects, the number of ants used and the management of pheromone trails.

Number of ants FANT makes use of only one ant, that is, no population is used. The use of a single ant allows the algorithm to find good solutions fast (also due to the choice of the local search algorithm (see Section 3.4)). Yet, using only one ant appears to be rather a specific parameter setting than an important novelty of the algorithm.

---

has been used for the solution construction. This rule makes with probability q the best possible choice according to the search experience, that is, facility i is put on a location j, for which τ_{ij} is maximal. With probability $1-q$ the location is chosen according to Equation 3.8. For the experimental results presented in Section 3.5 we do not use the pseudo-random-proportional action choice rule, that is, we set simply $q = 0$.

Pheromone update Like ANTS-QAP, FANT does not use pheromone trail evaporation. Initially, all pheromone trails are set to one and after each iteration pheromone is added as follows:

$$\tau_{ij}(t+1) = \tau_{ij}(t) + r \cdot \Delta\tau_{ij} + r^* \cdot \Delta\tau_{ij}^{gb} \tag{3.11}$$

where $\Delta\tau_{ij} = 1/J_\psi$ for every coupling (i,j) which is part of the solution generated in the current iteration, and $\Delta\tau_{ij}^{gb} = 1/J_\psi^{gb}$ for every coupling of the global-best solution ψ^{gb}. r and r^* are two parameters. The value of r may be modified during the algorithm's run (initially it is set to one), while the value of r^* is kept fixed. The two parameters determine the relative reinforcement provided by the current solution and the global-best one. In two occasions the pheromone trails are updated in a different way from the above described rule. (i) If the global best solution has been improved, parameter r is set to one and the pheromone trails are erased and all set to one (intensification of the search around the global best solution). (ii) If the ant constructs a solution which is the same as ψ^{gb}, r is increased by one and again all pheromone trails are erased and set to r (diversification of the search by decreasing the relative weight of ψ^{gb}).

3.3.5 HAS-QAP

HAS-QAP [17], although inspired by previous work on ant algorithms, differs rather strongly from the previously presented ACO algorithms.[6] The major difference is that pheromone trails are not used to construct new solutions from scratch but to modify the current solutions.

Solution modification In HAS-QAP each ant represents a complete solution and pheromone trails are used to direct the modification of the ants' current solutions. The solution modification consists in repeating R of the following swaps of facilities. First, a facility i is randomly chosen in $\{1,\ldots,n\}$. Then, a second facility $j \neq i$ is chosen and their locations ψ_i and ψ_j are exchanged. The second facility j is chosen in the following way. With probability $1-q$, where q is a parameter, it is chosen to maximize the value of $\tau_{j\psi_i^k} + \tau_{i\psi_j^k}$ (exploitation), with probability q the second index j is chosen with probability:

$$p_{ij}^k(t) = \frac{\tau_{i\psi_j^k}(t) + \tau_{j\psi_i^k}(t)}{\sum_{l\neq i}(\tau_{i\psi_l^k}(t) + \tau_{l\psi_i^k}(t))} \tag{3.12}$$

After a solution has been modified, local search is applied to the so generated solution. In this sense, HAS-QAP is similar in spirit to iterated local search algorithms [28, 35] in which a local search algorithm is repeatedly applied from mutations of the current solution. In HAS-QAP additional features of search intensification and diversification are added, but we refer to [17] for a detailed description of the techniques used.

[6] In fact, it doesn't belong to the ACO meta-heuristic.

Table 3.1: Summary of the characteristics discussed in this section. $\surd$ means that the feature is used or present and $\neg$ that it is not used.

Feature	AS-QAP	ANTS-QAP	$\mathcal{MM}$AS-QAP	FANT-QAP	HAS-QAP
Heuristic	$\surd$	$\surd$	$\neg$	$\neg$	$\neg$
Population	$\surd$	$\surd$	$\surd$	$\neg$	$\surd$
Evaporation	$\surd$	$\neg$	$\surd$	$\neg$	$\surd$
Solution construction	$\surd$	$\surd$	$\surd$	$\surd$	$\neg$

Pheromone update In HAS-QAP only the global-best solution is allowed to add pheromone. The pheromone update follows Equation 3.9, where $\Delta\tau_{ij}^{best}$ is defined as

$$\Delta\tau_{ij}^{best}(t) = \begin{cases} Q/J_{\psi}^{gb} & \text{if facility } i \text{ is put on location } j \text{ in } \psi^{gb} \\ 0 & \text{otherwise} \end{cases} \tag{3.13}$$

where Q is a parameter which determines the amount of added pheromone.

3.3.6 Synopsis

As indicated at the beginning of this section, the proposed ant algorithms for the QAP have some common features like the use of local search to improve solutions and that pheromone trails are only associated to couplings of facilities and locations. Yet, they also differ in certain aspects of how the ideas of the ACO meta-heuristic are put to work. In Table 3.1 we summarize some of the general differences concerning the presence of heuristic information[7], the use of a population, whether pheromone evaporation is used[8], or whether solutions are constructed from scratch. Apart from these differences, the presented algorithms also differ in several specific details like discussed before.

3.4 Local Search for the QAP

Local Search starts from some initial assignment and repeatedly tries to improve the current assignment by local changes. If in the neighborhood of the current assignment a better assignment ψ' is found, it replaces the current assignment and the local search is continued from ψ'.

In the QAP case, the neighborhood of a permutation ψ is typically defined by the set of permutations which can be obtained by exchanging two facilities. The objective function

[7] Current experience suggests that to obtain very good performance, heuristic information is not absolutely necessary if solutions are improved by local search [38, 35].

[8] By using evaporation we refer to lowering the phermone trail strength after each algorithm iteration by a constant factor. If an algorithm does not use pheromone evaporation, it typically uses other additional techniques like reinitialization of pheromones or similar to avoid stagnation of the search.

difference $\Delta(\psi, r, s)$ obtained by exchanging facilities ψ_s and ψ_r can be computed in $O(n)$, using the following equation [40]:[9]

$$\begin{aligned}\Delta(\psi, r, s) = {} & b_{rr} \cdot (a_{\psi_s\psi_s} - a_{\psi_r\psi_r}) + b_{rs} \cdot (a_{\psi_s\psi_r} - a_{\psi_r\psi_s}) + \\ & b_{sr} \cdot (a_{\psi_r\psi_s} - a_{\psi_s\psi_r}) + b_{ss} \cdot (a_{\psi_r\psi_r} - a_{\psi_s\psi_s}) + \\ \sum_{k=1, k\neq r,s}^{n} & (b_{kr} \cdot (a_{\psi_k\psi_s} - a_{\psi_k\psi_r}) + b_{ks} \cdot (a_{\psi_k\psi_r} - a_{\psi_k\psi_s}) + \\ & b_{rk} \cdot (a_{\psi_s\psi_k} - a_{\psi_r\psi_k}) + b_{sk} \cdot (a_{\psi_r\psi_k} - a_{\psi_s\psi_k}))\end{aligned} \quad (3.14)$$

The effect of a particular swap can be evaluated faster using information from preceding iterations. Let $\psi^{'}$ be the solution which is obtained by exchanging facilities r and s in solution ψ, then for swapping facilities u and v, with $(\{u, v\} \cap \{r, s\} = \emptyset)$ the move can be evaluated in constant time [40]:

$$\begin{aligned}\Delta(\psi', u, v) = \Delta(\psi, r, s) + {} & (b_{ru} - b_{rv} + b_{sv} - b_{su}) \cdot (a_{\psi_s\psi_u} - a_{\psi_s\psi_v} + a_{\psi_r\psi_v} - a_{\psi_r\psi_u}) \\ & (b_{ur} - b_{vr} + b_{vs} - b_{us}) \cdot (a_{\psi_u\psi_s} - a_{\psi_v\psi_s} + a_{\psi_v\psi_r} - a_{\psi_u\psi_r})\end{aligned} \quad (3.15)$$

The simplest local search algorithm based on the above described neighborhood is iterative improvement, in the following referred to as `2-opt`. Iterative improvement can be implemented using a *first-improvement* or a *best-improvement* pivoting rule. While in the first case an improving move is immediately performed, in the second case the whole neighborhood is examined and a move which gives the best improvement is chosen. Best-improvement `2-opt` for the QAP benefits from the fact that the effect of exchanging two facilities can be calculated fast using the information of previous iterations (see Equation 3.15); the first iteration is of complexity $\mathcal{O}(n^3)$, while the subsequent iterations can be done in $\mathcal{O}(n^2)$. With first-improvement `2-opt` usually more moves have to be performed to reach a local minimum and every complete neighborhood scan is of $\mathcal{O}(n^3)$. Yet, first-improvement algorithms can be executed faster if only a limited number of iterations are performed or additional techniques like the use of *don't look bits* [2] are applied. Additionally, by examining the neighborhood in random order, different local optima may be obtained also when starting from the same initial solution.

A disadvantage of iterative improvement algorithms is that they stop at the first local minimum encountered. This disadvantage may be avoided by using *short* runs of tabu search (TS) [39, 1] or simulated annealing (SA) [5, 6], which allow the local search to leave local minima.[10] Generally, TS or SA will return solutions of better quality than a single application of `2-opt`, yet at the cost of longer computation times. Hence, for a given amount of computing time one faces the following tradeoff: Is it worthwhile to apply fewer times

[9] If both matrices A and B are symmetric with a null diagonal, the formula can be simplified using the fact that $a_{ij} = a_{ji}$ and $b_{\psi_i\psi_j} = b_{\psi_j\psi_i}$.

[10] Obviously, only short runs of TS or SA should be allowed, since in a hybrid ACO algorithm a local search algorithm should be applied several times.

a local search algorithm which, on average, returns better quality solutions or one better applies more often a weaker local search algorithm? The later may be advantageous to faster identify regions of the search space with high quality solutions. Additionally, `2-opt` benefits from the good solutions generated by ACO algorithms in later stages of the search and only few improvement steps are necessary to reach a local minimum.

The presented ant algorithms use the following local search algorithms. In AS-QAP best-improvement `2-opt` and short SA runs have been applied, while in AS2-QAP and ANTS-QAP only best-improvement `2-opt` is applied. The computational results presented with AS-QAP in [27] suggested that using SA better performance can be obtained. In $\mathcal{MM}$AS-QAP short runs of the robust tabu search (Ro-TS) [39] and best-improvement `2-opt` are applied. HAS-QAP and FANT are both using a truncated first-improvement `2-opt`. In particular, at most two complete scans of the neighborhood are done. Hence, the local search not necessarily returns a locally optimal solution, but it is fast. The experimental results presented in the following section for $\mathcal{MM}$AS-QAP will show that the appropriate choice of the local search algorithm depends strongly on the QAP instance type.

3.5 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[11] and compare them with robust tabu search (Ro-TS) [39], a genetic hybrid algorithm (GH) proposed in [15], and HAS-QAP [17]. 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]. We show that the instance type has a strong influence on which local search algorithm should be used in $\mathcal{MM}$AS-QAP. In particular we use $\mathcal{MM}$AS-QAP either with `2-opt` (referred to as $\mathcal{MM}$AS-QAP$_{\texttt{2-opt}}$) or with short robust tabu search runs of length $4n$, where n is the problem dimension (referred to as $\mathcal{MM}$AS-QAP$_{\texttt{TS}}$). Then, we present the computational results and compare them to the above mentioned algorithms.

3.5.1 Parameter settings

Suitable parameter settings for $\mathcal{MM}$AS-QAP were determined in some preliminary experiments. We use $m = 5$ ants (all ants apply local search to the solution they generate) and $\rho = 0.8$. The low number of ants is motivated by the fact that local search for larger QAP instances is computationally rather demanding, but a reasonable number of iterations should be performed to learn the pheromone trails. 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 amount of pheromone deposited by the trail update. Hence, the interval for the allowed values of the pheromone trail strength is determined to reflect this

[11] Accessible via the WWW at address `http://fmatbhp1.tu-graz.ac.at/~karisch/qaplib`.

intuition. We set $\tau_{\max} = \frac{1}{1-\rho} \cdot \frac{1}{J_\psi^{gb}}$, where J_ψ^{gb} is the objective function value of the global best solution (this setting corresponds to the maximally possible trail level for longer runs). The lower trail limit is set $\tau_{\min} = \tau_{\max}/2 \cdot n$.

In case `2-opt` is applied, we apply a specific schedule to alternate the pheromone trail update between ψ^{gb} and ψ^{ib} (see also Section 3.3.3). Let u^{gb} indicate that every u^{gb} iterations ψ^{gb} is used to add pheromone. Then we set u^{gb} to 3 in the 11 first iterations, u^{gb} to 2 up to iteration 25, and from then on we set $u^{gb} = 1$. By shifting the emphasis from the iteration best to the global best solution for trail update, we are shifting from a stronger exploration of the search space to an exploitation of the best solution found so far. After a reinitialization of the pheromone trails, we again apply the given schedule from the start. When applying tabu search for the local search, preliminary experiments suggested that updating pheromone trails using ψ^{gb} in every second iteration gives reasonably good results.

3.5.2 Types of QAP instances

According to [40] the instances of QAPLIB can be classified in four classes (see also Table 3.2).

(i) **Unstructured, randomly generated instances**: Instances with the distance and flow matrix entries generated randomly according to an uniform distribution (among those are instances `taixxa` used in Section 3.5.4). 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].

(ii) **Unstructured instances with grid-distances**: Instances with the distance matrix defined as the Manhattan distance between grid points on a $n_1 \times n_2$ grid and with random flows.

(iii) **Real-life instances**: 'Real-life" instances from practical applications of the QAP. Among those are the instances of Steinberg [33] (instances `ste36x`), layout problem for a hospital [14, 21] (instances `kra30x`), instances corresponding to the layout of typewriter keyboards [4] (instances `bur26x`), and a new type of instances proposed in [40]. The real-life instances have in common that the flow matrices have (in contrast to the previously mentioned randomly generated instances) many zero entries and the remaining entries are clearly not uniformly distributed.

(iv) **Real-life like instances**: Because the real-life instances in QAPLIB are mainly of rather small size, a type of randomly generated problems has been proposed in [40] (instances `taixxb`). These instances are generated in such a way that the matrix entries resemble the 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.

Table 3.2: Dominance values for some QAPLIB instances used in the experiments. The dominance values are given for the first A and the second B matrix ($dd(A)$ and $fd(B)$, respectively) as given in QAPLIB. The problem instances are ordered according to the 4 classes described.

Problem instance	$dd(A)$	$fd(B)$
unstructured, randomly generated (i)		
`tai40a`	63.10	60.23
`tai50a`	60.75	62.24
`tai60a`	61.41	60.86
`tai80a`	59.22	60.38
unstructured, grid-distances (ii)		
`nug30`	52.75	112.48
`sko56`	51.46	110.53
`sko64`	51.18	108.38
`sko72`	51.14	107.13
`sko81`	50.93	106.61

Problem instance	$dd(A)$	$fd(B)$
real-life instances (iii)		
`bur26a`	15.09	274.95
`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)		
`tai40b`	66.75	317.22
`tai50b`	73.44	313.91
`tai60b`	76.83	317.82
`tai80b`	64.05	323.17

$$fd(B) = 100 \cdot \frac{\sigma}{\mu}, \text{ where}$$

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

In case few facilities comprise a large part of the overall flow, this is indicated by a high flow dominance. Randomly generated problems according to a uniform distribution have a rather low flow dominance, whereas real-life problems, in general, have a rather high flow dominance. Analogously to the flow dominance also a *distance dominance* (dd) can be defined. In Table 3.2 we give dominance values for both matrices of some instances of each type. Obviously, real-life instances and the randomly generated real-life like instances have considerably higher dominance values for at least one of the matrices.

3.5.3 Which local search?

ACO algorithms for the QAP are hybrid algorithms which combine two components: solution construction by artificial ants and local search algorithms. Here we focus on the influence of the local search algorithm on the final performance and show that no single best choice for the local search algorithm exists. In particular, the type of the QAP instance has a strong influence on which local search algorithm should be applied. We exemplify this aspect for the application of $\mathcal{MM}$AS to the QAP. In Figure 3.1 we present the average solution quality as a function of computation time, averaged over 10 independent runs, of $\mathcal{MM}$AS-QAP$_{\texttt{2-opt}}$ and $\mathcal{MM}$AS-QAP$_{\texttt{TS}}$. Additionally we present results for a multiple descent approach (MD) that restarts the local search from randomly generated initial solutions. The average solution quality is measured on instance `tai50a` (class (i) as defined above) and on instance `tai50b` (class (iv) as defined above). The two instances show largely

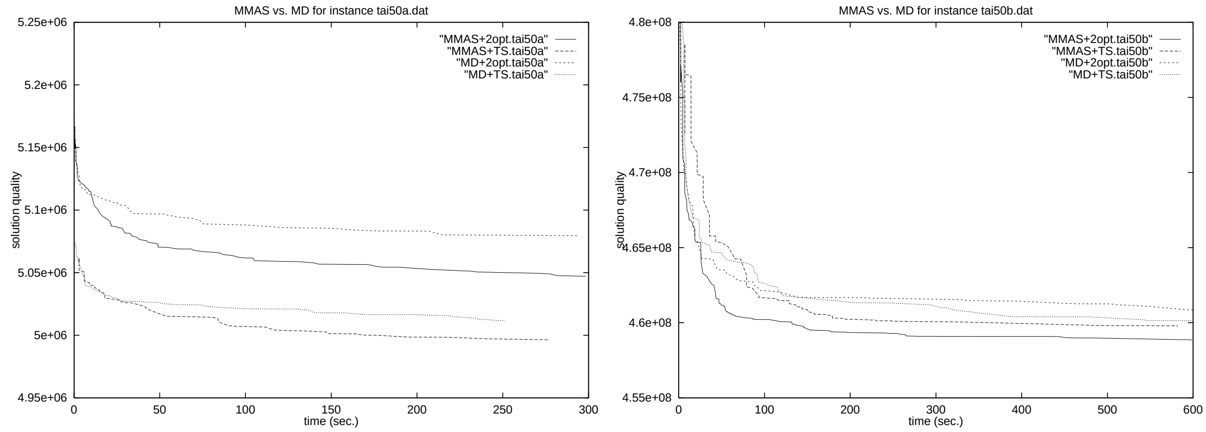

Figure 3.1: Average solution quality development on `tai50a` (left side) and `tai50b` (right side) for $\mathcal{MM}$AS-QAP$_{\texttt{2-opt}}$, $\mathcal{MM}$AS-QAP$_{\texttt{TS}}$, multiple descent with `2-opt` (MD$_{\texttt{2-opt}}$), and multiple descent with short tabu searches (MD$_{\texttt{TS}}$). Best results are in boldface. See text for more details

different flow dominance values ($fd = 62.24$ in case of `tai50a` and $fd = 313.91$ in case of `tai50b`).

Two important observations can be made from the experimental results. First, $\mathcal{MM}$AS is able to guide the local search towards better solutions since the results with $\mathcal{MM}$AS are better than that of the multiple descent approach using the same local search. Second, the local search algorithm which should be used by $\mathcal{MM}$AS-QAP strongly depends on the instance type. For instance, for `tai50a` tabu search gives significantly better solution quality than `2-opt`. On the contrary, for instance `tai50b` $\mathcal{MM}$AS-QAP$_{\texttt{2-opt}}$ outperforms $\mathcal{MM}$AS-QAP$_{\texttt{TS}}$. 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 facilities instead of the more computation intensive tabu search procedure that yields slightly better local optima. The computational results of the next section confirm this observation.

3.5.4 Computational results

We now present computational results for the application of $\mathcal{MM}$AS-QAP to a wide range of QAP instances taken from QAPLIB. We only used instances with $n \geq 20$, since smaller instances are easily solved. When using tabu search we allow a total of 250 applications of tabu search. For $\mathcal{MM}$AS-QAP$_{\texttt{2-opt}}$ we stop the algorithm either after a maximum number of 1000 local search applications or after the same computation time as needed by $\mathcal{MM}$AS-QAP$_{\texttt{TS}}$.

We compare the performance of $\mathcal{MM}$AS-QAP to the robust tabu search (Ro-TS) algorithm, also used for the hybrid $\mathcal{MM}$AS-QAP$_{\texttt{TS}}$, to a genetic hybrid (GH) method which uses short tabu search runs for the local search, and to HAS-QAP. In [40] it was shown that GH performed best for more structured instances of class (*iii*) and (*iv*), whereas tabu search

Table 3.3: Experimental results for heuristic algorithms on unstructured QAP instances (classes (*i*) and (*ii*)). We give the average excess from the best known solutions over 10 independent runs of the algorithms. Best results are in boldface. See text for details.

Problem instance	Ro-TS	GH	HAS-QAP	$\mathcal{MM}$AS-QAP$_{\texttt{TS}}$	$\mathcal{MM}$AS-QAP$_{\texttt{2-opt}}$
random problems with uniformly distributed matrix entries					
tai20a	**0.108**	0.268	0.675	0.191	0.428
tai25a	**0.274**	0.629	1.189	0.488	1.751
tai30a	0.426	0.439	1.311	**0.359**	1.286
tai35a	**0.589**	0.698	1.762	0.773	1.586
tai40a	0.990	**0.884**	1.989	0.933	1.131
tai50a	1.125	**1.049**	2.800	1.236	1.900
tai60a	1.203	**1.159**	3.070	1.372	2.484
tai80a	0.900	**0.796**	2.689	1.134	2.103
random flows on grids					
nug30	0.013	**0.007**	0.098	0.026	0.042
sko42	0.025	**0.003**	0.076	0.015	0.104
sko49	0.076	**0.040**	0.141	0.067	0.150
sko56	0.088	**0.060**	0.101	0.068	0.118
sko64	0.071	0.092	0.129	**0.042**	0.171
sko72	0.146	0.143	0.277	**0.109**	0.243
sko81	0.136	0.136	0.144	**0.071**	0.223
sko90	**0.128**	0.196	0.231	0.192	0.288

algorithms like Ro-TS performed best on the more unstructured instances (classes (*i*) and (*ii*)). Ro-TS is allowed $1000 \cdot n$ iterations, resulting in similar run-times to $\mathcal{MM}$AS-QAP$_{\texttt{TS}}$. In GH 250 short robust tabu search runs of the same length as in $\mathcal{MM}$AS-QAP$_{\texttt{TS}}$ are applied. Hence, the computation times are comparable. HAS-QAP is allowed 1000 applications of the truncated first-improvement local search. Since in [17] it is detailed that the computation times for HAS-QAP are similar to those of GH, the results of HAS-QAP are also roughly comparable with respect to computation times to those of $\mathcal{MM}$AS. In fact, own experience suggest that HAS-QAP takes roughly 75% of the computation time $\mathcal{MM}$AS is given. The computational results for HAS-QAP and GH are taken directly from [17].

The computational results are presented in Table 3.3 for instances of type (*i*) and (*ii*), and in Table 3.4 for those of type (*iii*) and (*iv*). The average solution quality is reported, measured as the percentage deviation from the best known solution and obtained in 10 runs of the algorithms; best results are indicated in **bold-face**. In general, which method performs best depends strongly on the instance type. For the instances of type (*i*) and (*ii*) the hybrids using short tabu search runs and Ro-TS show the best performance (of those compared).[12] Yet, $\mathcal{MM}$AS-QAP$_{\texttt{2-opt}}$ and HAS-QAP perform significantly worse than Ro-TS, GH, and $\mathcal{MM}$AS-QAP$_{\texttt{TS}}$ on these instances.

On the real-life (like) instances, the performance characteristics of the algorithms are very different. Here, $\mathcal{MM}$AS-QAP$_{\texttt{2-opt}}$ and HAS-QAP show much improved performance

[12] In [40] variants of strict tabu search have been shown to perform best on instances of type (*i*).

Table 3.4: Experimental results for heuristic algorithms on structured QAP instances (classes (*iii*) and (*iv*)). We give the average excess from the best known solutions over 10 independent runs of the algorithms. n.a. indicates that an algorithm has not been applied to a specific instance. Best results are in boldface. See text for details.

Problem instance	Ro-TS	GH	HAS-QAP	$\mathcal{MM}$AS-QAP$_{\mathtt{TS}}$	$\mathcal{MM}$AS-QAP$_{\mathtt{2-opt}}$
			real-life instances		
`bur26a-h`	0.002	0.043	**0.0**	0.006	**0.0**
`kra30a`	0.268	**0.134**	0.630	**0.134**	0.314
`kra30b`	**0.023**	0.054	0.071	**0.023**	0.049
`ste36a`	0.155	n.a.	n.a.	**0.036**	0.181
`ste36b`	0.081	n.a.	n.a.	**0.0**	**0.0**
			randomly generated real-life like instances		
`tai20b`	**0.0**	**0.0**	0.091	**0.0**	**0.0**
`tai25b`	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
`tai30b`	0.107	0.0003	**0.0**	**0.0**	**0.0**
`tai35b`	0.064	0.107	0.026	0.051	**0.0**
`tai40b`	0.531	0.211	**0.0**	0.402	**0.0**
`tai50b`	0.342	0.214	0.192	0.172	**0.002**
`tai60b`	0.417	0.291	0.048	**0.005**	**0.005**
`tai80b`	1.031	0.829	0.667	0.591	**0.096**

and, in fact, $\mathcal{MM}$AS-QAP$_{\mathtt{2-opt}}$ is the best algorithm for instances `taixxb` and `bur26x`. For example, for all instances `bur26x` the best known solution value is found in every run. $\mathcal{MM}$AS-QAP$_{\mathtt{2-opt}}$ finds these best known solutions (which are conjectured to be optimal) on average in 3.8 seconds on a SUN UltraSparc I processor (167Mhz). Neither $\mathcal{MM}$AS-QAP$_{\mathtt{TS}}$ nor Ro-TS could find the best known solutions in all runs. In Ro-TS, for example, the best solutions of each run were only found on average after 18.5 seconds. For instances `taixxb` the Ro-TS performs, except for the smallest instances, significantly worse than the $\mathcal{MM}$AS hybrids or HAS-QAP. Only on the `kra30x` and `ste36a` Ro-TS can catch up with the other algorithms.

Interestingly, using a simple local search procedure is sufficient to yield very high quality solutions on the 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. In fact, the ACO algorithms are able to exploit the structure of the real life (like) QAP instances and are able to guide the local search towards high quality solutions. Once such a promising region is identified it is rather easy to find very high quality solutions.[13] Since $\mathcal{MM}$AS-QAP$_{\mathtt{TS}}$ and GH show similar performance characteristics, our results also suggest that replacing the tabu search runs used in GH with a `2-opt` local search should allow the genetic algorithm to find better quality solutions in the same computation time. In fact, this has been done in [41] and their computational results confirm our conjecture.

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

One might conjecture 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 choice appears to be the use of the tabu search algorithms like $\mathcal{MM}$AS-QAP$_{\texttt{TS}}$, GH, or Ro-TS, while for high flow and/or distance dominance, the best would be to apply a hybrid algorithm with a fast local search. Although such a simple rule would work reasonably well, exceptions occur. For example, although instance `ste36a` has the highest flow dominance among the real-life instances, $\mathcal{MM}$AS-QAP$_{\texttt{TS}}$ and even Ro-TS give slightly better average performance than $\mathcal{MM}$AS-QAP$_{\texttt{2-opt}}$.

It should be mentioned here that also in the applications of AS-QAP, AS2-QAP, and ANTS-QAP very good performance compared to other algorithms is obtained, with the ANTS-QAP approach performing best. For example, in [25] the performance of ANTS-QAP has been compared to robust tabu search and GRASP [24]: The computational results show that ANTS-QAP was the best on all the instances tested.[14]

In summary, the computational results show that algorithms based on the ACO meta-heuristic are currently among the best available algorithms for real-life, structured QAP instances.

3.6 Acknowledgements

This work was supported by a Madame Curie Fellowship awarded to Thomas Stützle (CEC-TMR Contract No. ERB4001GT973400). Marco Dorigo acknowledges support from the Belgian FNRS, of which he is a Research Associate.

[14] The results presented in [25] are not directly comparable to the ones presented here due to the different experimental setup.

Bibliography

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

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

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

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

[7] 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.

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

[9] M. Dorigo. *Optimization, Learning, and Natural Algorithms.* PhD thesis, Dip. di Elettronica, Politecnico di Milano, Italy, 1992.

[10] M. Dorigo and L.M. Gambardella. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, 1997.

[11] M. Dorigo, V. Maniezzo, and A. Colorni. Positive Feedback as a Search Strategy. Technical Report 91-016, Dip. di Elettronica, Politecnico di Milano, Italy, 1991.

[12] 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.

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

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

[15] 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.

[16] L.M. Gambardella and M. Dorigo. HAS-SOP: Hybrid Ant System for the Sequential Ordering Problem. Technical Report Technical Report IDSIA-11-97, IDSIA, Lugano, Switzerland, 1997.

[17] L.M. Gambardella, É.D. Taillard, and M. Dorigo. Ant Colonies for the QAP. Technical Report IDSIA-4-97, IDSIA, Lugano, Switzerland, 1997. Accepted for publication in the *Journal of the Operational Research Society (JORS)*.

[18] 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.

[19] P.C. Gilmore. Optimal and Suboptimal Algorithms for the Quadratic Assignment Problem. *Journal of the SIAM*, 10:305–313, 1962.

[20] D.S. Johnson and L.A. McGeoch. The Travelling Salesman Problem: A Case Study in Local Optimization. In E.H.L. Aarts and J.K. Lenstra, editors, *Local Search in Combinatorial Optimization*, pages 215–310. John Wiley & Sons, 1997.

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

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

[23] E.L Lawler. The Quadratic Assignment Problem. *Management Science*, 9:586–599, 1963.

[24] 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.

[25] V. Maniezzo. Exact and Approximate Nondeterministic Tree-Search Procedures for the Quadratic Assignment Problem. Technical Report CSR 98-1, C.L. in Scienze dell'Informazione, Universitá di Bologna, Sede di Cesena, 1998.

[26] V. Maniezzo and A. Colorni. The Ant System Applied to the Quadratic Assignment Problem. Accepted for publication in *IEEE Transactions on Knowledge and Data Engineering*, 1999.

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

[28] O. Martin, S.W. Otto, and E.W. Felten. Large-Step Markov Chains for the Traveling Salesman Problem. *Complex Systems*, 5(3):299–326, 1991.

[29] P. Merz and B. Freisleben. A Genetic Local Search Approach to the Quadratic Assignment Problem. In T. Bäck, editor, *Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA'97)*, pages 465–472. Morgan Kaufmann, 1997.

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

[31] S. Sahni and T. Gonzalez. P-complete Approximation Problems. *Journal of the ACM*, 23:555–565, 1976.

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

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

[34] T. Stützle. $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System for Quadratic Assignment Problems. Technical Report AIDA-97-04, Intellectics Group, Department of Computer Science, Darmstadt University of Technology, Germany, July 1997.

[35] T. Stützle. *Local Search Algorithms for Combinatorial Problems — Analysis, Improvements, and New Applications.* PhD thesis, Intellectics Group, Department of Computer Science, Darmstadt University of Technology, Germany, 1998.

[36] T. Stützle and H.H. Hoos. The $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System and Local Search for the Traveling Salesman Problem. In T. Baeck, Z. Michalewicz, and X. Yao, editors, *Proceedings 1997 IEEE International Conference on Evolutionary Computation, ICEC'97*, pages 309–314, 1997.

[37] T. Stützle and H.H. Hoos. Improvements on the Ant System: Introducing the $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System. In R.F. Albrecht G.D. Smith, N.C. Steele, editor, *Artificial Neural Networks and Genetic Algorithms*, pages 245–249. Springer Verlag, Wien New York, 1998.

[38] T. Stützle and H.H. Hoos. $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System and Local Search for Combinatorial Optimization Problems. In S. Voss, S. Martello, I.H. Osman, and C. Roucairol, editors, *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, pages 313–329. Kluwer, Boston, 1999.

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

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

[41] É.D. Taillard and L.M. Gambardella. Adaptive Memories for the Quadratic Assignment Problem. Technical Report IDSIA-87-97, IDSIA, Lugano, Switzerland, 1997.

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