Local Search and Metaheuristics for the Quadratic Assignment Problem

Thomas Stützle
Intellectics Group, Department of Computer Science
Darmstadt University of Technology
email: stuetzle@informatik.tu-darmstadt.de
WWW: www.intellektik.informatik.tu-darmstadt.de/˜tom

Marco Dorigo
IRIDIA, Université Libre de Bruxelles
email: mdorigo@ulb.ac.be
WWW: iridia.ulb.ac.be/˜mdorigo

1 Introduction

The quadratic assignment problem (QAP) is an important problem in theory and practice. It was , which was introduced by Koopmans and Beckmann in 1957 [28] and is a model for many practical problems like backboard wiring [53], campus and hospital layout [15, 17], typewriter keyboard design [9], scheduling [23] and many others [16, 29] can be formulated as QAPs. Intuitively, 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:

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

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. [1]

[1] A more general form of the QAP was introduced by Lawler [31], but here we will focus on the above given

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:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{l=1}^{n} \sum_{k=1}^{n} a_{ij} b_{kl} x_{ik} x_{jl} \tag{2}$$

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 [48]; even finding a solution within a factor of $1 + \epsilon$ of the optimal one remains $\mathcal{NP}$-hard [48, 45]. It is considered as one of the hardest optimization problems, because exact algorithms show a very poor performance on the QAP. In fact, the largest, non-trivial instance solved to optimality today is of size $n = 30$! Therefore, to practically solve the QAP one has to apply heuristic algorithms. Several such heuristic algorithms have been proposed which include algorithms ranging from simple iterative improvement to sophisticated metaheuristic implementations.

In this report we will give an overview of local search and metaheuristics for the Quadratic Assignment Problem. For a more complete overview of the QAP, including exact algorithms, applications and mathematical results we refer to the overview articles by Finke, Burkard and Rendl [18] and by Pardalos, Rendl and Wolkowicz [44] and the book be Cela [11].

The report is structured as follows. In Section 2 we give an overview of existing benchmark problems, a classification of these benchmark problems, and indicate some results from a search space analysis of these problems. Next, in Section 3 we explain the functioning of local search for the QAP and indicate some local search variants. Section 4 contains an overview over metaheuristic approaches to the QAP and we conclude in Section 5.

2 Benchmark instances

Approximate algorithms are commonly tested on benchmark instances which posess interesting properties, have been attacked by a number of algorithmic approaches, or stem from applications. In the QAP case, a large number of such benchmark instances is available via QAPLIB, a library for research on the QAP. QAPLIB contains currently over 100 instances that have been used in earlier researches and in part they stem from real applications like hospital layout (like the `kra30*` or the `els19` instances), typewriter design (like the `bur26*` instances), etc. In addition, QAPLIB also contains a number of other resources like pointers to literature, some source codes and links for QAP related research including a list of people with research interests in the QAP.

Regarding the instances available from QAPLIB, disadvantages are that only a small number of instances is available that are large enough to pose a real challenge to state-of-the-art metaheuristics. Second, the instances in QAPLIB do not vary systematically in their characteristics and therefore their utility is limited for analyzing the performance of metaheuristics in depen-

Koopmans–Beckmann formulation because it is much more widely used.

dence of the instance characteristics. Nevertheless, QAPLIB is very useful and certainly the first address for QAP related information on the WWW.

Further benchmark instances may be obtained by encodings of related problems like the Traveling Salesman Problem or Graph Partitioning into QAP, although such instances are not very commonly used.

2.1 Classification of benchmark instances

For the QAP it is known that there are several different types of instances and that the particular instance type has a considerable influence on the performance of heuristic methods. According to Taillard [60], the instances of QAPLIB can be classified into the following four classes.

(i) **Unstructured, randomly generated instances**: Instances with the distance and flow matrix entries generated randomly according to a uniform distribution. 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.

(ii) **Grid-based distance matrix**: In this class of instances the distance matrix stems from a $n_1 \times n_2$ grid and the distances are defined as the Manhattan distance between grid points. These instances have multiple global optima (at least 4 in case $n_1 \neq n_2$ and at least 8 in case $n_1 = n_2$) due to the definition of the distance matrices.

(iii) **Real-life instances**: Instances from this class are "real-life" instances from practical applications of the QAP. Real-life problems have in common that the flow matrices have many zero entries and the remaining entries are clearly not uniformly distributed. The matrix entries exhibit a clear structure which is the main difference between these real-life instances and the randomly generated instances from class (i).

(iv) **Real-life-like instances**: Since the real-life instances in QAPLIB are mainly of a rather small size, a particular type of randomly generated problems has been proposed in [60]. These instances are generated in such a way that the matrix entries resemble the distributions found in real-life problems.

Let us add two further remarks on the instances from QAPLIB. First, some of the instances were generated randomly with known optimal solutions with a generator proposed by Li and Pardalos [32]. This allows to evaluate the solution quality of metaheuristics with respect to known optimum also on large instances, which otherwise is not possible because of the poor behavior of exact algorithms. Yet, these instances often tend to be somewhat easier than other benchmark instances and for many large instances "pseuod-optimal" solutions [60] are known. Here we mean with "pseudo-optimal" that many algorithms have found the same best known solutions and one may assume that these are actually the optimal ones. Second, it has to be mentioned that for many QAP instance types, the difference between the worst and the optimal solution becomes arbitrarily small with a probability tending to one as the problem size tends to infinity [8, 21, 46, 47, 58].

2.2 Instance classes and search space characteristics

To differentiate among the classes of QAP instances the flow dominance (*fd*) can be used. It is defined as the coefficient of variation of the flow matrix entries multiplied by 100.

$$\mathit{fd}(A) = 100 \cdot \frac{\sigma}{\mu}, \text{ where} \tag{3}$$

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

A high flow dominance indicates that a large part of the overall flow is exchanged among relatively few facilities. Randomly generated problems according to a uniform distribution will have a rather low flow dominance whereas real-life problems, in general, have a rather high flow dominance. A disadvantage of the flow dominance is that it captures only the structure of one of two matrices, neglecting that of the distance matrix. Therefore, analogously to the flow dominance, we can also use a *distance dominance* (*dd*).

In general, real life problems often have many zero entries and, hence, the sparsity of the matrix can give an indication of the instance type. Let n_0 be the number of "0" entries in a matrix, then we define its sparsity sp as $sp = n_0/n^2$.

In Table 1 and 2 are given the flow and the distance dominance and the sparsity of the sparser of the two matrices (typically, at most one matrix of these QAP instances has a sparsity larger than 0.1) for some instances of QAPLIB, ordered according to the four instance classes (the other table entries are explained in the following). In general, class iii and iv instances have the highest dominance values; the instances of class ii still habe larger flow dominance than the randomly generated instances of class i.

Today it is widely agreed that the performance of metaheuristics depends strongly on the shape of the underlying search space. Central to the search space analysis of combinatorial optimization problems is the notion of *fitness landscape* [52, 64]. Intuitively, the fitness landscape can be imagined as a mountainous region with hills, craters, and valleys. The performance of metaheuristics strongly depends on the ruggedness of the landscape, the distribution of the valleys, craters and the local minima in the search space, and the overall number of the local minima.

Formally, the fitness landscape is defined by

(1) the set of all possible solutions $\mathcal{S}$;

(2) an objective function that assigns to every $s \in \mathcal{S}$ a fitness value $f(s)$;

(3) a distance measure $d(s, s')$ which gives the distance between solutions s and s'.

The fitness landscape determines the shape of the search space as encountered by a local search algorithm.

Table 1: Given are the instance identifier, the flow, the distance dominance and the sparsity of some QAPLIB instances (columns 1 to 4). The dominance values are calculated for the first *A* and the second *B* matrix as given in QAPLIB. The number in the instance identifier is the instance dimension. The instances are ordered according to the 4 classes described in Section 4. The remaining entries give summary results of an analysis of the fitness-distance correlation of the QAP search space (see Section 5). In particular, N_{opt} is the number of pseudo-optimal solutions found, avg^{ls}_{d-opt} and avg^{ils}_{d-opt} are the average distance to the closest optimum solution for local search and iterated local search executed for n iterations, respectively, and r_{ls} and r_{ils} are the correlation coefficients for the solution cost versus the distance to the closest pseudo-optimal solution.

instance	$dd(A)$	$fd(B)$	sp	N_{opt}	avg^{ls}_{d-opt}	avg^{ils}_{d-opt}	r_{ls}	r_{ils}
			unstructured, randomly generated i					
`tai20a`	67.02	64.90	0.015	1	18.78	18.62	0.065	0.088
`tai25a`	61.81	64.29	0.016	1	23.83	23.64	0.064	0.059
`tai30a`	58.00	63.21	0.013	1	28.69	28.32	0.100	0.208
`tai35a`	61.64	61.57	0.010	1	33.75	33.61	0.041	0.054
`tai40a`	63.10	60.23	0.009	1	38.86	38.81	0.020	0.107
`tai60a`	61.41	60.86	0.011	1	58.82	58.71	0.025	0.006
`tai80a`	59.22	60.38	0.009	1	78.90	78.77	0.022	0.049
`rou20`	65.65	64.43	0.010	1	18.50	18.10	0.124	0.114
			Instances with grid-distances ii					
`nug30`	52.75	112.48	0.316	4	25.93	23.81	0.262	0.406
`tho30`	59.25	137.86	0.484	4	26.27	24.86	0.328	0.472
`tho40`	53.20	155.54	0.585	4	36.11	35.21	0.194	0.273
`sko42`	51.96	108.48	0.292	4	37.96	35.18	0.302	0.499
`sko49`	51.55	109.38	0.304	8	44.58	43.40	0.213	0.234
`sko56`	51.46	110.53	0.305	4	51.62	49.51	0.254	0.448
`sko64`	51.18	108.38	0.308	8	58.88	56.33	0.303	0.353
`sko72`	51.14	107.13	0.299	4	67.38	65.31	0.264	0.284

Two important measures have been proposed to analyze the fitness landscape. One is the fitness distance correlation (FDC) [6, 26]. The FDC measures between the cost and the distance of solutions to the closest global optima or best-known solutions if global optima are not available. Given a set of cost values $C = \{c_1, \ldots, c_m\}$ and the corresponding distances $D = \{d_1, \ldots, d_m\}$ to the closest global optimum the correlation coefficient is defined as:

$$r(C, D) = \frac{c_{CD}}{s_C \cdot s_D} \tag{4}$$

where

$$c_{CD} = \frac{1}{m} \sum_{i=1}^{m} (c_i - \bar{c})(d_i - \bar{d}) \tag{5}$$

and $\bar{c},\ \bar{d}$ are the average cost and the average distance, s_C and s_D are the standard deviations of the costs and distances, respectively.

The second measure analyses the *ruggedness* of the fitness landscape. To measure the rugged-

Table 2: Given are the instance identifier, the flow, the distance dominance and the sparsity of some QAPLIB instances (columns 1 to 4). The dominance values are calculated for the first *A* and the second *B* matrix as given in QAPLIB. The number in the instance identifier is the instance dimension. The instances are ordered according to the 4 classes described in Section 4. The remaining entries give summary results of an analysis of the fitness-distance correlation of the QAP search space (see Section 5). In particular, N_{opt} is the number of pseudo-optimal solutions found, avg^{ls}_{d-opt} and avg^{ils}_{d-opt} are the average distance to the closest optimum solution for local search and iterated local search executed for n iterations, respectively, and r_{ls} and r_{ils} are the correlation coefficients for the solution cost versus the distance to the closest pseudo-optimal solution.

instance	$dd(A)$	$fd(B)$	sp	N_{opt}	avg^{ls}_{d-opt}	avg^{ils}_{d-opt}	r_{ls}	r_{ils}
real-life instances iii								
`bur26a`	15.09	274.95	0.223	96	21.12	20.15	0.027	0.457
`bur26b`	15.91	274.95	0.223	690	21.26	19.72	0.021	0.678
`bur26c`	15.09	228.40	0.257	96	22.31	15.39	0.569	0.867
`bur26d`	15.91	228.40	0.257	790	20.29	18.19	0.471	0.787
`bur26e`	15.09	254.00	0.312	97	18.43	14.91	0.479	0.853
`bur26g`	15.09	279.89	0.211	96	18.47	13.89	0.666	0.876
`chr25a`	424.27	57.97	0.883	2	22.92	21.71	0.252	0.359
`els19`	52.10	531.02	0.637	1	16.85	13.76	0.550	0.654
`kra30a`	49.22	149.98	0.6	257	25.23	24.10	0.251	0.413
`kra30b`	49.99	149.98	0.6	128	24.83	23.25	0.312	0.379
`ste36a`	55.65	400.30	0.707	8	30.98	28.37	0.295	0.504
`ste36b`	100.79	400.30	0.707	8	29.59	24.76	0.381	0.778
real-life like instances iv								
`tai20b`	128.25	333.23	0.410	1	17.32	14.69	0.420	0.576
`tai25b`	87.02	310.40	0.387	1	21.65	23.83	0.456	0.703
`tai30b`	85.20	323.91	0.432	1	27.71	25.47	0.264	0.518
`tai35b`	78.66	309.62	0.524	1	32.04	30.17	0.328	0.525
`tai40b`	66.75	317.22	0.503	1	37.76	35.39	0.329	0.626
`tai50b`	73.44	313.91	0.548	1	47.94	45.39	0.156	0.363
`tai60b`	76.83	317.82	0.548	1	56.88	52.28	0.366	0.540
`tai80b`	64.05	323.17	0.552	1	77.47	75.56	0.150	0.457
`tai100b`	80.42	321.34	0.552	1	95.23	92.34	0.546	0.608

ness of fitness landscapes the autocorrelation function has been proposed [64]. It is given by

$$\rho(d) = \frac{\mathbf{E}[(f(s) - f(s'))^2]_{d(s,s')=d}}{\mathbf{E}[f^2] - (\mathbf{E}[f])^2} \tag{6}$$

$\mathbf{E}[X]$ denotes the expectation of a random variable (note that here the objective function value of a particular solution is considered to be a random variable) and $\rho(d)$ is the correlation coefficient between two solutions that are at distance d. Hence, $\rho(1)$ gives the correlation between two neighboring solutions. If this correlation is very high, the average cost difference between two neighboring solutions is relatively small. Thus, the landscape is only little rugged and a local search algorithm should show good performance on such a problem. Yet, if the correlation is low, the solution quality of neighboring solutions may differ very strongly. Note that in such a case the solution quality of the current solution gives only a very limited indication of the quality of neighboring solutions (in fact, if the correlation is zero, the current solution gives no information on the quality of neighboring solutions). The correlation can be calculated exactly for some problems if the probability distribution of $f(s)$ can be derived from the underlying instance generating process [2]. Yet, for a given particular instance often the correlation coefficient has to be estimated. To do so, in [64] it is proposed to perform a random walk over the fitness landscape, to interpret the trajectory as a time series and to calculate the empirical autocorrelation function $\hat{\rho}(d)$.

To summarize the information of the autocorrelation function, several measures have been proposed. One of these is the *correlation length* l [52] which is

$$l = -\frac{1}{\ln \hat{\rho}(1)} \tag{7}$$

The correlation length gives information on how far from the current solution – on average – one has to move such that there is not any more a significant correlation between the cost values of solutions. Clearly, smoother landscapes will have a longer correlation length. A similar measure is the *autocorrelation coefficient* [1] defined as $\xi = 1/(1 - \rho(1))$.

Here we present some results of an analysis of the fitness distance correlation for different QAP instances. A natural distance measure between solutions is the number of facilities which are placed on distinct locations in two solutions ϕ and ϕ', i.e., $d(\phi, \phi') = |\{i \mid \phi_i \neq \phi'_i\}|$, which is a direct extension of the Hamming distance. In the FDC analysis we measure the distance to the a globally optimal solution if these are available. Where optimal solutions are not available, we measure the distance to the best known solution. These best known solutions are conjectured to be optimal for instances of class ii, iii, and iv with up to 80 facilities, because they are *pseudo-optimal*.

For the FDC analysis for the QAP one has to take into account the fact that the instances can have multiple optimal solutions. For example, this is known to be the case for instances where the distance matrix is defined by the distance between grid positions (class ii), where it is further known that these optimal solutions may be at the maximally possible distance from the other optimal solutions (this is due to symmetries in the distance matrix). Hence, on instances with multiple global optima or pseudo-optimal solutions we measure the distance to the *closest global optimum*. Unfortunately, the exact number of global optima for these instances is not known. Therefore, we determined in preliminary runs of an Iterated Local Search (ILS) algorithm a (possibly large) number of pseudo-optimal solutions: For small instance with $n < 40$ we stopped searching for more pseudo-optimal solutions if in 1000 runs of our ILS algorithm, each run of at least 500 iterations, we did not find any more a pseudo-optimal solution which differs from any previously found pseudo-optimal solution; only on the larger instances of type ii we searched for the expected number of optimal solutions (either four or eight depending on the grid dimension). In this process we have found only one single pseudo-optimal solution for the instances of classes i and iv. Therefore, we conjecture that these instances have unique optimal solutions. The number of pseudo-optimal solutions found for each instance are indicated by N_{opt} in Tables 1 and 2.

We run two experiments in the fitness-distance analysis: In a first experiment (denoted as E-ls) we generated 5000 local optima (identical solutions at distance 0 among the generated local optima have been eliminated) with a 2-opt local search algorithm like described in the next section and measured the distance to the closest pseudo-optimal solution; in a second series of experiments (denoted as E-ils) we run 1000 times an ILS algorithm for n iterations (indicated by ILS(n)), again eliminating identical solutions. This second series of experiments is motivated by the observation that for the Travelling Salesman Problems the FDC among higher quality solutions is larger and the average distance to the globally optimal solutions is smaller [6].

Tables 1 and 2 give, in addition to the flow and distance dominance and the sparsity, the number of pseudo-optimal solutions identified in the preliminary runs (N_{opt}), the average distances of the local minima to the closest global optimum in E-ls (avg^{ls}_{d-opt}) and E-ils (avg^{ils}_{d-opt}), and the empirical FDC coefficients found in E-ls and E-ils (r_{ls} and r_{ils}, respectively).

The fitness-distance analysis shows clear differences among the behavior for the different problem classes. For class i, all correlation coefficients are close to zero. Also the better solutions generated by ILS(n) do not show a much higher correlation. Regarding the average distances from the global optima, it can be observed that avg^{ls}_{d-opt} and avg^{ils}_{d-opt} are very large for these QAP instances, close to the maximal possible value which is n and the difference between avg^{ls}_{d-opt} and avg^{ils}_{d-opt} is minimal. Differently, for most instances of the other three classes, significant correlations exist. In fact, all correlations are statistically significant at the $\alpha = 0.05$ level with the only exception of r_{ls} for instances `bur26a` and `bur26b`. Comparing r_{ls} and r_{ils} for instances of classes $ii - iv$ we find that r_{ils} is typically much larger than r_{ls}. It is also notable that avg^{ils}_{d-opt} is always smaller than avg^{ls}_{d-opt} by a factor of about 0.96. It is also in these two latter points where the instances of classes $ii - iv$ show significant differences to those of class i. Comparing the instances of classes $ii - iv$ we find that the correlation coefficients for instances of classes iii and iv are typically higher than those of class ii, which may indicate that the instances of these later two classes are easier for the ILS algorithms than those of class ii. Additionally, one can observe that for the instances with a high flow or distance dominance and high sparsity also a significant FDC can be observed. Hence, these more simpler measures already give a strong indication whether a significant fitness-distance correlation can be expected.

In summary we can conclude that—on average—the better the solution quality the closer a solution is to an optimal solution for the instances of classes $ii - iv$. These instance show a structure in the following sense: The more locations of facilities a solution has in common with an optimal solution, the better will be that solution.

3 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 difference $\Delta(\psi, r, s)$ obtained by exchanging facilities ψ_s and ψ_r can be computed in $O(n)$, using the following equation [60]:[2]

$$\begin{aligned}\Delta(\psi, r, s) = b_{rr} \cdot (a_{\psi_s\psi_s} - a_{\psi_r\psi_r}) + b_{rs} \cdot (a_{\psi_s\psi_r} - a_{\psi_r\psi_s}) + \\ b_{sr} \cdot (a_{\psi_r\psi_s} - a_{\psi_s\psi_r}) + b_{ss} \cdot (a_{\psi_r\psi_r} - a_{\psi_s\psi_s}) + \\ \sum_{k=1, k\neq r,s}^{n} (b_{kr} \cdot (a_{\psi_k\psi_s} - a_{\psi_k\psi_r}) + b_{ks} \cdot (a_{\psi_k\psi_r} - a_{\psi_k\psi_s}) + \\ b_{rk} \cdot (a_{\psi_s\psi_k} - a_{\psi_r\psi_k}) + b_{sk} \cdot (a_{\psi_r\psi_k} - a_{\psi_s\psi_k})) \end{aligned} \quad (8)$$

The effect of a particular swap can be evaluated faster using information from preceding iterations. Let ψ' 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:

$$\begin{aligned}\Delta(\psi', u, v) = \Delta(\psi, r, s) + (b_{ru} - b_{rv} + b_{sv} - b_{su}) \cdot (a_{\psi_s\psi_u} - a_{\psi_s\psi_v} + a_{\psi_r\psi_v} - a_{\psi_r\psi_u}) \\ (b_{ur} - b_{vr} + b_{vs} - b_{us}) \cdot (a_{\psi_u\psi_s} - a_{\psi_v\psi_s} + a_{\psi_v\psi_r} - a_{\psi_u\psi_r})\end{aligned} \quad (9)$$

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 9); 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* [5] 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 variable depth local search algorithm was proposed by Murthy, Pardalos, and Li [41, 44] and it was shown to be PLS-complete [25] (PLS is a particular complexity class for local search algorithms, where PLS is the class of local search algorithms for which each local search step requires polynomial time, but it is not specified how many exchange steps they take to reach a local optimum). PLS completeness was also shown for a 2-opt local search algorithm for the QAP [49].

[2] 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 $b_{\psi_i\psi_j} = b_{\psi_j\psi_i}$.

4 Metaheuristic approaches to the QAP

A major disadvantage of local search is that it may get stuck in poor quality local optima of the search space. To improve the performance of local search, metaheuristics have been proven as extremely sucessful for the QAP and for a wide variety of other combinatorial problems. In this section we present an overview over the available metaheuristic approaches to tackle the QAP. Because it is infeasible to describe all the algorithms in detail, we proceed as follows: for each metaheuristic implemented in the Metaheuristic Network we describe one particular approach in some detail.[3] For the other approaches we only give some pointers to the literature and refer the reader to the related articles for further details.

4.1 Simulated Anealing

Simulated Annealing (SA) was one of the first available metaheuristics. Therefore it is not astonishing that it was also the first one to be applied to the QAP by Burkard and Rendl in 1984 [10]. Following this implementation, some few others were proposed and currently the one due to Conolly [13] appears to be the best performing. Conolly compared a large number of different neighborhood examination schemes from random to sequential ones and many different types of annealing schedules. The final SA algorithm has a quite particular shape. First, it uses a sequential search of the neighborhood using a fixed ordering. The advantage of such a scheme is that especially at low temperatures one can assure that the whole neighborhood is searched. A second particularity is that the SA was found to perform best using a fixed temperature! Still the optimal temperature remains to be fixed. To this aim, Conolly proposed a scheme of how to adapt automatically the temperature during the run of the algorithm.

An additional SA approach for the QAP was proposed earlier by Wilhelm and Ward [65]. The tradeoff between the computation time and the solution quality t taken by (several) short SA runs within a fixed time budget T (given time T, one can run T/t times a short SA), is investigated by Laursen [30]. Thonemann and Bölte have proposed an improved SA algorithm for the QAP. Finally, also threshold accepting, a metaheuristic closely related to SA, was applied to the QAP by Nissen and Paul [43].

4.2 Tabu Search

Probably the best known tabu search algorithm for the QAP is the robust tabu search (RoTS) algorithm of Taillard [59]. This algorithm is based on the 2-opt best-improvement local search algorithm we have described in the previous section. As tabu attributes, the algorithms uses assignment of facilities to specific objects, that is a tabu attribute $t(i, j)$ refers to the fact that it is forbidden to assign facility i to location j.

In RoTS a random solution is chosen as the initial solution. At each iteration k, the best, non-tabu solution of $\mathcal{N}(\psi_k)$ (the neighborhood of the current solution) is chosen as the new solution, even if it is worse than the current solution. A neighboring solution that places facilities i

[3] Here we assume that the reader is familiar with all the metaheuristics.

and j on locations r and s, respectively is tabu, if in the past tl iterations local search moves were done that removed facility i from location r and facility j from location j. Here, tl is a parameter called *tabu tenure* or *tabu list length*. This tabu condition is only overridden if such a move would lead to a new best solutions since the start of the search; this latter rule implements the so called aspiration criterion. RoTS has one additional rule which introduces a type of diversification mechanism to the local search and which is important to achieve very good computational results in the long run: If a facility i has not been placed on a specific location r during the last u iterations, any move that does not place facility i on location r is forbidden. In fact, in such a situation the algorithm forces to place a facility on one particular location.

Certainly, the performance of RoTS depends strongly on the parameter settings. Taillard proposed the following parameters: the tabu tenure tl is chosen randomly during the search from the interval $[0.9 \cdot n, 1.1 \cdot n]$ (in fact, a new random value for tl is chosen every $2.2 \cdot n$ iterations). This random choice of a tabu list length from a specific interval is actually the reason for calling this algorithm *robust* tabu search, because the author claimed that this would make the algorithm more robust with respect to a particular choice of tl. The second parameter, u, has to be chosen larger than the size of the neighborhood. Here, good settings appear to be in the range from $2 \cdot n^2$ and $5 \cdot n^2$ [59, 60].

There exists a variety of other tabu search implementations for the QAP. The first such implementation is due to Skorin-Kapov in 1990 [50]. Some extensions of this early tabu search algorithms were proposed again by Skorin-Kapov in [51]; this research direction also includes a massively parallel tabu search implementation [12]. A particularly interesting tabu search variant, the *reactive* tabu search (RTS), was proposed by Battiti and Tecchiolli [3]. The main idea of RTS is to use the search history to dynamically adapt the tabu tenure during the run of an algorithm. In addition, the RTS uses a diversification mechanism based on iteratively applying a number of random moves if the search is deemed to be stuck.

A comparison of RoTs, RTS, strict tabu search algorithms and additionally a genetic hybrid algorithm (described below) is found in [60].

4.3 Iterated Local Search

A first application of iterated local search (ILS) to the QAP is reported in [54]. To apply ILS to the QAP four procedures have to defined. The role of the procedures is to (i) generate an initial solution, (ii) implement a local search algorithm, (iii) implement a solution perturbation, and, (iv) implement an acceptance criterion. Several variants of ILS algorithms, which mainly differ in the type of acceptance criterion chosen but also include population-based extensions, were presented in [54]. Here, we only give the details of a "basic" ILS approach.

This basic ILS starts from a random initial solution.

For the local search a first-improvement 2-opt algorithm, as described in Section 3 was used. To avoid that in each local search iteration the full neighborhood has to be searched, a technique was adopted that is called *don't look bits*; originally, don't look bits were proposed to speed up local search algorithms for the TSP [5, 38]. When applied to the QAP, a don't look bit is associated with every item. When starting the local search, all don't look bits are turned off (set to 0). If during the neighborhood scan for an item no improving move is found, the don't look bit is turned on (set to 1) and the item is not considered as a starting item for a neighborhood scan in the next iteration. Yet, if an item is involved in a move and changes its location, the don't look bit is turned off again. The don't look bits restrict the attention to the most interesting part of the local search where still further improvement can be expected. Hence, after a solution perturbation, which is described in the next paragraph, only the don't look bits of items which change their location due to the mutation are reset to 1. Together with this resetting strategy of the don't look bits, the speed of the first-improvement local search could be increased considerably.

The solution perturbation exchanges k randomly chosen items. This corresponds to a random move in the k-opt neighborhood. To make the particular choice of the kick-move strength, i.e., the value of k, more robust, a scheme for changing k as used in variable neighborhood search was adopted [24].

In the basic version of the ILS the *Better*(ψ, ψ'') acceptance criterion was used. It is defined as follows:

$$s = \textit{Better}(\psi, \psi'') = \begin{cases} \psi'' & \text{if } J_{\psi''} < J_{\psi} \\ \\ \psi & \text{otherwise} \end{cases} \tag{10}$$

This acceptance criterion actually implements a randomized descent in the space of locally optimal solutions. However, experimental results in [54] suggest that with other acceptance criteria, which allow also moves to worse local optima, yield better performance than *Better*(s, s'').

4.4 Ant Colony Optimization

When applying ant colony optimization (ACO) to the QAP, first we have to define a solution construction mechanism and to decide how pheromones should be used. Here we present the details of how $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS), a particular ACO algorithm, is applied to the QAP. When applying $\mathcal{MM}$AS to the QAP, the pheromone trails τ_{ij} correspond to the desirability of assigning a facility i to a location j. For the solution construction, it is convenient to use a preordering of the facilities (or, equivalently, the locations) and assign facilities in the given order. Then, at each construction step an ant probabilistically decides on which location the next facility should be put. These construction steps are then repeated until a complete assignment is obtained.

In $\mathcal{MM}$AS, 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:

$$p_{ij}^{k}(t) = \frac{\tau_{ij}(t)}{\sum_{l \in \mathcal{N}_i^k} \tau_{il}(t)} \qquad \text{if } j \in \mathcal{N}_i^k, \tag{11}$$

where $\mathcal{N}_i^k$ is the feasible neighborhood of ant k, that is, the set of locations that are not yet assigned a facility.

Once all ants have constructed a solution, the pheromone trails are updated according to

$$\tau_{ij}(t+1) = \rho \cdot \tau_{ij}(t) + \Delta\tau_{ij}^{best} \tag{12}$$

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

$$\Delta\tau_{ij}^{best} = \begin{cases} 1/J_{\psi}^{best} & \text{if facility } i \text{ is assigned to location } j \text{ in solution } \psi^{best} \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

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 [55, 57]. Additionally, during or after the pheromone update one has to ensure that the pheromone trail strength respects the lower ($\tau_{\min}$) and the upper ($\tau_{\max}$) pheromone trail limits that are used by $\mathcal{MM}$AS to avoid search stagnation (see [55, 57] for details). 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 $\tau_{ij} = \tau_{\min}$.

Additionally, $\mathcal{MM}$AS uses an occasional re-initialization of the pheromone trails to $\tau_{\max}$ to increase the search diversification. For a detailed description including parameter settings etc. we refer to [55, 57]

There exist a number of other applications of ACO algorithms to the QAP. These include the application of ant system, the first ACO algorithm, by Maniezzo, Colorni and Dorigo [36], an extension of this application by Maniezzo and Colorni [35], the ANTS algorithm of Maniezzo [34], the FANT algorithm by Taillard and Gambardella [61] and an ant algorithm by Gambardella, Taillard, and Dorigo [22] (yet, this algorithm does not fall into the framework of ACO, because it does not use solution construction).

For a short overview if the different existing ACO algorithms we refer to the paper by Stützle and Dorigo [56], where also an experimental comparison of some of the algorithms including the RoTS algorithm can be found.

4.5 Evolutionary Algorithms

A number of different evolutionary algorithms, mostly based on genetic algorithms were implemented for the QAP. During a long time, the most prominent of the approaches was the genetic hybrid (GH) approach by Fleurent and Ferland [19]. They adapt a genetic algorithm to the QAP by using a special crossover operator and the mutation is replaced by a short run of a tabu search algorithm (similar to the RoTS, but with fixed tabu tenure).

The crossover operator, originally proposed by Tate and Smith [62] for a pure genetic algorithm, takes two solutions and generates a new one as follows. First, all the objects that are assigned the same location in both parent solutions are directly copied to the new (child) solution. In a second step, locations that are still free are assigned objects. Actually this second step takes place in two sub-steps. In the first sub-step, unassigned locations are scanned from left to right. For an unassigned location, if possible, one of the two objects is assigned at random that is in this same location in the parent solutions; if this is not possible, because the objects already occur in the solution, the location is left free. In the second sub-step, finally, the remaining objects are assigned randomly to the free locations.

After a new solution is generated, this solution is improved by a short RoTS run and the best solution found during the local search is returned.

Since in the original paper not all parameters are specified, we rather present the settings chosen by Taillard in his implementation of the GH: For instances up to $n = 50$, $2 \cdot n$ solutions are used, for larger instances a constant number of 100 solutions is used. The algorithm is initialized with random solutions that are immediately improved by the RoTS, which is run for $4 \cdot n$ steps. At each iteration two new solutions are generated and the two worst ones are removed. For the selection to the crossover a rank-based selection is done, where the ith worst solution has a probability of $2 \cdot (p - i)/p \cdot (p - 1)$ of being selected.

In [60] a re-implementation of GH was compared to several tabu search variants and it was shown that the GH performed particularly well on structured QAP instances like those of classes (iii) and (iv) described in Section 2.

There are several other approaches for the QAP which are based on evolutionary algorithms. These include standard genetic algorithms without including local search [62], parallel genetic algorithms [7, 40], evolutions strategies [42], and genetic local search algorithms in the spirit of GH [39]. The probably best performing variant of evolutionary algorithms is the memetic algorithm proposed by Merz and Freisleben [39].

4.6 Other metaheuristic implementations for the QAP

There are number of applications of other metaheuristics to the QAP. Guided local search was applied by Voudouris in his PhD thesis [63]. A deterministic local search approach, based on a tabu search, with diversification features was applied to the QAP by Kelly, Laguna, and Glover [27]. Using a deterministic local search has the advantage that only one single run needs to be performed to evaluate the algorithm, but this may also cause poor and not very robust behavior.

Results for applying GRASP to the QAP are presented in [33]; this algorithm seems to be outperformed by current state-of-the-art algorithms. A constructive multi-start algorithm that exploits a memory on the past search process for the solution construction was presented by Fleurent and Glover [20]. Finally, a scatter search algorithm is presented by Cung, Mautor, Michelon, and Tavares [14]; this algorithm obtained very good results with a local search based on short runs of a tabu search algorithm.

Despite the large number of metaheuristic approaches, no systematic comparison of state-of-the-art algorithms for the QAP is available. Early comparisons include comparative tests of a simulated annealing and a tabu search algorithm [4], comparisons of a number of different early approaches to the QAP [37], and the comparative study of Taillard [60]. Regarding the most recent implementations, only partial comparisons are available [22, 39, 56, 57, 54]. Therefore, the research on the QAP in the context of the Metaheuristics Network has the potential to provide a systematic comparison of highly performing metaheuristics for the QAP.

5 Conclusions

In this article we have given an overview of existing algorithmic approaches to tackle the QAP. The best performing algorithms for the QAP are currently those based on tabu search and hybrid algorithms that typically combine a mechanism to generate initial solutions for a local search procedure with powerful local search engines. For the generation of starting solutions, typically genetic algorithms, ant colony optimization, or iterated local search are used. Crucial for the performance of these hybrid algorithms is to find a good balance between the degree of diversification obtained by the solution generation mechanism, the solution quality returned by the local search and its speed (the speed determines strongly the frequency of applying local search).

When comparing the computational results obtained with the different metaheuristics, a striking fact is that the relative performance of these metaheuristics depends very much on the particular instance class to which they are applied. The current state-of-the-art suggests that tabu search algorithms are particularly successful on instances with essentially no strong structure, while the iterative restart type metaheuristics like iterated local search, ant colony optimization, and genetic algorithms are particularly successful on more structured QAP instances.

Acknowledgments This work was supported by the “Metaheuristics Network”, a Research Training Network funded by the Improving Human Potential programme of the CEC, grant HPRN-CT-1999-00106. The information provided is the sole responsibility of the authors and does not reflect the Community’s opinion. The Community is not responsible for any use that might be made of data appearing in this publication.

References

[1] E. Angel and V. Zissimopoulos. Towards a classification of combinatorial optimization problems relatively to their difficulty for generalized local search algorithms. Submitted to *Discrete Applied Mathematics*, 1997.

[2] E. Angel and V. Zissimopoulos. Autocorrelation coefficient for the graph bipartitioning problem. *Theoretical Computer Science*, 191:229–243, 1998.

[3] R. Battiti and G. Tecchiolli. The reactive tabu search. *ORSA Journal on Computing*, 6(2):126–140, 1994.

[4] R. Battiti and G. Tecchiolli. Simulated annealing and tabu search in the long run: A comparison on QAP tasks. *Computer and Mathematics with Applications*, 28(6):1–8, 1994.

[5] J.L. Bentley. Fast algorithms for geometric traveling salesman problems. *ORSA Journal on Computing*, 4(4):387–411, 1992.

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

[7] E. Donald Brown, L. Christopher Huntley, and R. Andrew Spillance. A parallel genetic heuristic for the quadratic assignment problem. In *Proc. 3rd Conf. on Genetic Algorithms, 406-415*, 1989.

[8] R.E. Burkard and U. Finke. Probabilistic asymptotic properties of some combinatorial optimization problems. *Discrete Applied Mathematics*, 12:21–29, 1985.

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

[10] R.E. Burkard and F. Rendl. A thermodynamically motivated simulation procedure for combinatorial optimization problems. *European Journal of Operational Research*, 17:169–174, 1984.

[11] E. Çela. *The Quadratic Assignment Problem — Theory and Algorithms*, volume 1 of *Combinatorial Optimization*. Kluwer Academic Publishers, Dordrecht, NL, 1998.

[12] J. Chakrapani and J. Skorin-Kapov. Massively parallel tabu search for the quadratic assignment problem. *Annals of Operations Research*, 41:327–341, 1993.

[13] D.T. Connolly. An improved annealing scheme for the QAP. *European Journal of Operational Research*, 46:93–100, 1990.

[14] 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 the 1997 IEEE International Conference on Evolutionary Computation (ICEC'97)*, pages 165–170. IEEE Press, Piscataway, NJ, USA, 1997.

[15] J.W. Dickey and J.W. Hopkins. Campus building arrangement using TOPAZ. *Transportation Science*, 6:59–68, 1972.

[16] H.A. Eiselt and G. Laporte. A combinatorial optimization problem arising in dartboard design. *Journal of the Operational Research Society*, 42:113–118, 1991.

[17] A.N. Elshafei. Hospital layout as a quadratic assignment problem. *Operations Research Quarterly*, 28:167–179, 1977.

[18] G. Finke, R.E. Burkard, and F. Rendl. Quadratic assignment problems. *Annals of Discrete Mathematics*, 31:61–82, 1987.

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

[20] C. Fleurent and F. Glover. Improved constructive multistart strategies for the quadratic assignment problem using adaptive memory. *INFORMS Journal on Computing*, 11(2):198–204, 1999.

[21] J.C.B. Frenk, M. van Houweninge, and A.H.G. Rinnooy Kan. Asymptotic properties of the quadraticassignment problem. *Mathematics of Operations Research*, 10:100–116, 1985.

[22] L.M. Gambardella, È.D. Taillard, and M. Dorigo. Ant colonies for the quadratic assignment problem. *Journal of the Operational Research Society*, 50(2):167–176, 1999.

[23] A.M. Geoffrion and G.W. Graves. Scheduling parallel production lines with changeover costs: Practical applications of a quadratic assignment/LP approach. *Operations Research*, 24:595–610, 1976.

[24] P. Hansen and N. Mladenović. An introduction to variable neighborhood search. In S. Voss, S. Martello, I.H. Osman, and C. Roucairol, editors, *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, pages 433–458. Kluwer, Boston, 1999.

[25] D.S. Johnson, C.H. Papadimitriou, and M. Yannakakis. How easy is local search? *Journal of Computer System Science*, 37:79–100, 1988.

[26] T. Jones and S. Forrest. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In L.J. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms (ICGA'95)*, pages 184–192. Morgan Kaufmann Publishers, San Francisco, CA, USA, 1995.

[27] J.P. Kelly, M. Laguna, and F. Glover. A study of diversification strategies for the quadratic assignment problem. *Computers & Operations Research*, 21:885–893, 1994.

[28] T.C. Koopmans and M.J. Beckman. Assignment problems and the location of economic activities. *Econometrica*, 25:53–76, 1957.

[29] G. Laporte and H. Mercure. Balancing hydraulic turbine runners: A quadratic assignment problem. *European Journal of Operational Research*, 35:378–381, 1988.

[30] P.S. Laursen. Simulated annealing for the QAP – optimal tradeoff between simulation time and solution quality. *European Journal of Operational Research*, 69:238–243, 1993.

[31] E.L. Lawler. The quadratic assignment problem. *Management Science*, 9:586–599, 1963.

[32] Y. Li and P.M. Pardalos. Generating quadratic assignment test problems with known optimal permutations. *Computational Optimization and Applications*, 1:163–184, 1992.

[33] 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, Providence, Rhode Island, USA, 1994.

[34] V. Maniezzo. Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem. *INFORMS Journal on Computing*, 11(4):358–369, 1999.

[35] V. Maniezzo and A. Colorni. The Ant System applied to the quadratic assignment problem. *IEEE Transactions on Data and Knowledge Engineering*, 11(5):769–778, 1999.

[36] V. Maniezzo, M. Dorigo, and A. Colorni. The ant system applied to the quadratic assignment problem. Technical Report IRIDIA/94-28, Université Libre de Bruxelles, Belgium, 1994.

[37] 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*, 81:188–204, 1995.

[38] 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.

[39] P. Merz and B. Freisleben. Fitness landscape analysis and memetic algorithms for the quadratic assignment problem. *tec*, 4(4):337–352, 2000.

[40] H. Mühlenbein. Parallel genetic algorithms, population genetics and combinatorial optimization. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms (ICGA'89)*, pages 416–421. Morgan Kaufmann Publishers, Palo Alto, CA, USA, 1989.

[41] K.A. Murthy, P.M. Pardalos, and Y. Li. A local search algorithm for the quadratic assignment problem. *Informatica*, 3(4):524–538, 1992.

[42] V. Nissen. Solving the quadratic assignment problem with clues from nature. *IEEE Transactions on Neural Networks*, 5(1):66–72, 1994.

[43] V. Nissen and H. Paul. A modification of threshold accepting and its application to the quadratic assignment problem. *OR Spektrum*, 17:205–210, 1995.

[44] P.M. Pardalos, F. Rendl, and H. Wolkowicz. The quadratic assignment problem: A survey and recent developments. 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 1–42. American Mathematical Society, Providence, Rhode Island, USA, 1994.

[45] M. Queyranne. Performance ratio of polynomial heuristics for triangle inequality quadratic assignment problems. *Operations Research Letters*, 4:231–234, 1986.

[46] W.T. Rhee. A note on asymptotic properties of the quadratic assignment problem. *Operations Research Letters*, 7(4):197–200, 1988.

[47] W.T. Rhee. Stochastic analysis of the quadratic assignment problem. *Mathematics of Operations Research*, 16:223–239, 1991.

[48] S. Sahni and T. Gonzalez. P-complete approximation problems. *Journal of the ACM*, 23(3):555–565, 1976.

[49] A.A. Schäffer and M. Yannakakis. Simple local search problems that are hard to solve. *SIAM Journal on Computing*, 20:56–87, 1991.

[50] J. Skorin-Kapov. Tabu search applied to the quadratic assignment problem. *ORSA Journal on Computing*, 2:33–45, 1990.

[51] J. Skorin-Kapov. Extensions of tabu search adaptation to the quadratic assignment problem. *Computers & Operations Research*, 21(8):855–865, 1994.

[52] P.F. Stadler. Towards a theory of landscapes. Technical Report SFI–95–03–030, Santa Fe Institute, 1995.

[53] L. Steinberg. The backboard wiring problem: A placement algorithm. *SIAM Review*, 3:37–50, 1961.

[54] T. Stützle. Iterated local search for the quadratic assignment problem. Technical Report AIDA-99-03, FG Intellektik, FB Informatik, TU Darmstadt, Darmstadt, Germany, March 1999.

[55] T. Stützle. *Local Search Algorithms for Combinatorial Problems — Analysis, Improvements, and New Applications*, volume 220 of *DISKI*. Infix, St. Augustin, Germany, 1999.

[56] T. Stützle and M. Dorigo. ACO algorithms for the quadratic assignment problem. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 33–50. McGraw Hill, London, UK, 1999.

[57] T. Stützle and H.H. Hoos. $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System. *Future Generation Computer Systems*, 16(8):889–914, 2000.

[58] W. Szpankowski. Combinatorial optimization problems for which almost every algorithm is asymptotically optimal. *Optimization*, 33:359–367, 1995.

[59] É.D. Taillard. Robust taboo search for the quadratic assignment problem. *Parallel Computing*, 17:443–455, 1991.

[60] É.D. Taillard. Comparison of iterative searches for the quadratic assignment problem. *Location Science*, 3:87–105, 1995.

[61] É.D. Taillard and L.M. Gambardella. Adaptive memories for the quadratic assignment problem. Technical Report IDSIA-87-97, IDSIA, Lugano, Switzerland, 1997.

[62] D.M. Tate and A.E. Smith. A genetic approach to the quadratic assignment problem. *Computers & Operations Research*, 22(1):73–83, 1995.

[63] C. Voudouris. *Guided Local Search for Combinatorial Optimization Problems*. PhD thesis, Department of Computer Science, University of Essex, Colchester, UK, 1997.

[64] E.D. Weinberger. Correlated and uncorrelated fitness landscapes and how to tell the difference. *Biological Cybernetics*, 63:325–336, 1990.

[65] M.R. Wilhelm and T.L. Ward. Solving quadratic assignment problems by simulated annealing. *IIE Transactions*, 19(1):107–119, 1987.