A Comparison of Memetic Algorithms, Tabu Search, and Ant Colonies for the Quadratic Assignment Problem

Peter Merz and Bernd Freisleben
Department of Electrical Engineering and Computer Science (FB 12)
University of Siegen
Hölderlinstr. 3, D–57068 Siegen, Germany
E-Mail: {pmerz, freisleb}@informatik.uni-siegen.de

Abstract- A memetic algorithm (MA), i.e. an evolutionary algorithm making use of local search, for the quadratic assignment problem is presented. A new recombination operator for realizing the approach is described, and the behavior of the MA is investigated on a set of problem instances containing between 25 and 100 facilities/locations. The results indicate that the proposed MA is able to produce high quality solutions quickly. A comparison of the MA with some of the currently best alternative approaches – reactive tabu search, robust tabu search and the fast ant colony system – demonstrates that the MA outperforms its competitors on all studied problem instances of practical interest.

1 Introduction

The problem of assigning a set of facilities (with given flows between them) to a set of locations (with given distances between them) in such a way that the sum of the product between flows and distances is minimized is known as the facilities location problem [1] or the quadratic assignment problem (QAP) [24]. More formally, given a set \( \Pi(n) \) of all permutations of \( \{1, 2, \ldots, n\} \) and two \( n \times n \) matrices \( A = (a_{ij}) \) and \( B = (b_{ij}) \), where \( a_{ij} \) denotes the distance between location \( i \) and location \( j \), and \( b_{kl} \) represents the flow of materials from facility \( k \) to facility \( l \), the goal is to minimize the quantity

\[
C(\pi) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} b_{\pi(i)\pi(j)}, \quad \pi \in \Pi(n). \tag{1}
\]

Many practical problems from such areas like location science [49], architectural design [12, 21], and hardware/chip design [45, 21] can be formulated as instances of the QAP; other well known combinatorial optimization problems, such as the traveling salesman problem and the graph partitioning problem, are special cases of the QAP [14]. Since the QAP is a \( NP \)-hard problem [20], exact solution approaches are currently only effective for instances of size \( n < 30 \). Therefore, several heuristics have been proposed that attempt to find near-optimum solutions to large QAP instances in reasonable time. Such heuristic approaches include ant colonies [27, 19, 47] (a more general introduction to the ant system can be found in [10, 11]), evolution strategies [41], genetic algorithms [5, 15, 39], simulated annealing [8], neural networks [22], tabu search [3, 43, 44, 49], threshold accepting [40], tree search heuristics [26], and randomized greedy search (GRASP) [25].

In this paper, a memetic algorithm (MA) [37, 38], i.e. an evolutionary algorithm incorporating a local search heuristic [23, 53, 55], for the QAP is presented. The general idea behind memetic algorithms is to combine the advantages of evolutionary operators that determine interesting regions of the search space with local neighborhood search that quickly finds good solutions in a small region of the search space. Since memetic algorithms have been applied with great success to several other combinatorial optimization problems [16, 17, 30, 32, 31], it is quite natural to investigate whether a MA could also be beneficial for producing good solutions for the QAP. In fact, in a previous paper [29] we have already designed a MA for the QAP and evaluated its performance on a set of QAP instances. The contributions of the present paper are: (a) a new, highly effective recombination operator will be presented, (b) a slightly improved local search procedure is used, and (c) the performance of the MA is compared with the currently best heuristics developed for the QAP, including two tabu search approaches [48, 3] and an ant colony algorithm [18, 51]. The results indicate the superiority of the MA: it outperforms the other approaches on all real-life instances of the QAP studied.

The paper is organized as follows. Section 2 presents the MA approach for solving combinatorial optimization problems in general terms. Section 3 describes the evolutionary operators and MA components designed to solve the QAP. In section 4, results for a set of 11 QAP instances produced by the MA are presented, and a comparison with other high-quality heuristics proposed for the QAP is made. Section 5 concludes the paper and outlines areas for future research.

2 Memetic Algorithms

Memetic algorithms (MA) [37, 38] are population-based heuristic search approaches for combinatorial optimization problems based on cultural evolution. They are in some respects similar to genetic algorithms (GAs) which simulate the process of biological evolution. MAs are inspired by Dawkins’ notion of a meme [9] defined as a unit of information that reproduces itself while people exchange ideas. In contrast to genes, memes are typically adapted by the people who transmit them before they are passed on to the next generation.

According to Moscato and Norman [38], ‘memetic evolution’ can be mimicked by combining genetic algorithms with local refinement strategies such as local neighborhood search or simulated annealing. Thus, the genetic local search ap-
procedure QAP_MA;
    begin
        initialize population $P$;
        foreach individual $i \in P$ do $i :=$ Local-Search($i$);
        repeat
            for $i := 1$ to #recombinations do
                select two parents $i_a, i_b \in P$ randomly;
                $i_c :=$ Recombine($i_a, i_b$);
                $i_c :=$ Local-Search($i_c$);
                add individual $i_c$ to $P$;
            endfor;
            $P :=$ select($P$);
            if $P$ converged then
                foreach individual $i \in P \setminus \{best\}$ do $i :=$ Local-Search(Mutate($i$));
            endif
        until terminate=true;
    end;

Figure 1: The Memetic Algorithm for the QAP

proach proposed in [16, 17] is a special case of a memetic algorithm, which has been shown to be very effective for several combinatorial optimization problems, such as the traveling salesman problem (TSP) [30], the graph bi-partitioning problem (GBP) [31, 36], NK-Landscapes [32], and binary quadratic programming [34].

In contrast to hybrid evolutionary algorithms that use local refinement techniques as additional operators, MAs are designed to search in the space of locally optimal solutions instead of searching in the space of all candidate solutions. This is achieved by applying local search after each of the genetic operators. Thus, in any generation, the population of individuals consists solely of local optima. The general template of the memetic algorithm used in this paper is shown in Fig. 1.

The creation of the initial population of (candidate) solutions for a given optimization problem is done in two steps. First, the desired number of feasible solutions is generated, and then a local search procedure is applied to obtain local optima. An obvious way to generate initial solutions is to construct them in a purely random fashion and make sure that the feasibility constraints are satisfied, but alternatively, problem-dependent heuristics can be used.

The recombination and mutation operators are supposed to explore the search space by “jumping” to new regions which are in turn efficiently searched by the local search procedure. While mutation realizes a random undirected jump, the recombination operator allows the region between two or more points in the search space to be explored. This region may contain one or more local optima with better fitness, depending on the structure of the problem.

3 The Memetic Algorithm for the QAP

The problem-specific parts of the MA are the representation of individuals and the initial population, the local search, and the genetic operators. Each of these components together with the problem-independent selection and diversification strategies are described in detail in the following.

3.1 Representation of Individuals and Initial Population

In our approach, the permutation $\pi$ is encoded in the genotype, such that the allele $j$ at position $i$ in the genome indicates that facility $j$ is assigned to location $i$ ($\pi(i) = j$). Individual $A$ in Figure 2 represents a solution where facility 2 is assigned to location 1, facility 4 is assigned to location 2 and so on. The initial population of the MA is created by randomly generated solutions, since (a) effective constructive heuristics for the QAP are rather complex, and (b) random starting solutions in combination with local search lead to relatively good solutions.

![Individual A: 2 4 7 1 8 9 3 5 6, Individual A': 2 4 9 1 8 7 3 5 6](image-url)
3.2 Local Search Procedure

A variant of the 2-opt heuristic, also known as the pairwise interchange heuristic [6], has been selected as the local search method of the proposed MA approach. In the QAP, the 2-opt neighborhood is defined as the set of all solutions that can be reached from the current solution by swapping two elements in the permutation \( \pi \). Figure 2 illustrates such a 2-opt move. The number of swaps and consequently the size of this neighborhood grows quadratically with \( n \), while the change in the total cost \( C(\pi) \) by a swap of the elements \( i \) and \( j \) can be calculated in linear time:

\[
\Delta C(\pi, i, j) = 2 \sum_{k=1, k \neq i, j}^{n} (a_{jk} - a_{ik}) (b_{\pi(i) \pi(k)} - b_{\pi(j) \pi(k)}).
\]

However, in this particular case it is assumed that both matrices \( A \) and \( B \) are symmetric and their diagonals contain zeros. If this does not hold, the formula for calculating \( \Delta C \) is even more complicated [8, 49]. If possible, we use the simplified formula above for calculating \( \Delta C \) in our algorithm. In contrast to the original 2-opt heuristic, our variant is based on performing the first swap found that reduces the total cost \( C(\pi) \). Furthermore, don’t look bits are incorporated to reduce computation time for checking the neighborhood [4]. This reduces the running time without a loss in quality of the solutions. Our experiments have shown that similar to the 2-opt for the TSP, it is not worth spending time to search for the best 2-opt move in our local search procedure, as it is done, for example, in tabu search. Compared to the previously proposed approach [29], the implementation for evaluating \( \Delta C \) has been improved considerably. A detailed description of our local search can be found in [33].

3.3 The Recombination Operator

A distance measure between solutions may help in defining effective genetic operators for a given problem. There are several possibilities for measuring distances between permutations. The distance measure used in our approach is as follows. Let \( \pi_1 \) and \( \pi_2 \) be valid permutations and hence valid solutions to a given QAP instance. The distance between the solutions \( \pi_1 \) and \( \pi_2 \) is defined as:

\[
D(\pi_1, \pi_2) = \{i \in \{1, \ldots, n\} \mid \pi_1(i) \neq \pi_2(i)\}.
\]

The new recombination operator proposed for the QAP preserves the information contained in both parents in the sense that all alleles of the offspring are taken either from the first or from the second parent. In other words, the operator does not perform any implicit mutation, since a facility that is assigned to location \( i \) in the child is also assigned to location \( i \) in one or both parents. The recombination operator works as follows:

- Starting with a randomly chosen location that has no facility assigned yet, a facility is randomly chosen from the two parents. After that, additional assignments are made to ensure that no implicit mutation occurs. Then, the next non-assigned location to the right (in case we are at the end of the genome, we proceed at its beginning) is processed in the same way until all locations have been considered.

Consider the example shown in Figure 3.

![Figure 3: The Recombination Operator for the QAP](image)

First, all facilities that are assigned to the same location in the parents are inherited by the offspring. These are the facilities 4, 9, and 6. Then, beginning with a randomly chosen location (position in the genome, in this case location 3), a facility is randomly selected from one of the parents and is assigned to the same location in the child. In the example, this is facility 7. Now, other facilities have to be assigned to guarantee that no implicit mutation occurs. By assigning facility 7 of parent A to location 3 we prevent the possibility to assign facility 5 of parent B to that location. Hence, we have to assign facility 5 to the same location as in parent A. Again, assigning facility 5 to location 8 requires that facility 2 has to be assigned, too. After that, facility 2 is located at site 1 in the genome. Since the facility at location 1 in parent B is 7, and 7 is already included in the child, we can proceed in choosing a facility for the next free location to the right in the offspring. In our example, facility 8 of parent B is inserted into the offspring, and to avoid implicit mutations, we have to insert facility 1 at location 7 and facility 3 at location 5. Then, the algorithm terminates since all facilities have been assigned.

To limit the region where the local search takes place, the genes with the same alleles contained in both parents are fixed. Hence in the above example, swaps can only be performed between locations 1, 3, 4, 5, 7, and 8. This restricts the local search to the subspace defined by the two parents. The neighborhood size for the local search is reduced.
from \(|V_{2opt}| = \frac{1}{2} n \cdot (n - 1)| \) to \(|V| = \frac{1}{2} d \cdot (d - 1)| \) (with \(d = D(\pi_1, \pi_2))\), which results in an increased performance of the local search phase, since the average distances between individuals of the population decreases during the evolution.

### 3.4 The Mutation Operator

The mutation operator used in the MA approach works by exchanging a sequence of facilities in the solution until the offspring has a predefined distance to its parent. The second facility chosen in each step will be exchanged again in the succeeding step so that the resulting distance between parent and offspring is one plus the number of exchanges. To illustrate the mutation operator, consider the example shown in Figure 4.

![Figure 4: The Mutation Operator for the QAP](image)

In the first step, facilities 9 and 4 are exchanged, then facility 4 and 1, and in the last step facility 1 is exchanged with 3. Thus, the resulting distance between parent and offspring is 4.

### 3.5 Selection and Diversification

Mating selection prior to recombination is performed on a purely random basis without bias to fitter individuals, while selection for survival is done by choosing the best individuals from the pool of parents and children by taking care that each phenotype exists only once in the new population. Thus replacement in our algorithm is similar to the selection scheme in the \((\mu + \lambda)\)-ES [42].

Due to the computation times consumed by the local searches, the population size of a memetic algorithm is typically small compared to genetic algorithms: a size of 10 up to 40 is common in the MA. Such a small population size may lead to a premature convergence of the algorithm. To overcome this drawback, the restart technique has been proposed by Eshelman [13]. In our MA we have adapted this technique to achieve a much more robust search algorithm: Upon convergence (the average distance of the population has dropped below a threshold \(d=10\) or there was no change in the population for more than 30 generations), the whole population is mutated except the best individual, using the mutation operator described above with a high mutation rate (per gene) to avoid falling back into the same local optimum. After mutation, each individual is improved by the local search algorithm to again obtain local optima. Afterwards, the algorithm proceeds with performing recombination as usual. Thus the MA continues with a population of arbitrarily distant local optima. During the run, the points in the search space move closer together until they are concentrated on a small fraction of the search space: the search is said to have converged. The restarts perturb the points so that they are again far away from each other. So, the probability that the algorithm converges to a region in the search space with suboptimal solutions is minimized. The points in the population can be viewed as being repeatedly tied together during the run of a MA.

This technique together with the selection used has been shown to be very effective, but we do not claim that this combination is the best possible for the QAP. Additional studies are needed to verify this.

### 4 Experimental Results

The MA was implemented in C++ on a Pentium II PC (300 MHz) running Solaris 2.6. To evaluate its performance, we selected several QAP instances from the QAPLIB [7], ranging from \(n = 25\) locations up to \(n = 100\).

The QAPLIB contains different types of QAP instances which may be distinguished by their flow dominance and their distance dominance [54, 29]. The flow dominance \(fd\) for the flow matrix \(B\) is defined as the coefficient of variation multiplied by 100:

\[
fd(B) = 100 \cdot \frac{\sigma}{\mu},
\]

\[
\mu = \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} b_{ij},
\]

\[
\sigma = \sqrt{\frac{1}{n^2 - 1} \sum_{i=1}^{n} \sum_{j=1}^{n} (b_{ij} - \mu)^2}.
\]

In case a few entries comprise a large part of the overall flow, the flow dominance is high, and if almost all entries are equally sized, the flow dominance is low. The distance dominance \(dd\) can be defined in a similar manner for the distance matrix \(A\). QAP instances with randomly generated flows (distances) using a uniform distribution typically have a low flow (distance) dominance, whereas real-life instances and (non-uniformly) randomly generated instances close to real-life instances have considerably higher dominance values for at least one of the matrices. Our experiments are based on a set of 11 instances, including problems with high and low flow and/or distance dominance values.

All runs were performed with a population size \(P = 40\). The number of recombinations per generation was set to 0.5 -
Table 1: Comparison of four algorithms for the QAP

<table>
<thead>
<tr>
<th>instance</th>
<th>MA</th>
<th>Ro-TS</th>
<th>Re-TS</th>
<th>FANT</th>
</tr>
</thead>
<tbody>
<tr>
<td>chr25a</td>
<td>29/30</td>
<td>0.077 %</td>
<td>13</td>
<td>5.933 %</td>
</tr>
<tr>
<td>nug30</td>
<td>27/30</td>
<td>0.007 %</td>
<td>20</td>
<td>0.009 %</td>
</tr>
<tr>
<td>kra30a</td>
<td>30/30</td>
<td>0.000 %</td>
<td>20</td>
<td>0.089 %</td>
</tr>
<tr>
<td>ste36a</td>
<td>17/30</td>
<td>0.078 %</td>
<td>30</td>
<td>0.085 %</td>
</tr>
<tr>
<td>tae60a</td>
<td>0/30</td>
<td>1.311 %</td>
<td>92</td>
<td>1.282 %</td>
</tr>
<tr>
<td>tae80a</td>
<td>0/30</td>
<td>1.027 %</td>
<td>180</td>
<td>1.005 %</td>
</tr>
<tr>
<td>tae100a</td>
<td>0/30</td>
<td>1.135 %</td>
<td>300</td>
<td>0.899 %</td>
</tr>
<tr>
<td>sko100a</td>
<td>0/30</td>
<td>0.169 %</td>
<td>161</td>
<td>0.172 %</td>
</tr>
<tr>
<td>tae60b</td>
<td>30/30</td>
<td>0.000 %</td>
<td>90</td>
<td>1.057 %</td>
</tr>
<tr>
<td>tae80b</td>
<td>20/30</td>
<td>0.014 %</td>
<td>180</td>
<td>2.732 %</td>
</tr>
<tr>
<td>tae100b</td>
<td>14/30</td>
<td>0.026 %</td>
<td>301</td>
<td>2.304 %</td>
</tr>
</tbody>
</table>

P. Restarts are performed by mutating almost all positions in the genome (n exchanges of facilities during mutation); no additional mutations are performed between the restarts.

Table 1 displays the results obtained by our MA, together with the results produced by the robust tabu search algorithm (Ro-TS) [48], the reactive tabu search algorithm (Re-TS) [3], and the fast ant colony algorithm combined with local search (FANT) [51, 50]. The three competitors belong to the best currently available heuristic approaches to the QAP. However, other ant system approaches exist for the QAP, such as MMAS-QAP [47, 46] and HAS-QAP [18], which perform better than FANT in most cases [51], but their source code is not publicly available. To enable a fair comparison with the MA, we have used the code developed by the corresponding authors and executed the programs on our hardware and operating system platform.

In the table, instance denotes the name of the QAP instance from the QAPLIB (the number indicates its size n), N_{tr,s} denotes the number of runs in which a best known solution was found, avg. shows the average deviation from the best solution obtained within 30 runs, and t/s displays the average time in seconds required within the 30 runs. In each case, the runtime of the MA was adapted to the runtimes of one of its competitors to enable a fair comparison (the only stop criterion used in our MA is the predefined running time). Ro-TS and Re-TS were run for 40000 iterations, whereas 1000 iterations per run were used for the FANT approach.

The results indicate that the MA is superior, in terms of solution quality within a given time limit, to our previous algorithm [29], and furthermore to all alternative approaches for all but three instances, namely tai60a, tai80a, and tai100a, which are solved more effectively by Ro-TS and Re-TS. These three instances were randomly generated by Taillard using a uniform distribution, and their flow dominance is about 60% while the other instances have a fd value higher than 100%. In [49], Taillard mentions that for this type of randomly generated instances, finding good solutions (about 1% and 2%) is easy, but it is extremely difficult to find the optimum. Taillard [49] further notes that these randomly generated instances are not really significant for practical applications of the QAP and therefore has defined a set of non–uniformly generated random instances (tai* b) with the same characteristics as real–life problems.

In fact, it is hardly surprising that the MA is not the top performer for the three uniformly randomly generated QAP instances, since the MA is designed to exploit some kind of (assumed) structure in the QAP search space; if there is no structure at all, there is not much the MA can do (except for randomly jumping around). However, considering that the performance of the MA for tai60a, tai80a, and tai100a is still acceptable (better than FANT and nearly as good as Ro-TS), and the MA outperforms its competitors on the remaining instances, the MA appears to be the method of choice for the QAP instances studied. To illustrate its behavior in more detail, Table 2 shows the shortest and average times (of 30 runs) required by the MA to reach the best known solution. In the figure, gen denotes the number of generations, average (dev.) the average value of the objective function (equation (1)), and the average deviation from the known optimum, min. t in s the minimum and avg. t in s the average runtime in seconds. The results illustrate that the algorithm is able to find the best known solutions in all 30 out of 30 runs in short average running times for all structured problems.

Although the three alternative approaches used in the comparison are – to the best of our knowledge – among the most powerful methods available today for solving QAP instances, several other techniques have been published. For example, MMAS-QAP or HAS-QAP may be in some cases superior to FANT, but the results presented in [51, 47] indicate that the approaches are inferior to the MA presented in this paper at least for the tai* b problems. Compared with other approaches, it seems that the proposed MA algorithm is also superior. In [52], a genetic algorithm is proposed and tested for problems of up to size 30. For the largest problem in-
vestigated there (nug30), the optimum could not be found. In [27], results are reported for solving nug30 with the Ant System (AS). The best solution found by a combination of AS and simulated annealing had a fitness value of 6128. In [39], a genetic algorithm called ASPARAGOS has been used to solve the instances ste36a and nug30. The optima for both problems could be found in 279 seconds and 363 seconds, respectively, on a transputer with 64 processors. However, on our state-of-the-art workstation, we found the optima in less than 10 seconds. Comparison studies have been made in [2, 28] for relatively small instances with the conclusion that genetic algorithms perform relatively poor, but that hybridization may enhance the search significantly. The combinatorial evolution strategy (CES) proposed in [41] produces remarkable results for problems of up to size 64 without any kind of domain knowledge, but the results are worse in both quality and computation time than the results presented here.

5 Conclusions

In this paper, a memetic algorithm for solving several instances of the quadratic assignment problem was presented. The ingredients of the memetic algorithm, evolutionary operators and local neighborhood search, were described. In particular, a new highly effective recombination operator was proposed. The performance of the memetic algorithm was investigated on a set of QAP instances and compared to the performance of three very good heuristic approaches to the QAP: reactive tabu search [3], robust tabu search [48], and the fast ant colony system [51]. The MA was able to outperform these alternative heuristics on all QAP instances of practical interest. Furthermore, the approach proves to be very robust, since the best known solutions could be found in all runs with short average running times.

There are several issues for future research. First, the algorithm should be applied to larger instances of the QAP to investigate its scalability. Second, the algorithms performance with other parameter settings for population size, operator rates and running times should be investigated. Third, a detailed analysis of the QAP search space similar to investigations we have already conducted for other combinatorial optimization problems [36, 35] will certainly be beneficial to understand and possibly predict the behavior of the memetic algorithm.

Bibliography


