A Genetic Local Search Approach to 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

Augmenting genetic algorithms with local search heuristics is a promising approach to the solution of combinatorial optimization problems. In this paper, a genetic local search approach to the quadratic assignment problem (QAP) is presented. New genetic operators for realizing the approach are described, and its performance is tested on various QAP instances containing between 30 and 256 facilities/locations. The results indicate that the proposed algorithm is able to arrive at high quality solutions in a relatively short time limit: for the largest publicly known problem instance, a new best solution could be found.

1 INTRODUCTION

In the quadratic assignment problem (QAP), n facilities have to be assigned to n locations at minimum cost. Given a set $\Pi(n)$ of all permutations of $\{1, 2, \ldots, n\}$ and two $n \times n$ matrices $A = (a_{ij})$ and $B = (b_{ij})$, the task is to minimize the quantity

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

Matrix A can be interpreted as a distance matrix, i.e. a_{ij} denotes the distance between location i and location j, and B is referred to as the flow matrix, i.e. b_{kl} represents the flow of materials from facility k to facility l.

The QAP belongs to the class of $\mathcal{NP}$-hard problems [11]. Compared to other combinatorial optimization problems such as the traveling salesman problem and the graph partitioning problem (which are special cases of the QAP), there are still no effective methods available which are able to find a guaranteed optimal solution for large problems: instances of size $n > 20$ are still considered intractable.

Therefore, several heuristics have been proposed for finding near–optimum solutions to large QAP instances, including *ant colonies* [15], *evolution strategies* [20], *genetic algorithms* [3, 8, 18], *simulated annealing* [7], *tabu search* [2, 21, 22, 23], *threshold accepting* [19], and *randomized greedy search* (*GRASP*) [14].

Genetic local search [13, 25, 27] is a hybrid heuristic that combines the advantages of population–based search and local optimization. Local search iteratively moves from one solution to a better one in its neighborhood until a local minimum is reached. While it quickly finds good solutions in small regions of the search space, the genetic operators are suitable for exploring the whole search space in order to identify interesting regions. In case of the traveling salesman problem, the combination of genetic algorithms and local search heuristics has led to a robust search algorithm that is able to find near–optimum solutions for large instances in short time [9, 10, 17].

In this paper, a straightforward application of the *genetic local search* (GLS) approach to the QAP is proposed, and results for a set of 11 problem instances containing between 30 and 256 facilities/locations are presented. The results indicate that GLS is a promising approach to facility layout problems.

The paper is organized as follows. Section 2 describes the general GLS approach and discusses its applicability to combinatorial optimization problems. In Section 3, the genetic operators used to solve the QAP are presented. Section 4 discusses the properties of different types of QAP instances. Results for a set of 11 instances are presented in section 5. Section 6 concludes the paper and outlines areas for future research.

2 GENETIC LOCAL SEARCH

The idea of combining genetic algorithms (GA) and local search (LS) heuristics for solving combinatorial optimization problems has been investigated by many researchers.

```
procedure GLS;
    begin
        initialize population P;
        foreach individual i ∈ P do i := Local-Search(i);
        repeat
            for i := 1 to #crossovers do
                select two parents i_a, i_b ∈ P randomly;
                i_c := Crossover(i_a, i_b);
                i_c := Local-Search(i_c);
                add individual i_c to P;
            endfor;
            for i := 1 to #mutations do
                select an individual i ∈ P randomly;
                i_m := Mutate(i);
                i_m := Local-Search(i_m);
                add individual i_m to P;
            endfor;
            P := select(P);
        until converged;
    end;
```

Figure 1: The GLS Algorithm

For example, in the traveling salesman problem genetic algorithms have been shown to be quite ineffective without incorporating some kind of domain knowledge [10, 12]. For instances with more than one hundred cities, the quality of the solutions is much worse than that of simulated annealing or suitable local search procedures. If local search is applied several times with different starting solutions (multi–start local search), the percentage excess over the optimum still lies in the range of betwen 2% and 4%. In order to find solutions closer to the optimum ($< 0.1\%$), techniques like GLS are required.

In Figure 1, a generic template for a genetic local search algorithm is given. In the first step, an initial population has to be created. A constructive heuristic can be used for that purpose. Afterwards, local search is performed for each individual to obtain a population of local minima. Thus, it is best to choose a construction heuristic that produces solutions which are well suited for the subsequent local search. In the main loop, crossover and mutation operators are applied to randomly chosen individuals for a predefined number of times. To maintain local optimality, the local search procedure is applied to the newly created individuals resulting from the crossover or mutation operator; the new individuals are added to the population afterwards. Before a new generation is processed, some individuals are selected for survival in order to keep the population size constant. The main aim of an adequate selection strategy is to keep the search goal-oriented and also to maintain the diversity of the population.

To successfully realize such a hybrid approach, it is crucial that the interaction of the GA and LS leads to a search algorithm that is better than both of its components. An important question is how to design genetic operators that yield better (locally optimal) individuals than those produced by a multi–start local search. The crossover operator enables us to define new starting points for a local search based on the information contained in the current population. Assuming that good solutions lie relatively close to each other in the search space, one can "travel" from one solution to another in order to get (close) to the optimum. This approach is implicitly taken by many heuristics, but they differ in how they climb "uphill" to get from a (locally optimal) solution to a better one. Therefore, the aim of the crossover is to determine the regions of the search space where better solutions are most likely to be found by the local search procedure. The assumption of several solutions lying close to each other in the search space implies some kind of a distance criterion. The notion of a distance between solutions helps the crossover (in conjunction with the selection strategy) to decide where to guide the search.

On the other hand, the mutation operator should attempt to focus the search on randomly chosen regions, such that the algorithm is able to identify solutions that are hard to find

by the distance-controlled crossover with its regional limitations. The different properties of the crossover and the mutation operator combined with the selective pressure of the GA lead to a robust search algorithm.

3 OPERATORS FOR THE QAP

In order to define the operators of the genetic algorithm for the QAP, an adequate representation of the individuals must be chosen first. We decided to code the permutation π 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 GA 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.

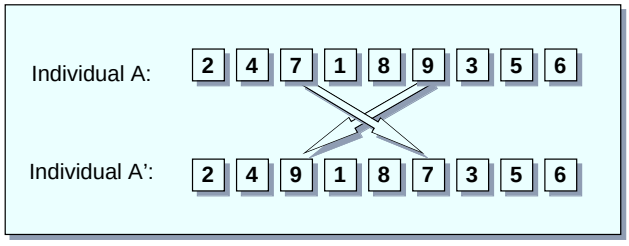

Figure 2: Local Search for the QAP

A variant of the *2-opt* heuristic, also known as the pairwise interchange heuristic [4], has been selected as the local search method of the proposed GLS 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 π. 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 ΔC is even more complicated [7, 23]. In contrast to the original *2-opt* heuristic, our variant is based on performing the first swap found that reduces the total cost $C(\pi)$. This reduces 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, as it is done, for example, in tabu search.

As already mentioned, a distance measure between solutions may help in defining an effective crossover operator for a given problem. There are several possibilities for measuring distances between permutations. The distance measure used in our approach is as follows. Let π_1 and π_2 be valid permutations and hence valid solutions to a given QAP instance. The distance between the solutions π_1 and π_2 is defined as:

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

In [9, 10], we introduced a new crossover operator, called *distance preserving crossover* (DPX), for the traveling salesman problem. Since this operator only relies on the notion of a distance between solutions, it can be used for the QAP, too. In general, the DPX is aimed at producing an offspring which has the same distance to each of its parents, and this distance is equal to the distance between the parents themselves. The alleles that are identical for the same genes in both parents will be copied to the offspring. The alleles for all other genes change. Thus, although the crossover is highly disruptive, the local search procedure subsequently applied is forced to explore a region of the search space that is defined by the genes with different alleles in the parents, which is the region where better solutions are most likely to be found.

The DPX operator for the QAP works as follows. Suppose that two parents A and B as shown in Figure 3 are given.

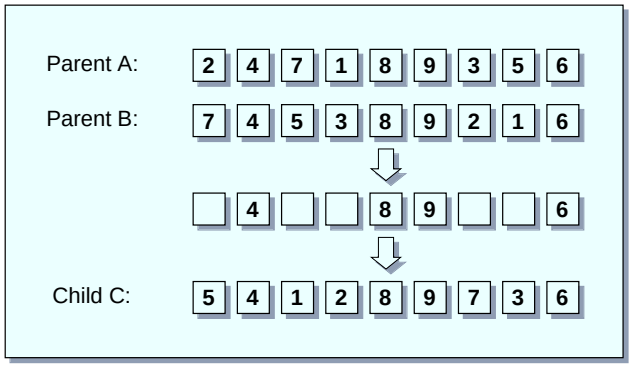

Figure 3: Crossover DPX

First, all assignments that are contained in both parents are copied to the offspring C. The remaining positions of the genome are then randomly filled with the yet unassigned facilities, taking care that no assignment that can be found in just one parent is inserted into the child. After that, we end up with a child C, for which the condition $d = D(C, A) = D(C, B) = D(A, B)$ holds. In the example in Figure 3 the distance d is 5.

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, 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 $|\mathcal{N}_{2opt}| = \frac{1}{2}n \cdot (n-1)$ to $|\mathcal{N}| = \frac{1}{2}d \cdot (d-1)$ (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.

The mutation operator used in the GLS approach simply inverts a part of the genome, as illustrated in Figure 4.

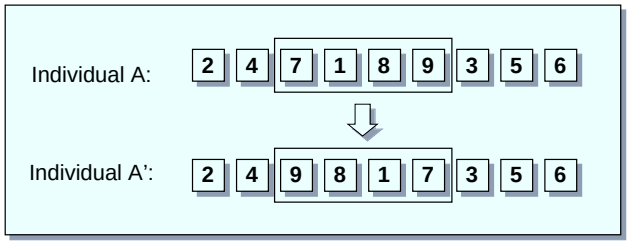

Figure 4: The Mutation Operator

Starting at position k, the order of the entries of a substring of length l is reversed in the genome, leading to a new permutation π' with distance l to the original permutation, if l is even, and distance $l-1$ otherwise.

4 TYPES OF QAP INSTANCES

As mentioned above, QAP instances can be divided into symmetric and asymmetric instances. Whether matrix A or B is symmetric or asymmetric has an effect on how fast the fitness update can be calculated when a *2-opt* move is performed. However, it is not possible to conclude from this property whether an instance is particularly difficult to solve or not. In order to define different types of QAP instances, the measure of *flow dominance* has been proposed in the literature [26]. The flow dominance *fd* for matrix B is defined as the coefficient of variation:

$$
\begin{aligned}
fd(B) &= 100 \cdot \frac{\sigma}{\mu}, \\
\mu &= \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} b_{ij}, \\
\sigma &= \sqrt{\frac{1}{n^2-1} \sum_{i=1}^{n} \sum_{j=1}^{n} (b_{ij} - \mu)^2}.
\end{aligned}
$$

A high flow dominance indicates that few entries in the flow matrix have a high influence on the total cost. This may have a large impact on the efficiency of the search for both heuristics and exact methods. If almost all entries are equally sized, the flow dominance is low. Local changes in a solution can result in an extremely different fitness, and each assignment of a facility to a location has a large impact on the other assignments, making the instance hard to solve. However, it is impossible to decide solely upon the flow dominance whether an instance is hard to solve for a given heuristic or not. For example, it should be noted that the influence of the distance matrix A is not covered by the flow dominance measure.

Several types of QAP instances can be found in the literature. There are instances of practical interest as well as randomly generated ones. Instances with randomly generated distances and flows using a uniform distribution typically have a low flow dominance. These instances are structured in such a way that good solutions can be obtained easily, but approaching the optimum is extremely difficult. Taillard [23] argues that different local optima are only slightly correlated and that the objective function values of the solutions lie in relatively tight bounds, as a consequence of the findings of Burkard and Fincke [5] who showed that with a probability tending to one, the relative difference between the best and worst solution approaches 0 for problems of a size tending to infinity.

However, instances that are randomly generated by non-uniform distributions do not share these characteristics. They are generally easier to solve and are comparable to problems of practical interest.

5 RESULTS

The GLS approach has been implemented in *C++* on a DEC Alphastation running Digital Unix V4. To evaluate its performance, we selected several QAP instances from the QAPLIB [6], ranging from $n = 30$ locations up to $n = 256$. The set consists of 11 instances, including problems with symmetric and asymmetric flow matrices as well as high and low flow dominance.

All runs were performed with a population size $P = 40$ for problems up to size 60 and $P = 20$ for the larger ones. The number of crossovers per generation was set to $0.5 \cdot P$ and the number of mutations to $0.2 \cdot P$. Mutation was performed by inverting a substring of length $l = 10$ in the genome.

Figure 5 displays the results obtained by our algorithm within a predefined time limit.

In the table, *instance* denotes the name of the QAP instance from the QAPLIB (the number indicates its size n), *gen* denotes the number of generations performed by the algorithm, *best* displays the fitness of the best solution obtained

QAP Results				
instance	gen	best	average	t in s
ste36a	400	9526 (0.00 %)	9535.6 (0.10 %)	60
nug30	1000	6124 (0.00 %)	6125.6 (0.03 %)	120
kra30a	1600	88900 (0.00 %)	89242.0 (0.39 %)	180
tai60b	1000	608215040 (0.00 %)	608215040.0 (0.00 %)	600
tai60a	1000	7265232 (0.79 %)	7309143.4 (1.39 %)	600
sko100a	3600	152170 (0.11 %)	152253.0 (0.17 %)	1800
tai100b	2000	1186282112 (0.02 %)	1188197862.4 (0.19 %)	1800
tai100a	4000	21334458 (0.99 %)	21372797.6 (1.17 %)	1800
tho150	2400	8145860 (0.14 %)	8160088.0 (0.32 %)	3600
tai150b	1400	500945216 (0.32 %)	502200800.0 (0.57 %)	3600
tai256c	1000	44812252 (-0.31 %)	44839138.3 (-0.25 %)	3600

Figure 5: The Computational Results

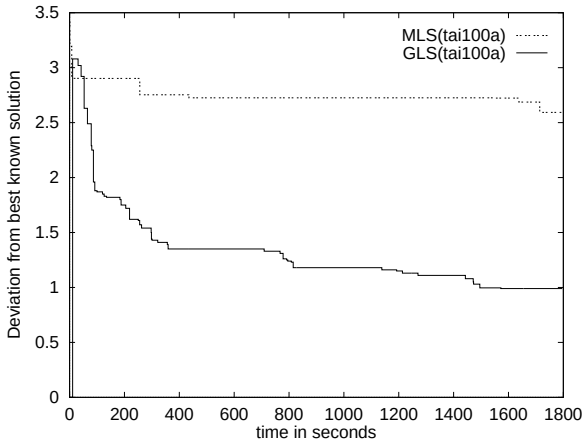

Figure 6: Instance tai100a

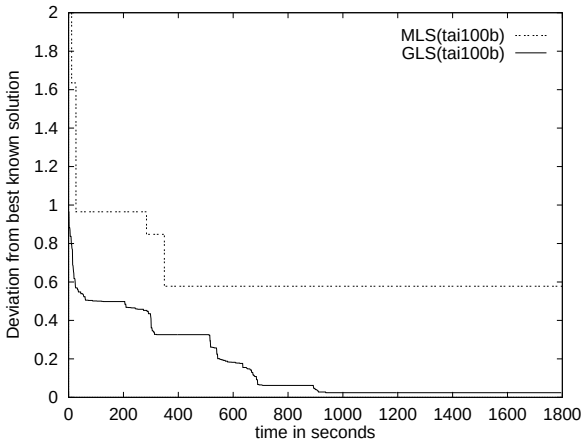

Figure 7: Instance tai100b

out of 10 runs, and *average* shows the average fitness of the final solutions of the 10 runs. In parentheses, the deviation from the best known solution is provided. In the last column of the table, the time in seconds for a single run is displayed, ranging from one minute to an hour of computation time on a DEC Alpha CPU with 233 MHz. In each case, the number of fitness function evaluations (equation (1)), a commonly used GA performance measure, is $0.7 \cdot P$ per generation. However, in the GLS approach the use of this performance measure is not appropriate, since it does not reflect the influence of the local search procedure on the runtime of the algorithm.

The experiments indicate that the straightforward application of GLS is a promising approach to the QAP. For small instances ($n < 60$), the best known upper bounds could be found easily, and for large instances the results are quite remarkable. Under the given time constraints, good qualities could be achieved. In particular, as indicated by the negative deviation from the best known upper bound for the problem tai256c in Figure 5, a new best upper bound for the currently largest problem contained in the QAPLIB could be found.

For two instances, the obtained results are much worse than for the others. The reason for this is that these two instances, tai60a and tai100a, are randomly generated with uniform distribution. The flow dominance for these instances is about 60% while the other instances have a *fd* value higher than 100%. In [23], Taillard mentions that these randomly generated instances are not of general interest and therefore defines a set of non–uniformly generated random instances (tai*b) with the same characteristics as real–life problems.

To illustrate the differences in the search process of the two types of problem instances, the quality of a typical run of the algorithm on tai100a and tai100b is displayed over time in Figure 6 and Figure 7. Furthermore, the typical be-

haviour of a multi-start local search procedure (using the *2-opt* variant employed in the GLS) is also shown to allow a comparison between both methods. It is obvious that the GLS algorithm produces relatively good solutions quickly. This indicates that the genetic algorithm used in the GLS approach is required for a high quality search.

Compared to other published approaches to the QAP, it seems that the proposed GLS algorithm is definitely competitive. In [24], a genetic algorithm is proposed and tested for problems of up to size 30. For the largest problem investigated there (`nug30`), the optimum could not be found. In [15], 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 [18], a GLS 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 13 seconds and 7 seconds, respectively. Comparison studies have been made in [1, 16] 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 [20] 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. For large instances, tabu search (TS) has been shown to be the most powerful technique, in particular when optimal solutions are desired. In [23], Taillard compares several variants of tabu search and a hybrid of a genetic algorithm and tabu search (GH) proposed in [8]. He shows that the variants of TS perform differently on several instances, so no method can be said to be better than all the others. GH, however, seems to be the most robust search algorithm. The results presented in [23] cannot be compared directly with the results presented here, because the running time is measured in the number of iterations of TS, and absolute computation times are not given. After $1000 \cdot n$ iterations of tabu search, the best results reported for problems `tai60b` and `tai100b` are 0.366% and 1.490% respectively, which is significantly worse than our results shown in Figure 5. The results obtained with GH after the same number of iterations is 0.115% and 0.241% respectively, and thus still worse than our results.

The best known solutions for almost all large QAP instances have been found with tabu search, although very long computation times were required. The running time for finding the best known solution for problem `tai100a` with the genetic hybrid (GH) took more than three days on a SUN Sparcstation 10, as reported in [8].

Taillard [23] comments:

> "However, even if they are not optimally solved and if it is still possible to glean minute improvements, we think that these problems (`tai*a`) are not interesting: all the recent heuristics find good solutions and the fact of having improved one of the best known solutions does not constitute a proof of the efficiency of a method."

Hence, research should be focused on developing techniques with a fast convergence to near–optimum solutions for realistic problems in a reasonable time frame. Therefore, we stopped our experiments after a relatively short time to see how well the algorithm performed. The fact that for the largest instance a new best solution was found indicates that the proposed approach may be scalable.

6 CONCLUSIONS

In this paper, an approach for solving large instances of the quadratic assignment problem has been presented. The technique, called genetic local search, was based on a combination of a genetic algorithm and local search, which has been successfully used to solve large instances of the traveling salesman problem. The presented results indicate that genetic local search is also applicable to the QAP, since the algorithm is able to arrive at high quality solutions in short time, especially for instances whose solution space has structures typically found in real-life problems. The proposed approach seems to be competitive to the best heuristic techniques for the QAP such as tabu search and other genetic hybrids. In one hour of computation time on a single workstation, a new solution for the largest publicly available instance was found, which is significantly better than the previously best known solution.

There are several issues for future research. First, a detailed comparison of our approach with tabu search and genetic hybrids would be interesting. Second, alternatives for some components of the algorithm may be investigated, e.g. a variable-k-change heuristic as the local search procedure may further enhance the performance of the algorithm. Third, a detailed analysis of the search space will certainly help to understand when the algorithm performs well and when not. Gaining more insight into the problem structure may lead to an algorithm that performs well even for instances that are considered to be hard to solve.

References

[1] V. Bachelet, P. Preux, and E.-G. Talbi, "Parallel Hybrid Meta–Heuristics: Application to the Quadratic

Assignment Problem," in *Proceedings of the Parallel Optimization Colloquium*, (Versailles, France), 1996.

[2] R. Battiti and G. Tecchiolli, "The Reactive Tabu Search," *ORSA Journal on Computing*, vol. 6, no. 2, pp. 126–140, 1994.

[3] E. D. Brown, L. C. Huntley, and R. A. Spillance, "A Parallel Genetic Heuristic for the Quadratic Assignment Problem," in *Proceedings of the Third Conference on Genetic Algorithms*, (J. D. Schaffer, ed.), pp. 406–415, Morgan Kaufmann, 1989.

[4] E. S. Buffa, G. C. Armour, and T. E. Vollmann, "Allocating Facilities with CRAFT," *Harvard Business Review*, pp. 136–158, March 1964.

[5] R. E. Burkard and U. Fincke, "Probabilistic Asymptotic Properties of Some Combinatorial Optimization Problems," *Discrete Applied Mathematics*, vol. 12, pp. 21–29, 1985.

[6] R. E. Burkard, S. Karisch, and F. Rendl, "QAPLIB – A Quadratic Assignment Problem Library," *European Journal of Operational Research*, vol. 55, pp. 115–119, 1991. Updated Version: `http://www.diku.dk/~karisch/qaplib`.

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

[8] C. Fleurent and J. A. Ferland, "Genetic Hybrids for the Quadratic Assignment Problem," in *Quadratic Assignment and Related Problems*, (P. M. Pardalos and H. Wolkowicz, eds.), pp. 137–187, Amer. Math. Soc., 1994.

[9] B. Freisleben and P. Merz, "A Genetic Local Search Algorithm for Solving Symmetric and Asymmetric Traveling Salesman Problems," in *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation*, (Nagoya, Japan), pp. 616–621, 1996.

[10] B. Freisleben and P. Merz, "New Genetic Local Search Operators for the Traveling Salesman Problem," in *Proceedings of the 4th Conference on Parallel Problem Solving from Nature - PPSN IV*, (H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, eds.), pp. 890–900, Springer, 1996.

[11] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, New York, 1979.

[12] J. J. Grefenstette, "Incorporating Problem Specific Knowledge into Genetic Algorithms," in *Genetic Algorithms and Simulated Annealing*, (L. Davis, ed.), pp. 42–60, Morgan Kaufmann Publishers, 1987.

[13] A. Kolen and E. Pesch, "Genetic Local Search in Combinatorial Optimization," *Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science*, vol. 48, pp. 273–284, 1994.

[14] Y. Li, P. M. Pardalos, and M. G. C. Resendre, "A Greedy Randomized Adaptive Search Procedure for the Quadratic Assignment Problem," in *Quadratic Assignment and Related Problems*, (P. M. Pardalos and H. Wolkowicz, eds.), pp. 237–261, Amer. Math. Soc., 1994.

[15] V. Maniezzo, A. Colorni, and M. Dorigo, "The Ant System Applied to the Quadratic Assignment Problem," Tech. Rep. 94/28, IRIDIA, Université de Bruxelles, 1994.

[16] V. Maniezzo, M. Dorigo, and A. Colorni, "Algodesk: An Experimental Comparison of Eight Evolutionary Heuristics Applied to the Quadratic Assignment Problem," *European Journal of Operational Research*, vol. 81, pp. 188–204, 1995.

[17] P. Merz and B. Freisleben, "Genetic Local Search for the TSP: New Results," in *Proceedings of the 1997 IEEE International Conference on Evolutionary Computation*, (Indianapolis), 1997.

[18] H. Mühlenbein, "Parallel Genetic Algorithms, Population Genetics and Combinatorial Optimization," in *Proceedings of the Third International Conference on Genetic Algorithms*, (Schaffer, ed.), pp. 416–421, Morgan Kaufmann, 1989.

[19] V. Nissen and H. Paul, "A Modification of Threshold Accepting and its Application to the Quadratic Assignment Problem," *OR Spektrum*, vol. 17, pp. 205–210, 1995.

[20] V. Nissen, "Solving the Quadratic Assignment Problem with Clues from Nature," *IEEE Transactions on Neural Networks*, vol. 5, no. 1, pp. 66–72, 1994.

[21] J. Skorin-Kapov, "Tabu Search Applied to the Quadratic Assignment Problem," *ORSA Journal on Computing*, vol. 2, no. 1, pp. 33–45, 1990.

[22] J. Skorin-Kapov, "Extensions of a Tabu Search Adaptation to the Quadratic Assignment Problem," *Computers and Operations Research*, vol. 21, no. 8, pp. 855–865, 1994.

[23] E. D. Taillard, “Comparison of Iterative Searches for the Quadratic Assignment Problem,” *Location Science*, vol. 3, pp. 87–105, 1995.

[24] D. M. Tate and A. E. Smith, “A Genetic Approach to the Quadratic Assignment Problem,” *Computers and Operations Research*, vol. 22, no. 1, pp. 73–83, 1995.

[25] N. L. J. Ulder, E. H. L. Aarts, H. J. Bandelt, P. J. M. van Laarhoven, and E. Pesch, “Genetic Local Search Algorithms for the Traveling Salesman Problem,” in *Parallel Problem Solving from Nature - Proceedings of 1st Workshop, PPSN I*, (H. P. Schwefel and R. Männer, eds.), (Berlin, Germany), pp. 109–116, Springer, 1991.

[26] T. E. Vollmann and E. S. Buffa, “The Facilities Layout Problem in Perspective,” *Management Science*, vol. 12, no. 10, pp. 450–468, 1966.

[27] T. Yamada and R. Nakano, “Scheduling by Genetic Local Search with Multi-Step Crossover,” in *Proceedings of the 4th Conference on Parallel Problem Solving from Nature - PPSN IV*, (H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, eds.), pp. 960–969, Springer, 1996.