Fitness Landscape Analysis and Memetic Algorithms for the Quadratic Assignment Problem

Peter Merz and Bernd Freisleben

***Abstract*—In this paper, a fitness landscape analysis for several instances of the quadratic assignment problem (QAP) is performed, and the results are used to classify problem instances according to their hardness for local search heuristics and meta-heuristics based on local search. The local properties of the fitness landscape are studied by performing an autocorrelation analysis, while the global structure is investigated by employing a fitness distance correlation analysis. It is shown that epistasis, as expressed by the dominance of the flow and distance matrices of a QAP instance, the landscape ruggedness in terms of the correlation length of a landscape, and the correlation between fitness and distance of local optima in the landscape together are useful for predicting the performance of memetic algorithms—evolutionary algorithms incorporating local search—to a certain extent. Thus, based on these properties, a favorable choice of recombination and/or mutation operators can be found. Experiments comparing three different evolutionary operators for a memetic algorithm are presented. It is shown in experiments that a memetic algorithm using our recently proposed information-preserving recombination operator for the QAP is able to outperform five of its competitors, two variants of tabu search, two ant-colony algorithms, and simulated annealing, on all tested instances of practical interest on a set of problem instances with a problem size of up to 256.**

***Index Terms*—Fitness landscapes, genetic local search, memetic algorithms, quadratic assignment problem, search space analysis.**

I. Introduction

THE quadratic assignment problem (QAP) has been introduced by Koopmans and Beckmann [1] to describe a location problem where a set of facilities has to be assigned to given locations at minimal cost. Mathematically, the QAP can be defined as follows.

Given two $n \times n$ matrices $\boldsymbol{A} = (a_{ij})$ and $\boldsymbol{B} = (b_{ij})$, the following cost function has to be minimized:

$$C(\pi) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} b_{\pi(i)\pi(j)} \tag{1}$$

where

- n denotes the number of facilities/locations,
- a_{ij} denotes the distance between location i and location j,
- b_{kl} denotes the flow of materials from facility k to facility l, and
- π denotes a permutation of the set $N = \{1, 2, \cdots, n\}$.

Manuscript received June 5, 1999; revised October 27, 1999.

The authors are with the Department of Electrical Engineering and Computer Science, University of Siegen, D-57068 Siegen, Germany (e-mail: pmerz, freisleb@informatik.uni-siegen.de).

Publisher Item Identifier S 1089-778X(00)04468-4.

The QAP arises in many practical applications, such as backboard wiring on electronic circuits [2] or the design of typewriter keyboards and control panels [3], [4]. Furthermore, it has been used in facility location problems, in particular, hospital planning [5], [6], and in finding locations for new buildings of a campus [7]. Besides other domains of engineering and design [8], a new application of the QAP in biology has recently been discovered in the context of *indirect gradient analysis* (reconstruction of the intensity of some latent environmental factors from species' responses) [9].

The QAP belongs to the class of $\mathcal{NP}$-hard problems [10]; thus, the exact solution of the problem is only practical for fairly small instances up to a size of 20–30 locations. Therefore, several heuristics have been proposed for finding near-optimum solutions to large QAP instances, including *ant colonies* [11]–[14] (a more general introduction to the ant system can be found in [15]–[17]), *evolution strategies* [18], *genetic algorithms* [19], *simulated annealing* [20], [21], *neural networks* [22], *memetic algorithms* [23]–[26] *tabu search* [27]–[31], *threshold accepting* [32], *randomized greedy search* (*GRASP*) [33], *scatter search* [34], and *tree search* [35].

In this paper, the fitness landscapes of several instances of the QAP are studied by performing an autocorrelation and fitness distance correlation analysis. The results of the landscape analysis are useful for designing a memetic algorithm (MA) [36], [37], i.e. an evolutionary algorithm incorporating a local search heuristic [38]–[40], for the QAP. 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 [41]–[45], it is quite natural to investigate whether a MA could also be beneficial for producing good solutions for the QAP. The paper presents memetic algorithms with various recombination and mutation operators. Experimental results with these operators demonstrate that our recently proposed recombination operator [46] appears to be preferable for instances with structured flow and distance matrices, and mutation is shown to be superior for instances with rather unstructured matrices A and B. Here, the correlation length of the landscape provides an upper bound for setting up the mutation jump distance for the mutation operator. A comparison with five alternative approaches to the QAP—two tabu search algorithms, ant colony systems and simulated annealing—is conducted to show the efficiency of the memetic algorithm. The MA proves to be superior on almost all instances tested, and is only outperformed slightly

by tabu search and ant colonies incorporating tabu search on a few instances that are not of practical interest. In comparison to our work presented in [46], this paper contains a) a detailed fitness distance analysis; b) a comparison of three evolutionary operators for our MA; c) an extended set of experiments on large problem instances up to $n = 256$; and d) a comparison study with two additional alternative approaches (simulated annealing and the Min–Max Ant System meta-heuristic incorporating local search and tabu search [14]).

The paper is organized as follows. In Section II, the QAP is discussed in more detail, and a local search for the QAP is described. In Section III, a fitness landscape analysis is performed for the QAP, and several types of instance of the QAP are discussed. Section IV presents the MA approach for solving combinatorial optimization problems in general terms and the evolutionary operators and MA components specially designed to solve the QAP. The results of the memetic algorithm with different evolutionary operators for selected instances and a comparison with five high quality algorithms for the QAP are provided in Section V. Section VI concludes the paper and outlines areas for future research.

II. The Quadratic Assignment Problem

In addition to its relevance in various fields including operations research and biology, the QAP serves as a generalization of some other important optimization problems. For example, the *graph partitioning problem* (GPP) and the *traveling salesman problem* (TSP) are known to be special cases of the QAP. The TSP can be formulated as a QAP by defining the matrix $\boldsymbol{A} = (a_{ij})$ and $\boldsymbol{B} = (b_{ij})$ as follows. Let

$$a_{ij} = \begin{cases} 1 & \text{if } j = (i+1 \bmod n) \\ 0 & \text{otherwise,} \end{cases} \qquad b_{ij} = d(i,j), \tag{2}$$

with $d(i,j)$ denoting the distance between city i and city j.

The graph bipartitioning problem, in which a partition of a graph $G = (V, E)$ into two equally sized sets with a minimum number of edges between the different sets is desired, can be expressed as a QAP by defining

$$a_{ij} = \begin{cases} 0 & \text{if } i,j \le \dfrac{n}{2} \vee i,j > \dfrac{n}{2} \\ 1 & \text{otherwise,} \end{cases}$$
$$b_{ij} = \begin{cases} 1 & \text{if } (i,j) \in E \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

A partition represented by a permutation π is defined as follows. A vertex j belongs to the first set, if $\pi(j) \le n/2$, and to the second set otherwise.

Hence, graph partitioning problems and traveling salesman problems can be solved by an algorithm developed for the QAP.

A. QAP Matrix Transformations

The transformation of instances of a given optimization problem into easier solvable problem instances is an interesting issue for designing heuristics. Thus, the goal is to find transformations that help in increasing the efficiency or the effectiveness of an algorithm. For the QAP, the following transformation may be used for developing promising heuristics.

Assuming that the matrix $\boldsymbol{B}$ of a QAP instance is symmetric; the matrix $\boldsymbol{A} = (a_{ij})$ can be transformed into $\boldsymbol{A}' = (a'_{ij})$ without changing the resulting cost if

$$a'_{ij} = \lambda_{ij}(a_{ij} + a_{ji}), \qquad \text{with } \lambda_{ij} = 1 - \lambda_{ji},\ \lambda_{ii} = \tfrac{1}{2}. \tag{4}$$

A proof can be found in [47].

Thus, setting λ_{ij} to 0.5 for all i, j, the transformation of an asymmetric matrix $\boldsymbol{A}$ leads to a symmetric matrix $\boldsymbol{A}'$ with

$$C(\pi) = \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{1}{2}(a_{ij} + a_{ji}) b_{\pi(i)\pi(j)} = \sum_{i=1}^{n}\sum_{j=1}^{n} a'_{ij} b_{\pi(i)\pi(j)}. \tag{5}$$

An analogous transformation can be utilized to obtain two symmetric matrices if matrix $\boldsymbol{A}$ is symmetric while $\boldsymbol{B}$ is not. We will see later that the above transformation does not change the fitness landscape of a problem instance, but by converting an asymmetric into a symmetric instance, the computation time of a local search procedure can be reduced considerably.

B. Representation of QAP Solutions

The encoding of QAP solutions used in our implementations is straightforward. We encode the permutation π as a vector of facilities, such that the value j of the ith component in the vector indicates that facility j is assigned to location i ($\pi(i) = j$). Individual A in Fig. 1 represents a solution where facility 2 is assigned to location 1, facility 4 is assigned to location 2, and so on.

C. A Local Search Procedure

Local search algorithms are simple, and often effective search algorithms for combinatorial optimization problems. In this paper, a local search (LS) is used for the fitness landscape analysis of the QAP, as well as for producing locally optimum solutions in our memetic algorithm described below.

The local search used in our algorithms is a variant of the *2-opt* heuristic, also known as the pairwise interchange heuristic [48]. The outline of the algorithm is shown in Fig. 2.

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. Fig. 1 illustrates such a 2-opt move. The number of swaps, and consequently the size of this neighborhood, grows quadratically with n. The change in the total cost $C(\pi) - C(\pi')$ by a swap of the elements i and j in the permutation π can be calculated in linear time.

Assuming that the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are symmetric, and that all of the diagonal elements of one of the matrices are zeros, the formula becomes

$$\Delta C(\pi, i, j) = 2 \sum_{k=1, k\neq i,j}^{n} (a_{jk} - a_{ik}) \left(b_{\pi(i)\pi(k)} - b_{\pi(j)\pi(k)}\right). \tag{6}$$

In the asymmetric case, the formula is more complicated. However, with the help of (5), asymmetric instances can be transformed easily into symmetric ones if one of the matrices is al-

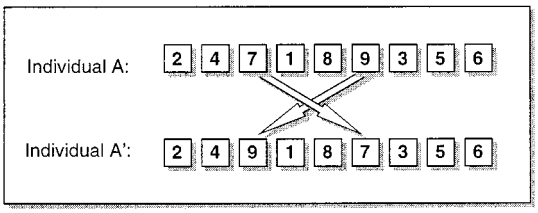

Fig. 1. A single step, called move, in the local search procedure. The two elements 7 and 9 are exchanged in the solution vector.

```
procedure Local-Search(π ∈ Π(n)): Π(n);
  begin
    repeat
      Generate neighboring solution π' ∈ N(π);
      if C(π') < C(π) then π := π';
    until ∀ π' ∈ N(π) : C(π') ≤ C(π);
    return π;
  end;
```

Fig. 2. A general template of a local search heuristic in pseudocode.

ready symmetric. In our algorithm, the transformation is performed once upon startup, and thus the computation time for the local search is reduced considerably (up to a factor of 4 in our implementation) due to the much faster evaluation of ΔC.

In contrast to the original *2-opt* heuristic, our *fast-2-opt* variant is based on performing the first swap found that reduces the total cost $C(\pi)$. Furthermore, we included another mechanism to speed up the local search: To reduce the time to search for an improving swap in the neighborhood of the current solution, a *don't look bit* for each location is maintained. The "don't look bit" technique has been proposed by Bentley [49] to speed up the 2-opt and 3-opt local search algorithm for the traveling salesman problem. However, it can also be used in a local search for the QAP. If the "don't look bit" for location i is set to one, the facility at location i will not be considered for an improvement swap in the current local search iteration. Initially, all "don't look bits" are set to zero. The "don't look bit" for location i will be set to one if no improving move could be found for the current solution with the facility at location i being one of the facilities to swap. If an improving swap is found in which the facility at location k has to be exchanged with the facility at location l, the "don't look bits" for location k and l are set to zero.

This technique reduces the running time without a significant loss in quality of the solutions. Our experiments have shown that similar to the *2-opt* for the TSP, if "don't look bits are used," it is not worth spending time to search for the best *2-opt* move, as is done, for example, in tabu search.

D. A Distance Measure

The fitness landscape analysis and the memetic algorithm described below rely on a distance metric for QAP solutions. 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:

$$d(\pi_1, \pi_2) = |\{i \in \{1, \cdots, n\} \,|\, \pi_1(i) \neq \pi_2(i)\}|. \quad (7)$$

III. Fitness Landscape Analysis

The concept of a *fitness landscape* [50], introduced to illustrate the dynamics of biological evolutionary optimization, has been proven to be very powerful in evolutionary theory. The concept furthermore has been shown to be useful for understanding the behavior of combinatorial optimization algorithms, and can help in predicting their performance. Viewing the search space, i.e., the set of all (candidate) solutions, as a landscape, a heuristic algorithm can be thought of as navigating through it in order to find the highest peak of the landscape; the height of a point in the search space reflects the fitness of the solution associated with that point.

More formally, a fitness landscape (S, f, d) of a problem instance for a given combinatorial optimization problem consists of a set of points (solutions) S, a fitness function $f\colon S \to \mathbb{R}$, which assigns a real-valued fitness to each of the points in S, and a distance measure d, which defines the spatial structure of the landscape. The fitness landscape can thus be interpreted as a graph $G_L = (V, E)$ with vertex set $V = S$ and edge set $E = \{(s, s') \in S \times S \,|\, d(s, s') = d_{\min}\}$, with $d_{\min}$ denoting the minimum distance between two points in the search space. The diameter $\operatorname{diam} G_L$ of the landscape is another important property: it is defined as the maximum distance between the points in the search space.

For binary-coded problems ($S = \{0,1\}^n$), the graph G_L is a hypercube of dimension n, and the distance measure is the hamming distance between bit strings. The minimum distance $d_{\min}$ is 1 (one bit with a different value), and the maximum distance is $\operatorname{diam} G_L = n$.

A. Properties of Fitness Landscapes

Several properties of fitness landscapes are known to have some influence on the performance of heuristic optimization algorithms. In this paper, we concentrate on the number of local optima (peaks) in the landscape, the distribution of the peaks in the search space, and the landscape ruggedness, i.e., the correlation between neighboring points in the search space.

Statistical methods have been proposed to measure landscape ruggedness and to analyze the distribution of the peaks, but the number of local optima cannot be determined in general. However, landscape ruggedness is tightly coupled to the number of local optima within the search space.

A fitness landscape is said to be rugged if the landscape consists of many peaks, and if there is low correlation between neighboring points. The autocorrelation functions proposed by Weinberger [51] measure the ruggedness of a fitness landscape.

Weinberger suggested to perform random walks to investigate the correlation structure of a landscape. The *random walk correlation function* [51]–[53]

$$r(s) = \frac{1}{\sigma^2(f)(m-s)} \sum_{t=1}^{m-s} (f(x_t) - \overline{f})(f(x_{t+s}) - \overline{f}) \quad (8)$$

of a time series $\{f(x_t)\}$ defines the correlation of two points s steps away along a random walk of length m through the fitness landscape ($\sigma^2(f)$ denotes the variance of the fitness values).

Based on this correlation function, the correlation length ℓ [53] of the landscape is defined as

$$\ell = -\frac{1}{\ln(|r(1)|)} \tag{9}$$

for $r(1), \neq 0$. The correlation length directly reflects the ruggedness of a landscape. The lower the value for ℓ, the more rugged the landscape.

If the landscape is *statistically isotropic*, i.e., the time series $\{f(x_t)\}$ forms a stationary random process, then a single random walk is sufficient to obtain $r(s)$. If a time series is *isotropic*, *Gaussian*, and *Markovian*, then the corresponding landscape is called an AR(1) landscape, and the random walk correlation function is of the form $r(s) = r(1)^s = e^{-s/\ell}$, with ℓ being the correlation length of the landscape. For example, AR(1) landscapes are found in the NK model and the TSP [51].

A ruggedness measure similar to the correlation length ℓ has been proposed in [54], called the *autocorrelation coefficient* λ, which has approximately the same value.

Kauffman has shown, for NK landscapes, that the number of local optima increases with the ruggedness of a landscape. Thus, the higher the correlation length, the smaller the number of local optima. Krakhofer and Stadler [55] have shown that, for random graph bipartitioning problems, there is one local optimum on the average in a ball of radius $R(\ell)$, where $R(s)$ denotes the average distance of two points s steps away on a random walk.

A further important measure is the *fitness distance correlation (FDC) coefficient*, proposed in [56] as a measure for problem difficulty for genetic algorithms. The FDC coefficient ϱ is defined as

$$\varrho(f, d_{\text{opt}}) = \frac{\text{cov}(f, d_{\text{opt}})}{\sigma(f)\sigma(d_{\text{opt}})} \tag{10}$$

and determines how closely the fitness and distance to the nearest optimum in the search space denoted by d_{opt} are related. If the fitness increases when the distance to the optimum becomes smaller, then search is expected to be easy for selection-based algorithms since there is a "path" to the optimum via solutions with increasing fitness. A value of $\varrho = -1.0$ ($\varrho = 1.0$) for a maximization (minimization) problem indicates that the fitness and distance to the optimum are perfectly related, and that search promises to be easy. A value of $\varrho = 1.0$ ($\varrho = -1.0$) means that, with increasing fitness, the distance to the optimum increases too. To gain insight into the global structure of the landscape, a fitness distance analysis (FDA) can be performed for locally optimum solutions for a given problem instance. Thus, it can be determined whether there is a structure in the distribution of locally optimum solutions which can be exploited by a meta-heuristic based on local search. The local optima may be contained in a small fraction of the search space or there may be a correlation between the fitness of the local optima with their distance to an optimum solution.

Fitness distance plots are well suited to visualize the results obtained from FDA. Several researchers have used FDA to analyze fitness landscapes, including Kauffman [57] for NK-landscapes, Boese [58] for the TSP, and Reeves for a flow-shop scheduling problem [59], and Merz and Freisleben for the graph bipartitioning problem [60].

Additionally, when performing FDA, it is useful to calculate other properties, such as the number of distinct local optima found, and the average distance between the local optima.

B. The Fitness Landscape of the QAP

Although there may be alternative landscapes for the QAP, we restrict our search space analysis to the landscape $\mathcal{L} = (S, f, d)$ with $S = \Pi(n)$ (the set of all permutations of $\{1, \cdots, n\}$), $f = c(\pi)$ as defined in (1), and d as defined in (7), since the local search is designed for that landscape. The diameter $\text{diam}\, G_{\mathcal{L}}$ of the landscape is equal to the problem size n. The minimum distance between two points in the search space is $d_{\min} = 2$.

Note that the transformation provided in (4) does not change the structure of the fitness landscape since the cost difference ΔC of neighboring solutions is the same for transformed and original instances.

In the following, the types of QAP instances used in our analysis are described.

1) Types of QAP Instances: QAPLIB [61] is a publicly accessible library of instances of the QAP. It contains different types of QAP instances; some of them were drawn from real-world applications, while others were generated randomly to assess the performance of heuristics. In order to find a complexity measure for QAP instances, Vollmann and Buffa [62] have introduced the *flow dominance*, which measures to what extent the flow matrix $\boldsymbol{B}$ shows "dominant" flow patterns. The dominance $\text{dom}(\boldsymbol{X})$ for a matrix $\boldsymbol{X}$ is defined as the coefficient of variation multiplied by 100:

$$\text{dom}(\boldsymbol{X}) = 100 \cdot \frac{\sigma(\boldsymbol{X})}{\overline{\boldsymbol{X}}}. \tag{11}$$

In the above formula, $\overline{\boldsymbol{X}}$ denotes the average value of the elements x_{ij} of the matrix $\boldsymbol{X}$, and $\sigma(\boldsymbol{X})$ denotes the standard deviation of the elements of matrix $\boldsymbol{X}$. Thus the flow dominance is denoted $\text{dom}(\boldsymbol{B})$. 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. However, the major drawback of this complexity measure is that it does not take the influence of the distance matrix $\boldsymbol{A}$ into account. Analogously, let $\text{dom}(\boldsymbol{A})$ be the distance dominance, then the dominance of a QAP instance can be defined according to [63], [64] as a vector:

$$\begin{aligned}\text{dom}(\boldsymbol{A}, \boldsymbol{B}) = (&\min\{\text{dom}(\boldsymbol{A}), \text{dom}(\boldsymbol{B})\}, \\ &\max\{\text{dom}(\boldsymbol{A}), \text{dom}(\boldsymbol{B})\}).\end{aligned} \tag{12}$$

QAP instances with randomly generated flows (distances) using a uniform distribution typically have a low flow (distance) dominance, whereas real-life instances and (nonuniformly) randomly generated instances similar to real-life instances have considerably higher dominance values for at least one of the matrices.

2) Epistasis and the QAP: The amount of gene interaction (epistasis) is known to have a strong influence on the performance of heuristics. If the solutions of a problem can be encoded in a bit string (binary vector), and the fitness can be decomposed into fitness contribution functions for each site, epistasis can be estimated easily. Consider the NK landscapes proposed by

TABLE I
AVERAGE DISTANCES AND FITNESS DISTANCE COEFFICIENTS FOR LOCAL MINIMA

Instance	n	$dom(\mathbf{A})$	$dom(\mathbf{B})$	$\overline{d}_{opt}$	$\overline{\Delta c}$	q	ϱ	$\overline{d}_{ls}$	i_{ls}	n/ℓ	ℓ
bur26a	26	15.1	274.9	23.3	18841.0	0.35	0.07	23.4	66.2	3.01	8.64
tai80a	80	59.2	60.4	78.9	547619.7	4.04	0.02	78.9	139.8	4.04	19.60
tai100a	100	59.3	60.3	97.7	776234.7	3.67	0.06	99.0	183.9	4.05	24.71
sko100a	100	50.8	106.6	97.6	3035.6	2.00	0.08	97.9	342.8	3.66	27.29
wil100	100	50.8	64.5	97.8	2909.4	1.07	0.03	97.6	343.2	3.59	27.82
tho150	150	51.5	147.2	147.9	185767.2	2.28	0.04	147.8	632.8	3.68	40.72
tai80b	80	64.0	323.2	77.3	44117586.0	5.39	0.22	77.7	366.9	3.41	23.48
tai100b	100	80.4	321.3	95.2	53694038.8	4.53	0.62	96.7	490.1	2.86	35.01
tai150b	150	51.8	314.1	148.1	15278698.1	3.06	0.05	148.1	839.4	3.75	39.95
tai256c	256	259.7	217.9	254.6	231150.2	0.52	0.05	255.0	193.5	4.04	63.29
lipa90a	90	18.4	41.9	88.8	2834.3	0.79	0.03	89.0	159.0	4.09	22.02
lipa90b	90	60.0	41.9	88.5	2758634.9	22.09	0.37	88.9	161.8	4.05	22.24
G124-02	124	100.0	711.3	123.0	20.7	79.74	0.00	123.0	42.9	4.00	30.97
G124-16	124	100.0	224.7	122.8	47.9	5.33	0.00	123.0	84.6	4.05	30.58
kro124p	100	995.0	49.6	99.0	25043.0	69.12	0.07	99.0	292.9	4.09	24.47
kroA100	100	995.0	54.8	99.1	20840.0	97.92	0.07	99.0	358.3	4.08	24.52

Kauffman [57]. The fitness f of a genome $\vec{x} = (x_1, \cdots, x_N) \in \{0,1\}^N$ is defined as follows:

$$f(\vec{x}) = \frac{1}{N} \sum_{i=1}^{N} f_i(x_i, x_{i_1}, \cdots, x_{i_K}). \tag{13}$$

Thus, the fitness contribution f_i for each site i depends on the gene value x_i of gene i and on the values of K other genes at the sites $i_1, \cdots, i_K$. Hence, the higher the value of K, the higher the epistasis. We have shown in [60] that there are problems for which the quantity denoting the (average) number of interacting genes does not sufficiently reflect the characteristics of the fitness landscape. To gain insight into the role of gene interactions, we introduced the notion of a dependency graph reflecting gene interaction. The vertices in the dependency graph represent the genes. An edge in the dependency graph from vertex i to vertex j indicates that the fitness contribution f_i of gene i depends on the value of x_j. Thus, the fitness contribution of gene i is of the form $f_i(x_i, \cdots, x_j, \cdots)$. For NK landscapes, the vertex degree of the dependency graph is $K+1$, including the edges going from the vertices to themselves.

Since the commonly used representation of the QAP is a permutation of the set $\{1, \cdots, N\}$, it is not obvious what gene interaction is in the QAP. The cost function $c(\pi)$ [see (1)] can be decomposed in the following way:

$$c(\pi) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} b_{\pi(i)\pi(j)} = \sum_{i=1}^{n} c_i(\pi). \tag{14}$$

Assuming that the matrix $\boldsymbol{B}$ does not contain zeros in the off-diagonal entries, a fitness (cost) contribution function c_i depends on those sites j for which $a_{ij} \neq 0$. In other words, the dependency graph is described by the matrix $\boldsymbol{A}$, where $a_{ij} \neq 0$ defines the weight of the edge (i,j). If matrix $\boldsymbol{B}$ has more elements in the off-diagonal entries equal to zero, matrix $\boldsymbol{B}$ should be considered instead of matrix $\boldsymbol{A}$. Note that exchanging the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ does not change the fitness landscape of a problem instance.

Thus, the dominance measure defined above is a good indicator for the amount of epistasis in a problem instance. For example, assuming that matrix $\boldsymbol{B}$ ($\boldsymbol{A}$) has no or only few zero entries, $\mathrm{dom}(\boldsymbol{A})$ [$\mathrm{dom}(\boldsymbol{B})$] is low and epistasis is high; if matrix $\boldsymbol{A}$ ($\boldsymbol{B}$) has many zero entries, the dominance is high and epistasis is low.

3) Fitness Distance Correlation Analysis: To analyze the fitness landscape of the QAP, we have selected 12 instances from QAPLIB, including real-world instances, randomly generated instances, and instances with known optimum solutions generated by an algorithm proposed in [65]. Furthermore, we have included two transformed instances of the graph-bipartitioning problem and two traveling-salesman problem instances transformed into QAP instances in our analysis. The chosen instances are displayed in Table I. Viewing the table from left to right, the name of the instance, the problem size n, and the dominance of matrix $\boldsymbol{A}$ and $\boldsymbol{B}$ ($\mathrm{dom}(\boldsymbol{A})$ and $\mathrm{dom}(\boldsymbol{B})$, respectively) are provided. The first ten instances (`bur26a`–`tai256c`) are either real-world or randomly generated QAP instances; the `lipa90a/b` QAP instances are randomly generated in a way that optimum solutions can be calculated in polynomial time. The two instances beginning with `G124` are transformed graph-bipartitioning instances. The transformed TSP instances are `kroA124p` and `kroA100`, an asymmetric and a symmetric TSP instance, respectively.

To investigate the distribution of local optima in the search space, we produced 10 000 local optima using the *fast 2-opt* local search described above. Additionally, we performed random walks to estimate the correlation length ℓ of the landscape. In Table I, the average distance to the optimum or best known solution $\overline{d}_{\mathrm{opt}}$, the cost difference $\Delta c = c - c_{\mathrm{opt}}$, the average quality $q = 100 \cdot (c_{\mathrm{opt}}/c - 1)$ of the local optima, the fitness distance correlation coefficient ϱ, and the mean distance between the local optima $\overline{d}_{ls}$ are provided. The last three columns contain the average number of iterations per local search i_{ls}, the correlation length in relation to the diameter of the landscape n/ℓ, and the correlation length ℓ itself. In Fig. 3, the scatter plots (FDC plots) are provided for representatives of the studied instances.

For all studied instances, the distances between the local optima and best known solutions $\overline{d}_{\mathrm{opt}}$, as well as the average distances between the local optima, are very close to the diameter

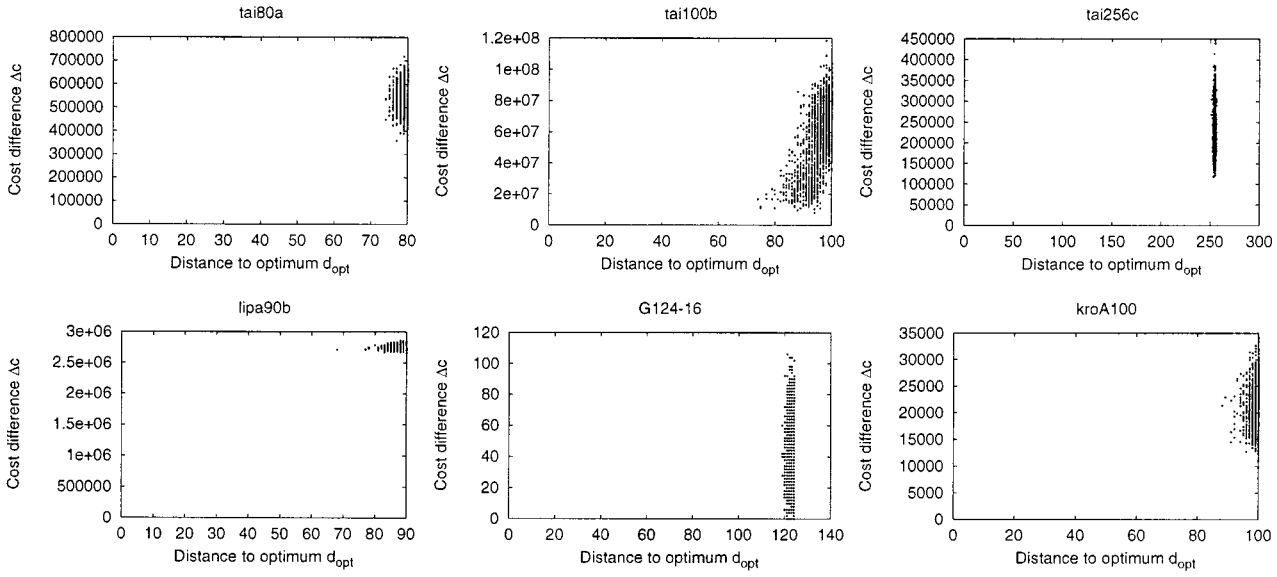

Fig. 3. FDC analysis of 10 000 local optima of selected instances from QAPLIB.

of the landscape. Thus, as can also be seen in the plots, the QAP instances have very unstructured landscapes. The local optima are neither restricted to a small region of the search space, nor do they seem to be correlated. An exception to this rule is instance `tai100b`, which exhibits a relatively high correlation of fitness and distance to the optimum. Surprisingly, the results cannot be predicted by looking at the dominances of the instances. For example, the TSP instances and the GBP instances have a high value for $\mathrm{dom}(\boldsymbol{A})$ and $\mathrm{dom}(\boldsymbol{B})$, respectively, but do not show correlations. Furthermore, instances `tai80b` and `tai100b` have similar flow and distance dominance values, but the former has significantly less correlated local optima. According to the distribution of the local optima, the instances can be divided into two types: the first type has correlated local optima and consists of the `tai`$*$`b` problems. The second type has no exploitable structure in the distribution of the local optima and consists of the remaining instances. Not surprisingly, the `tai`$*$`a` problems have uncorrelated rugged landscape since they are generated randomly with a uniform distribution. Thus, both matrices exhibit no exploitable structure. These instances are of no practical relevance, as noted in [30].

4) (Auto)correlation Analysis: Looking at the local structure of the landscapes, the instances can again be divided into two classes. Several instances have a highly rugged fitness landscape, as reflected by a low correlation length $(n/\ell \sim 4)$. The instances `bur26a`, `sko100a`, `wil100`, `tho150`, and `tai`$*$`b` have a higher relative correlation length as expressed by a lower n/ℓ. The smoothest landscape has instance `tai100b` with $n/\ell = 2.86$. Some of the rugged landscapes have another interesting property: For the instances `tai`$*$`a` and `tai256c`, the average number of improvements made by the 2-opt local search is very low compared to the other instances.

5) The Role of Epistasis and Structured Landscapes: Motivated by the fact that the results showed uncorrelated landscapes even for the TSP instances with low epistasis, we were interested to find out whether there are QAP landscapes that show a high correlation and have a correlation length close to $n/2$ and thus n/ℓ close to 2. We therefore created our own instances with varying epistasis. The two sets of problems consisting of 5 instances with $n = 100$ have been generated, as follows. The distance matrix $\boldsymbol{A}$ of each instance is constructed by creating n points randomly in the unit square. An entry a_{ij} is defined by the Euclidean distance $D(i,j)$ between point i and j multiplied by 100. Thus, the entries of $\boldsymbol{A}$ lie between 0 and 100. The matrix $\boldsymbol{B}$ is constructed in a similar fashion by creating n points randomly in another unit square. For the first set of instances, b_{ij} is set to the Euclidean distance $D(i,j)$ multiplied by 100 between point i and j if the distance is below or equal to a predefined maximum distance $D_{\max}$, zero otherwise. For the second set, b_{ij} is set to $100/(D(i,j)+1)$, if $D(i,j)$ is below or equal to a predefined maximum distance $D_{\max}$. Thus, by varying the threshold distance $D_{\max}$, we can create instances with arbitrary epistasis/flow dominance. The instances are structured since, for the elements in the matrices, the triangle inequality is obeyed. In real-world applications at least one of the matrices represent a form of distance, which naturally fulfills the triangle inequality. Thus, the generated instances are aimed to provide a structure that is found in real-world applications. Table II provides an overview of threshold distances used. We performed the same experiments on the newly created problems as for the previously described instances. The results are displayed in Table III and Fig. 4. To obtain the best-known solutions for the new instances, several long runs (2 h) with our MA using CX recombination as described below have been performed.

TABLE II
THRESHOLD PARAMETERS FOR THE GENERATED QAP INSTANCES

D_{max}	1	0.75	0.5	0.25	0.1
Set 1	`pmd100a`	`pmd100b`	`pmd100c`	`pmd100d`	`pmd100e`
Set 2	`pmr100a`	`pmr100b`	`pmr100c`	`pmr100d`	`pmr100e`

The `pmd`$*$ problems have a $\mathrm{dom}(\boldsymbol{A})$ value of approximately 50, while the flow dominances $\mathrm{dom}(\boldsymbol{B})$ are between 50 and

TABLE III
AVERAGE DISTANCES AND FITNESS DISTANCE COEFFICIENTS FOR LOCAL MINIMA

Instance	n	$dom(\mathbf{A})$	$dom(\mathbf{B})$	$\overline{d}_{opt}$	$\overline{\Delta c}$	q	ϱ	$\overline{d}_{ls}$	i_{ls}	n/ℓ	ℓ
pmd100a	100	48.2	50.9	93.9	49337.6	0.20	0.23	94.8	416.9	2.99	33.46
pmd100b	100	49.2	68.1	96.5	83625.5	0.54	0.26	97.0	450.5	2.97	33.67
pmd100c	100	48.9	119.9	93.4	115679.7	2.32	0.46	95.1	470.2	2.55	39.25
pmd100d	100	49.5	252.0	96.7	81714.1	17.80	0.29	97.6	446.9	3.10	32.30
pmd100e	100	49.3	686.7	98.0	5444.5	61.66	0.09	98.5	323.9	3.65	27.40
pmr100a	100	48.8	24.5	94.9	122075.6	0.40	0.24	95.5	427.9	2.42	41.27
pmr100b	100	48.4	52.4	95.1	192941.5	0.84	0.25	95.6	431.5	2.36	42.38
pmr100c	100	49.3	100.3	94.4	442360.6	3.62	0.42	95.8	447.9	2.50	39.95
pmr100d	100	48.8	237.9	97.3	497712.5	22.23	0.16	98.0	446.0	3.39	29.50
pmr100e	100	48.6	651.9	97.8	85554.9	67.60	0.15	98.5	341.6	3.73	26.82

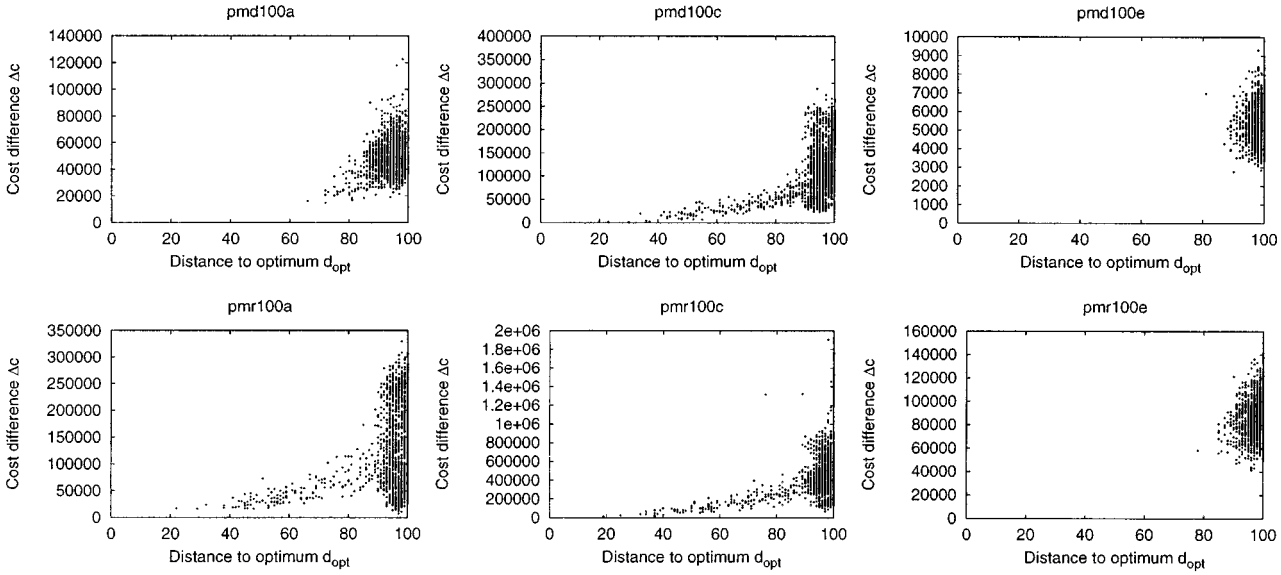

Fig. 4. FDC analysis of 10 000 local optima of selected instances from sets 1 and 2.

690. The pmr$*$ instances also have a distance dominance of 50, but the flow dominance varies from 24 to 650. Again, the average distance to the best known solution and the average distances between the local minima are close to the diameter of the landscape. The fitness distance correlation coefficient is significantly higher compared to the FDC of the other instances. However, all instances have a lower FDC than tai100b. pmd100c has the highest FDC ϱ in the first set, and pmr100c has the highest FDC in the second. The plots in Fig. 4 show that there exists a structure in the distribution of the local optima of the generated instances, except for the ones with high flow dominance. Furthermore, it can be seen for pmd100c and pmr100c that there are local optima which are much closer to the best known solution than most of the others. Even instance tai100b has no local optima that are similarly close to the best known solution.

The correlation length ℓ in the first set is highest for pmd100c, and for the second set, pmr100b has the highest correlation length. The values for n/ℓ are closer to 2 than the value for tai100b. Neither the correlation length nor the FDC nor $\overline{\Delta c}$ reflects the effectiveness of the local search to approach the optimum cost: the quality q of the locally optimum solutions decreases rapidly with decreasing epistasis [increasing $\mathrm{dom}(B)$]. For example, for pmd100a, the locally optimum solutions have a cost 0.20% above the optimum on average, while for pmd100e, the percentage excess is 61.71%! Thus, it seems that reaching the optimum cost is easy for meta-heuristics based on LS if the instances are structured and there is high epistasis (low flow dominance). For instances with low epistasis (high flow dominance), reaching the optimum cost seems to be harder.

IV. MEMETIC ALGORITHMS

Evolutionary algorithms—as a general term for *evolutionary programming* [66], *evolution strategies* [67], *genetic algorithms* [68], and *genetic programming* [69]—have been applied successfully in various domains of search, optimization, and artificial intelligence. In the field of combinatorial optimization, it has been shown that augmenting evolutionary algorithms with problem-specific heuristics can lead to highly effective approaches. These hybrid evolutionary algorithms combine the advantages of efficient heuristics incorporating domain knowledge and population-based search approaches. One form of hybridization is the use of local search in evolutionary algorithms [38], [70], [39], [40]. These algorithms, sometimes called genetic local search algorithms, belong to the class of memetic algorithms.

```
procedure MA;
  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;
      end-for;
      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);
      if P converged then
        foreach individual i ∈ P\{best} do i := Local-Search(Mutate(i));
      endif
    until terminate=true;
  end;
```

Fig. 5. General outline of a memetic algorithm in pseudocode.

MAs [36], [37] are population-based heuristic search approaches for optimization problems similar to genetic algorithms (GAs). GAs, however, rely on the concept of biological evolution, but MAs, in contrast, mimic cultural evolution. While, in nature, genes are usually not modified during an individual's lifetime, *memes* [71] are. They can be thought of as units of information that are replicated while people exchange ideas. A meme is usually modified by the person before he or she passes it on to the next generation. As with genes, memes also evolve over time; good ideas may survive, while bad ones may not.

In the context of evolutionary computation, a hybrid evolutionary algorithm is called memetic if the individuals representing solutions to a given problem are improved by a local search, simulated annealing, or another improvement technique. Thus, in a memetic algorithm, a population consists solely of locally optimum solutions. Consequently, in comparison to other (hybrid) evolutionary algorithms, the role of the evolutionary variation operators recombination and mutation changes. Therefore, recombination in MAs is not restricted to mimicking a crossing over of two parent chromosomes aimed at combining building blocks. Even recombination operators that introduce a high amount of new alleles are often used in MAs that have no biological analogy, but mimic obsequious/rebellious behavior found in cultural systems [72]. Such operators are of limited applicability in cases where no additional local search is applied. Fig. 5 provides a general template for a memetic algorithm. The algorithm as described by the template has been proven to be very effective for several combinatorial optimization problems, such as the traveling-salesman problem (TSP) [43], the graph-bipartitioning problem (GBP) [44], and *NK* landscapes [45].

Upon startup, the algorithm creates the initial population by utilizing a randomized heuristic or generating starting solutions randomly without incorporating domain knowledge. Afterwards, all solutions are improved by the local search procedure to obtain locally optimum solutions. In the main loop of the algorithm, the genetic operators are applied to randomly selected members of the population a predefined number of times. Since the genetic operators are generally not able to produce locally optimum solutions, the local search algorithm is applied to the produced offspring before they are added to the population. A new generation is formed by selecting the best individuals from the extended population for survival.

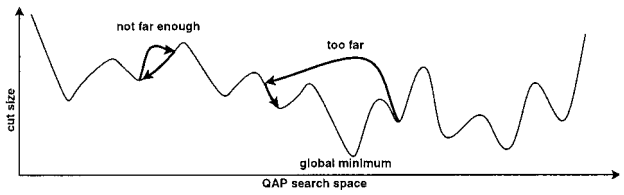

Fig. 6. Genetic operators performing jumps in the search space.

A unique feature of our memetic algorithm is the restart mechanism borrowed from Eshelman's CHC algorithm [73]. If the search has converged, e.g., the chances for generating new solutions with better fitness have approached zero, the population is diversified by mutating all but the best individual.

A. The Role of the Evolutionary Variation Operators

The recombination and mutation operators are aimed to explore the search space by "jumping" to new regions which are in turn efficiently searched by the local search procedure, as illustrated in Fig. 6.

Mutation operators allow jumping from one point in the search space to another in a random direction. Recombination operators, on the other hand, allow directed jumps. If the region between two or more points with high fitness in the search space contains solutions with high fitness more likely than other regions near the two points, a recombination operator can be designed to exploit this property. Thus, is distinction to other EAs, the variation operators are not intended to preserve building blocks in the solutions of a problem. In MAs, they are

responsible for finding new starting points for a neighborhood search aimed to result in local optima with higher fitness.

B. The Memetic Algorithm for the QAP

Although the memetic algorithm template is general in the sense that it can be used for every combinatorial optimization problem, some components are problem-specific. The creation of the initial population, as well as the local search and the genetic operators, is specific to the QAP, and will thus be described in the following. Furthermore, the selection mechanisms and the restart technique used in our algorithm are discussed.

1) Creating the Initial Population: To create the initial population of the MA, solutions are generated randomly without using any heuristic information, since a) effective constructive heuristics for the QAP have a high complexity, and b) random starting solutions in combination with local search produce relatively good solutions. Before the solutions are added to the population, the local search procedure is applied.

2) The DPX Recombination Operator: A distance measure between solutions is useful for explaining the behavior of evolutionary operators, since it defines the distance of the jumps made in the search space, and in some cases, the relative direction of a jump, if reference points exist. The evolutionary operators described in this section rely on the distance measure defined in (7).

The recombination operator DPX (*distance preserving crossover*) previously introduced in [74] relies on the notion of a distance between solutions. The basic idea behind DPX has also shown to be effective for the traveling salesman problem [41], [42], and can be described as follows. In general, the DPX is aimed at producing an offspring that 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 operator introduces a high amount of implicit mutations, the local search procedure applied subsequently 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 in some search spaces.

The DPX operator for the QAP works as follows. Suppose that two parents A and B as shown in Fig. 7 are given.

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 Fig. 7, the distance d is 5. Since it is computationally expensive to ensure that the child has exactly the distance d to both parents, we use a linear-time algorithm that produces offspring that have high probability d units away from their parents. Experiments have shown that there is no significant performance difference between the exact and approximate version of DPX in terms of solution quality. Therefore, we prefer the much simpler approximate algorithm.

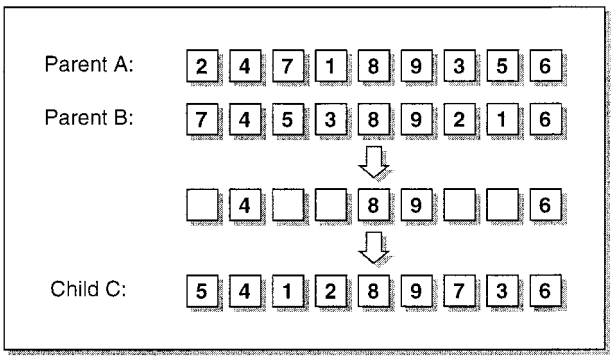

Fig. 7. DPX recombination operator for the QAP.

3) The CX Recombination Operator: The second recombination operator proposed recently for the QAP [46] 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. Although the operator is called UX in [46], we refer to it in this paper as CX, accounting for the fact that a similar operator has been proposed for the TSP called cycle crossover [75].

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.

In the first phase, all facilities found at the same locations in the two parents are assigned to the corresponding locations in the offspring. Then, starting with a randomly chosen location with no assignment, 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 unassigned 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 Fig. 8. First, all facilities that are assigned to the same location in the parents are inherited by the offspring. These are facilities 4, 9, and 6. Then, beginning with a randomly chosen location (in this case, location/position 3), a facility is selected randomly 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 of assigning 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 after the DPX or CX recombination operator has been applied, the genes with the same alleles contained in both parents are fixed. Hence,

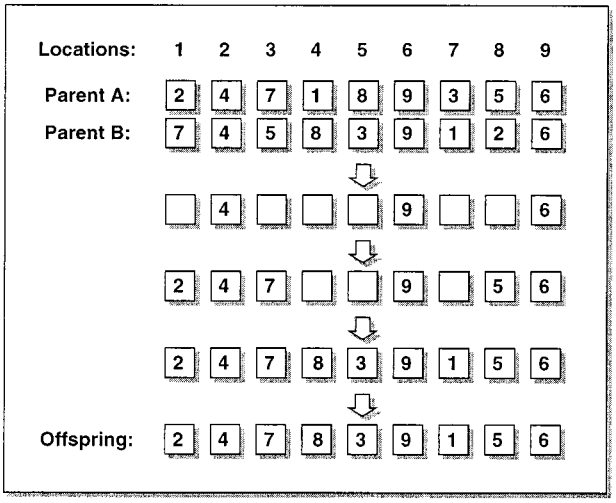

Fig. 8. CX recombination operator for the QAP.

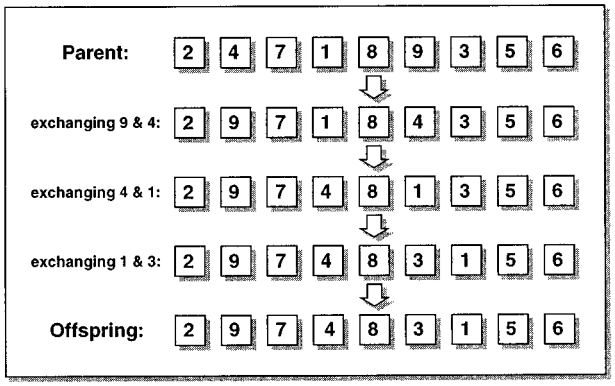

Fig. 9. Mutation operator for the QAP.

in the above example, swaps can only be performed among 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 $|\mathcal{N}_{2\text{opt}}| = (1/2)n \cdot (n-1)$ to $|\mathcal{N}| = (1/2)d \cdot (d-1)$ [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 decrease during the evolution.

4) The Mutation Operator: The mutation operator used in the MA approach exchanges a sequence of facilities in the solution until the offspring has a predefined distance to its parent, as shown in Fig. 9. To ensure that the offspring has the predefined distance, in each step, the second facility chosen is exchanged again in the succeeding step, such that the resulting distance between parent and offspring is one plus the number of exchanges. To illustrate the operation of the mutation operator, consider the example shown in Fig. 9.

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 (mutation jump) distance between parent and offspring is 4.

5) Selection and Diversification: Selection occurs two times in the main loop of the MA. *Selection for reproduction* is performed before a genetic operator can be applied, and *selection for survival* is performed after the offspring of a new generation have been created to reduce the population to its original size.

Selection for reproduction is performed on a purely random basis without bias to fitter individuals, while selection for survival is achieved by choosing the best individuals from the pool of parents and children. Thus, replacement in our algorithm is similar to the selection in the $(\mu + \lambda)$-ES (*evolution strategy*) [76]. Additionally, duplicates will be replaced by other solutions, so that each phenotype exists only once in the new population.

The population size of a MA is typically small compared to genetic algorithms: a size of 10 up to 40 is common in a MA, since the computational complexity of the local search does not allow evolving much larger populations in reasonable time. Such a small population size leads to a premature convergence of the algorithm, especially in the absence of mutation. To overcome this drawback, the restart technique proposed by Eshelman [73] is employed. During the run, it is checked whether the average distance of the population has dropped below a threshold $d = 10$, or the average fitness of the population did not change for more than 30 generations. If one of these conditions hold, the search is assumed to be converged and the whole population is mutated except the best individual, using the mutation operator described above with a high mutation jump distance. After mutation, each individual is improved by the local search algorithm to 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 solutions contained in the population 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 population so that the points are again far away from each other. Thus, the restart technique represents an escape mechanism from suboptimal regions of the search space.

6) Alternative Approaches: There are some other memetic algorithms, i.e., evolutionary approaches incorporating local search, that are based on recombination. These will be briefly described in the following.

In [26], a parallel genetic algorithm is presented that incorporates 2-opt local search. Recombination is performed by a voting mechanism: Each child has p parents. The number of parents in which a facility is assigned to the same location is counted. If the facility is assigned to a location more often than a predefined threshold, the facility is assigned to that location in the child. All other assignments are made at random. The DPX operator is similar to the voting recombination with $p = 2$ and a threshold of 1. However, DPX guarantees a distance to the parents, while voting recombination does not. Voting recombination is a highly disruptive recombination with a high degree of implicit mutations.

The SAGA algorithm proposed in [23] incorporates simulated annealing instead of a simple 2-opt local search. Recombination is performed similarly to the PMX crossover proposed in [77] for the TSP. A sequence of assignments between two randomly chosen crossover points is copied from the first parent to the offspring. Additional assignments are made that are found in the second parent, while maintaining feasibility. The remaining

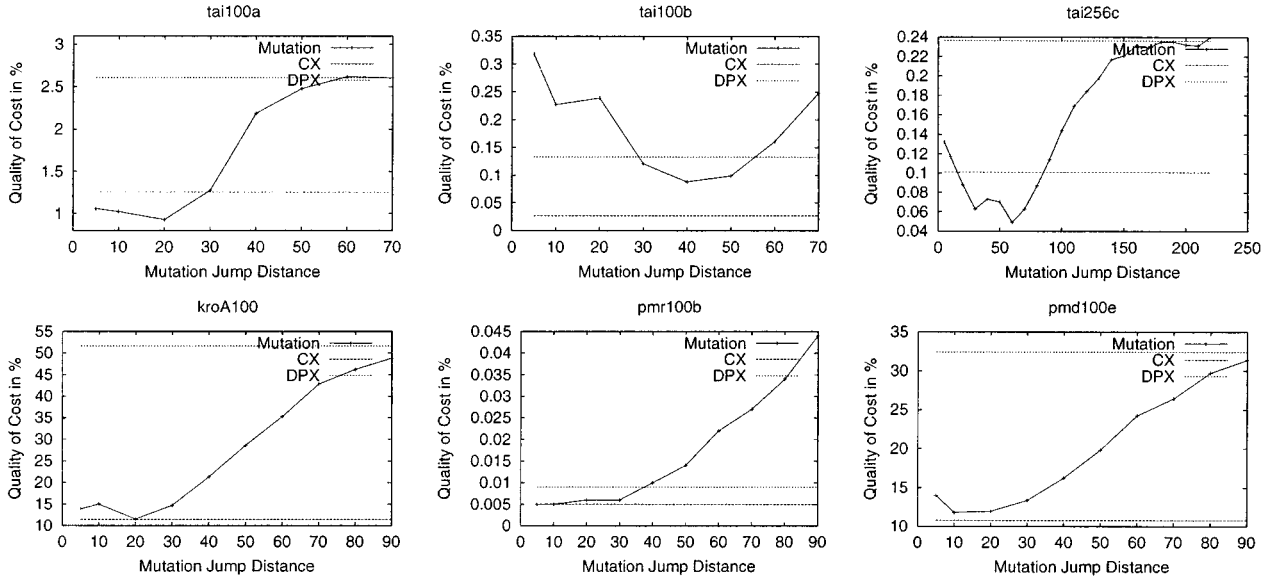

Fig. 10. Comparison of evolutionary operator performance for the instances `tai100a`, `tai100b`, and `tai256c`.

unassigned facilities are randomly allocated. Hence, the operator also performs implicit mutations.

The memetic algorithm described in [24] uses the crossover operator as in SAGA. But here, a tabu search is used instead of simulated annealing.

The genetic hybrids introduced in [25] are genetic algorithms incorporating local search or tabu search. The recombination operator used in these algorithms is borrowed from [19], and works as follows. First, all facilities assigned to the same location in both parents are copied to this location in the child. Second, the unassigned locations are scanned from left to right. For the unassigned locations, a facility is chosen at random from those occupying the location in the parents if they are not yet included in the child. Third, remaining facilities are assigned at random. Thus, implicit mutation occurs in the last step of the recombination scheme.

Our memetic algorithm incorporating CX differs from the above approaches since it introduces no implicit mutations during recombination. All described approaches use different selection strategies, and none of them incorporates the restart mechanism used in our approach.

V. Computational Experiments

We have performed several experiments with the memetic algorithm described above. In the first series of experiments, we investigated the performance of the MA with different genetic operators on the various types of landscapes described in Section III.B1). In the second series, a comparison was made with five other heuristics for the QAP. Our MA was implemented in C++. All experiments described in this paper were performed on Pentium II PC's (300 MHz) running Solaris 2.6 or Linux. All runs were performed with a population size $P = 40$, and the number of recombinations per generation was set to $0.5 \cdot P$. Restarts were performed by mutating R% of the locations in the genome (jump distance: $R \cdot n$); no additional mutations were performed between the restarts for the recombination-based MAs.

A. Memetic Algorithm Performance

In our first experiments, we tested the crossover operators DPX and CX described above, as well as the mutation operator with several jump distances. Runs were performed on the instances `tai100a`, `tai100b`, `tai150b`, `tho150`, `tai256c`, `kroA100`, and all `pmd`$*$ and `pmr`$*$ instances. Results are displayed for selected instances in Fig. 10. On the y axis, the solution quality, i.e., the percentage excess over the best known solution averaged over 30 runs, is shown. For all instances, the running time was set to 300 s, except for instance `tai256c`, for which the running time was set to 3600 s.

The experiments show that, for all instances except `tai100a` and `tai256c`, the CX operator is more effective than the mutation operator. Furthermore, the DPX recombination operator is outperformed by CX for all tested instances. In case of `tai100a` and `tai256c`, mutation becomes superior to recombination for some jump distances. These instances have an n/ℓ value close to 4, and are considered to be unstructured. Although `kroA100` has an extremely rugged landscape ($n/\ell \sim 4$), the MA with CX recombination is better than or as good as the MAs using mutation. The DPX recombination shows to be efficient only for the instances `tai100b` and `pmr100b`.

In our experiments, we could not find a general rule for setting the optimum mutation jump distance. Besides the fact that the optimum depends on the structure of the landscape (e.g., the optimum for `tai100a` is 20, while for `tai100b`, it is 40), we found that the running times of the algorithms also had an influence. However, it seems that the optimum jump distance lies below the correlation length of the landscape (below $n/4$) for unstructured landscapes.

The results can be summarized as follows. If the landscape is highly rugged ($n/\ell \sim 4$) and the average number of improvements (iterations) made by the local search is low, recom-

TABLE IV
COMPARISON OF FOUR ALGORITHMS FOR THE QAP

		$\mathbf{MA}_{R=1/3}$	$\mathbf{MA}_{R=1}$	**Ro-TS**	**Re-TS**	**FANT**	**MMAS**	**SA**	
instance	best known	avg.%	avg.%	avg.%	avg.%	avg.%	avg.%	avg.%	t/s
els19	17212548	**0.000**	**0.000**	0.124	**0.000**	**0.000**	**0.000**	31.96	5
chr25a	3796	1.222	**0.000**	6.567	5.297	4.549	*0.669	49.52	15
bur26a	5426670	0.003	**0.000**	0.001	0.029	0.020	**0.000**	0.390	20
nug30	6124	0.004	0.007	0.020	0.589	0.219	***0.000**	0.882	20
kra30a	88900	0.369	**0.000**	0.223	0.612	0.931	0.119	4.601	20
ste36a	9526	**0.045**	0.091	0.128	1.189	0.545	*0.051	12.55	30
tai60a	7208572	1.314	1.597	1.313	**0.794**	2.577	*1.159	3.199	90
tai80a	13557864	1.106	1.305	1.023	**0.482**	2.525	*0.768	3.298	180
tai100a	21125314	1.089	1.252	0.909	**0.385**	2.569	*0.728	1.848	300
sko100a	152002	**0.096**	0.127	0.191	0.397	0.474	*0.195	2.942	300
tai60b	608215054	**0.000**	**0.000**	1.898	0.929	0.213	0.075	1.760	90
tai80b	818415043	0.191	**0.004**	2.929	1.602	0.821	0.718	5.092	180
tai100b	1185996137	0.076	**0.038**	2.373	1.469	0.360	0.328	6.696	300
tai150b	498896643	**0.361**	0.397	2.851	1.775	1.176	1.167	3.787	600
tho150	8133484	**0.151**	0.202	0.548	0.488	0.765	*0.395	2.939	600
tai256c	44759294	0.070	0.099	0.326	0.266	0.273	*0.067	0.370	1200

bination becomes inefficient, and mutation with a jump distance below the correlation length of the landscape is the better choice. DPX recombination is only efficient if the landscape is highly structured (low n/ℓ and correlation between local optima). CX recombination is the better choice in comparison to DPX and mutation for all landscapes with low n/ℓ and for rugged landscapes ($n/\ell \sim 4$) with low epistasis (high flow or distance dominance). The reason why CX is superior to DPX on unstructured landscapes such as tai100a is that the local search converges much faster than applied to solutions produced by CX. For example, in a time limit of 300 s, the MA using CX processes 730 generations on average while the MA with DPX processes only 120 generations. Hence, the quantity of visited locally optimum solutions is much higher for the former MA.

B. Comparison of Heuristic Algorithms for the QAP

To evaluate the performance of our MA using CX relative to other proposed approaches for the QAP, we performed a comparison study with 5 of its competitors. Included in the comparison are the *robust tabu search algorithm* (Ro-TS) [78], the *reactive tabu search algorithm* (Re-TS) [27], the *fast ant colony algorithm* (FANT) incorporating local search [79], [80], *simulated annealing* (SA) according to Connolly [21], and the Min–Max Ant System (MMAS) [81], [13].

We selected 16 QAP instances from the QAPLIB, ranging from $n = 19$ locations up to $n = 256$. Included are structured as well as unstructured instances.

Table IV displays the results obtained by our MA, together with the results produced by the robust tabu search algorithm (Ro-TS), reactive tabu search algorithm (Re-TS), fast ant colony algorithm (FANT), simulated annealing (SA), and the min–max ant system (MMAS). The five competitors belong to the best heuristic approaches to the QAP currently 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. All algorithms were terminated after the same predefined time for each instance.

The MA was run in two variants, one with a diversification rate of $R = 1$, and one with a diversification rate of $R = 1/3$. In the table, *instance* denotes the name of the QAP instance from the QAPLIB (the number indicates its size n). The best-known solution is provided for each instance in the second column. For each algorithm, *avg.* shows the average percentage excess over the best-known solution obtained within 30 runs, and t/s displays the time in seconds allowed for each of the 30 runs. The min–max ant system was run in two variants, one incorporating 2-opt local search and the other using tabu search. The results for the best of the two variants are displayed in the column denoted MMAS. An asterisk indicates that the result was obtained by the tabu search variant.

The results indicate that the MA is superior, in terms of solution quality within a given time limit, to the alternative approaches for all but five instances. The completely unstructured instances tai60a, tai80a, and tai100a, are solved more effectively by Ro-TS, Re-TS, and MMAS, but these instances have no practical importance [30]. For instances nug30 and tai256c, MMAS shows a slightly better performance than the MA. However, the MA using the mutation operator finds the best-known solution of tai256c 8 out of 30 times with an average percentage excess of 0.038%, and thus outperforms MMAS. For the problems tai60a, tai80a, and tai100a, the results of the mutation based MA are 1.385%, 1.001%, and 0.936%, respectively. In an additional experiment, the MA with CX has been shown to be superior to an MA with a combination of CX and mutation for the remaining problems.

In fact, it is hardly surprising that the MA is not the top performer for the three uniformly randomly generated instances (tai60a, tai80a, and tai100a). 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.

TABLE V
SHORTEST AND AVERAGE TIMES TO REACH THE BEST-KNOWN SOLUTION WITH THE MA

instance	gen	average cost	quality	min. t in s	avg. t in s
chr25a	131	3796.0	0.00%	0.3	2.6
bur26a	20	5426670.0	0.00%	0.2	1.0
nug30	234	6124.0	0.00%	0.8	7.1
kra30a	83	88900.0	0.00%	0.4	2.7
ste36a	925	9526.0	0.00%	2.0	36.7
tai60b	151	608215054.0	0.00%	7.5	23.2
tai80b	787	818415043.0	0.00%	54.8	258.3
tai100b	1312	1185996137.0	0.00%	57.5	629.1
G124-02	12	26.0	0.00%	9.2	33.2
G124-16	2	898.0	0.00%	7.4	15.0
lipa90b	121	12490441.0	0.00%	5.2	72.1
pmd100c	58	4992484.0	0.00%	19.4	41.1
pmr100c	143	12223044.0	0.00%	21.1	72.0

An explanation why the tabu search algorithms and the MMAS with tabu search perform better than MMAS with 2-opt local search and our MA can be found with a little help from the fitness landscape analysis. For the instances tai60a, tai80a, and tai100a, the average number of iterations of the local search is low. Tabu search allows finding much better solutions since it does not stop in a local optimum. It searches efficiently for other improvements once a local optimum has been reached. For totally unstructured landscapes, this appears to be the best strategy since a new local search in another region of the search space performed by a meta-heuristic is ineffective due to the missing structure.

To illustrate the behavior of our MA using CX in more detail, Table V shows the times (of 30 runs) required by the MA to reach the best known solution. In the figure, *gen* denotes the number of generations, *average* the average value of the objective function [see (1)], *deviation* the average deviation from the known optima, and *min.* t *in* s and *avg.* t*t in* s the minimum and average runtime in seconds, respectively. 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 alternative approaches used in the comparison are—to the best of our knowledge—the most useful methods available today for solving QAP instances, several other techniques have been published. For example, another high-quality algorithm based on the ant system, called HAS–QAP [12], has been proposed which is not included in the comparison since the source code was not available to us. However, the MMAS algorithm appears to be superior to HAS–QAP when selecting the best of either MMAS with 2-opt or MMAS with tabu search [14]. The results presented in [12] indicate that our MA is superior to HAS–QAP in terms of CPU time and solution quality for problems with $n > 30$.

Other approaches for the QAP exist, such as, for example, the PGA [26], the SAGA [23], the Ant System [11], the *combinatorial evolution strategy (CES)* [18], and the genetic algorithm in [19]. These algorithms cannot be compared directly with the results presented here because only rather small instances $(n \leq 64)$ have been considered, and different hardware platforms have been used to perform the experiments.

Comparison studies have been made in [31], [82] for relatively small instances with the conclusion that genetic algorithms perform relatively poorly, but that hybridizing may enhance the search significantly. The genetic hybrids [25] have been proven to be effective for finding near optimum solutions: in long runs (up to 22 h on a Sun Sparcstation 10), new best solutions could be found for some of the problems contained in QAPLIB. In comparison studies [30], [79], the genetic hybrids have been shown to be the best algorithms for most of the studied instances. However, the results presented in [79] are worse than the results obtained with our MA on the structured tai $*$ b problems in both quality and running times. The latter, however, cannot be compared directly because of the different hardware architectures.

VI. CONCLUSIONS

In this paper, the fitness landscapes of several instances of the quadratic assignment problem (QAP) have been analyzed. To investigate the local and global properties of the fitness landscapes, an autocorrelation analysis and a fitness distance correlation analysis have been performed. While the local properties have a large impact on the effectiveness of a local search algorithm, the global structure can be exploited by a population-based search. Thus, the analysis is especially important for the design of hybrid algorithms.

The analysis has revealed that most problems are unstructured with respect to their global and local structure. Many instances represent highly rugged landscapes, and the local minima are totally uncorrelated; they appear to be distributed uniformly over the search space. Furthermore, we have shown that the low dominance proposed as a measure for problem difficulty for the QAP is not suited to predict the hardness of a problem instance since it does not reflect the structure in a fitness landscape. However, with flow and distance dominance, a measure for gene interaction (epistasis) can be defined that allows distinguishing between instances of the QAP if the correlation length of the landscape as a measure of the landscape ruggedness, and the correlation of the fitness and distance of the local minima in the landscape, are also considered.

To investigate the influence of epistasis on the structure of the fitness landscapes, we generated structured instances with

varying flow dominance and thus varying epistasis. We have shown that there are instances exhibiting highly structured, rather smooth landscapes in the QAP besides the ones obtained from QAPLIB, a public library for QAP instances.

Epistasis, correlation length and fitness distance correlation considered together, allows dividing QAP instances into four classes. The first class, characterized by high epistasis, high landscape ruggedness (low correlation length), and uncorrelated local optima, is the hardest of all types of instances. Both local search and population-based meta-heuristics are relatively ineffective since there is no local or global structure to exploit. The second class of instances has a rugged landscape but low epistasis and no correlation between fitness and distance of the local optima. These instances are considered to be easier to solve but still are very unstructured. The instances that are easiest to search have a relatively smooth landscape (high correlation length), and the local optima are correlated. Instances with a relatively smooth landscape, but very low epistasis and no global correlation structure constitute the fourth class. For this class, effective meta-heuristics are required, since the local search produces solutions which lie more than 20% above the optimum.

We applied a memetic algorithm (MA), i.e., an evoltuinary algorithm incorporating 2-opt local search to instances of the four classes with recombination or mutation operators. For the first and hardest class, it turned out that recombination is ineffective, which is not surprising since recombination is thought of as exploiting an existing structure in a problem, and for these problems the structure is missing. For all other types of instances the MA with our recently proposed recombination operator CX performed best. A form of recombination introducing a high amount of implicit mutations as realized in our DPX recombination operator appears to be effective only for the instances with correlated local optima and thus for the class of problems with an exploitable global structure. In the other cases, the high amount of implicit mutations causes the local search to iterate longer. Hence fewer local optima can be visited in a given time limit compared to the MA with CX.

Thus, we have shown how a fitness landscape analysis is useful for finding a suitable choice of evolutionary operators in a MA for the QAP. However, the fitness landscape analysis may also be used for other meta-heuristics such as the ant colony system: The best choice of local search or tabu search can be easily determined based on a landscape analysis.

To compare the performance of our memetic algorithm, we conducted a comparison study on a set of QAP instances with a problem size up to $n = 256$. In particular, we compared the MA with five very good alternative heuristic approaches to the QAP: *reactive tabu search*, *robust tabu search*, *fast ant colony system*, the *min–max ant system*, and *simulated annealing*. The MA using the CX recombination operator is shown to be superior to the other approaches, except for the hardest, totally unstructured instances. Furthermore, the MA approach proves to be very robust, since the best-known solutions could be found in *all* runs with short average running times for instances with a problem size up to $n = 100$.

There are several issues for future research. For example, there is still no way to determine the optimum parameter settings for the MA, including population size, number of generations, and operator rates. Instead of relying solely on experience, self-adaptation [83], [84] or meta-optimization [85] could be a possibility to adjust these parameters. Furthermore, up to now, there is not much knowledge about the structure of fitness landscapes. New techniques are required to find exploitable features within problems and alternative, reliable measures for problem difficulty are of interest. It would be helpful to find measures that give more detailed information about the design of a highly effective evolutionary operator for a given problem, since it is yet unknown when a recombination operator with a high amount of implicit mutations will be more useful than an information preserving operator like CX.

ACKNOWLEDGMENT

The authors wish to thank D. Fogel and the anonymous referees for their valuable comments on a previous version of the paper. They also would like to thank M. Dorigo and T. Stützle for providing the source code for the Min–Max Ant System, and for their very helpful suggestions to improve the quality of the paper.

REFERENCES

[1] T. C. Koopmans and M. J. Beckmann, "Assignment problems and the location of economic activities," *Econometrica*, vol. 25, pp. 53–76, 1957.
[2] L. Steinberg, "The backboard wiring problem: A placement algorithm," *SIAM Rev.*, vol. 3, pp. 37–50, 1961.
[3] R. E. Burkhard and J. Offerman, "Entwurf von Schreibmaschinentastaturen mittels quadratischer Zuordnungsprobleme," *Z. Oper. Res.*, vol. 21, pp. B121–B132, 1977.
[4] E. J. McCormick, *Human Factors Engineering*. New York: McGraw-Hill, 1970.
[5] J. Krarup and P. M. Pruzan, "Computer-aided layout design," *Math. Programming Study*, vol. 9, pp. 75–94, 1978.
[6] A. N. Elshafei, "Hospital layout as a quadratic assignment problem," *Oper. Res. Quart.*, vol. 28, pp. 167–179, 1977.
[7] J. W. Dicky and J. W. Hopkins, "Campus building arrangement using TOPAZ," *Transp. Res.*, vol. 6, pp. 59–68, 1972.
[8] J. S. Gero, V. A. Kazakov, and T. Schnier, "Genetic engineering and design problems," in *Evolutionary Algorithms in Engineering Applications*, D. Dasgupta and Z. Michalewicz, Eds. Berlin: Springer, 1997, pp. 47–68.
[9] P. Cejchan, private communication, 1998.
[10] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: Freeman, 1979.
[11] 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.
[12] L. M. Gambardella, É. D. Taillard, and M. Dorigo, "Ant colonies for the QAP," *J. Oper. Res. Soc.*, 1999. Also available as Tech. Rep. IDSIA-4-97, IDSIA, Lugano, Switzerland.
[13] T. Stützle and H. H. Hoos, "$\mathcal{MAX}$–$\mathcal{MIN}$ ant system and local search for combinatorial optimization problems," in *MIC'97: 2nd Metaheuristics Int. Conf.*, 1997.
[14] T. Stuetzle and M. Dorigo, "ACO algorithms for the quadratic assignment problem," in *New Ideas in Optimization*, D. Corne, M. Dorigo, and F. Glover, Eds. New York: McGraw-Hill, 1999. Also available as Tech. Rep. IRIDIA/99-2, Université Libre de Bruxelles, Belgium.
[15] M. Dorigo, V. Maniezzo, and A. Colorni, "Positive feedback as a search strategy," Tech. Rep. 91-016, Politecnico di Milano, Milano, Italy, 1991.
[16] ——, "The ant system: Optimization by a colony of cooperating agents," *IEEE Trans. Syst., Man, Cybern. B*, vol. 26, no. 1, pp. 29–41, 1996.
[17] M. Dorigo and G. Di Caro, "The ant colony optimization meta-heuristic," in *New Methods in Optimization*, D. Corne, M. Dorigo, and F. Glover, Eds. New York: McGraw-Hill, 1999. Also available as Tech. Rep. IRIDIA/99-1, Université Libre de Bruxelles, Belgium.
[18] V. Nissen, "Solving the quadratic assignment problem with clues from nature," *IEEE Trans. Neural Networks*, vol. 5, no. 1, pp. 66–72, 1994.

[19] D. M. Tate and A. E. Smith, "A genetic approach to the quadratic assignment problem," *Comput. Oper. Res.*, vol. 22, no. 1, pp. 73–83, 1995.
[20] R. E. Burkard and F. Rendl, "A thermodynamically motivated simulation procedure for combinatorial optimization problems," *Eur. J. Oper. Res.*, vol. 17, pp. 169–174, 1984.
[21] D. T. Connolly, "An improved annealing scheme for the quadratic assignment problem," *Eur. J. Oper. Res.*, vol. 46, pp. 93–100, 1990.
[22] S. Ishii and M. Sato, "Constrained neural approaches to quadratic assignment problems," *Neural Networks*, vol. 11, pp. 1073–1082, 1998.
[23] E. D. Brown, L. C. Huntley, and R. A. Spillance, "A parallel genetic heuristic for the quadratic assignment problem," in *Proc. 3rd Conf. Genetic Algorithms*, J. D. Schaffer, Ed. CA: Morgan Kaufmann, 1989, pp. 406–415.
[24] F. G. Tinetti, J. Carrizo, and P. Moscato, "A computational ecology for the quadratic assignment problem," in *Proc. 21st Meet. Informatics Oper. Res.*, SADIO, Buenos Aires, Argentina, 1992.
[25] 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: Amer. Math. Soc., 1994, vol. 16, DIMACS Ser. Discrete Math. Theoretical Comput. Sci., pp. 137–187.
[26] H. Mühlenbein, "Parallel genetic algorithms, population genetics and combinatorial optimization," in *Proc. 3rd Int. Conf. Genetic Algorithms*, J. D. Schaffer, Ed. CA: Morgan Kaufmann, 1989, pp. 416–421.
[27] R. Battiti and G. Tecchiolli, "The reactive tabu search," *ORSA J. Comput.*, vol. 6, no. 2, pp. 126–140, 1994.
[28] J. Skorin-Kapov, "Tabu search applied to the quadratic assignment problem," *ORSA J. Comput.*, vol. 2, no. 1, pp. 33–45, 1990.
[29] ——, "Extensions of a tabu search adaptation to the quadratic assignment problem," *Comput. Oper. Res.*, vol. 21, no. 8, pp. 855–865, 1994.
[30] É. D. Taillard, "Comparison of iterative searches for the quadratic assignment problem," *Location Sci.*, vol. 3, pp. 87–105, 1995.
[31] V. Bachelet, P. Preux, and E.-G. Talbi, "Parallel hybrid meta-heuristics: Application to the quadratic assignment problem," in *Proc. Parallel Optimiz. Colloquium*, Versailles, France, 1996.
[32] 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.
[33] 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: Amer. Math. Soc., 1994, vol. 16, DIMACS Ser. Discrete Math. Theoretical Comput. Sci., pp. 237–261.
[34] V.-D. Cung, T. Mautor, P. Michelon, and A. Tavares, "A scatter search based approach for the quadratic assignment problem," in *Proc. 1997 IEEE Int. Conf. Evol. Comput. (ICEC)*, T. Baeck, Z. Michalewicz, and X. Yao, Eds. Indianapolis, IN/New York: IEEE Press, 1997, pp. 165–170.
[35] V. Maniezzo, "Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem," Tech. Rep. CSR 98-1, C. L. in Scienze dell'Informazione, Universitá di Bologna, Sede di Cesena, Italy, 1998.
[36] P. Moscato, "On evolution, search, optimization, genetic algorithms and martial arts: Toward memetic algorithms," Caltech Concurrent Computation Program, California Institute of Technology, Pasadena, Tech. Rep. 790, 1989.
[37] P. Moscato and M. G. Norman, "A memetic approach for the traveling salesman problem implementation of a computational ecology for combinatorial optimization on message-passing systems," in *Parallel Computing and Transputer Applications*, M. Valero, E. Onate, M. Jane, J. L. Larriba, and B. Suarez, Eds. Amsterdam, The Netherlands: IOS Press, 1992, pp. 177–186.
[38] 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.
[39] 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—Proc. 1st Workshop, PPSN I*, H. P. Schwefel and R. Männer, Eds. Berlin, Germany: Springer, 1991, vol. 496, Lecture Notes in Computer Science, pp. 109–116.
[40] T. Yamada and R. Nakano, "Scheduling by genetic local search with multi-step crossover," in *Proc. 4th Conf. Parallel Problem Solving from Nature—PPSN IV*, H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, Eds. Berlin, Germany: Springer, 1996, vol. 1141, Lecture Notes in Computer Science, pp. 960–969.
[41] B. Freisleben and P. Merz, "A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems," in *Proc. 1996 IEEE Int. Conf. Evol. Comput.*, T. Bäck, H. Kitano, and Z. Michalewicz, Eds. Piscataway, NJ: IEEE Press, 1996, pp. 616–621.
[42] ——, "New genetic local search operators for the traveling salesman problem," in *Proc. 4th Int. Conf. Parallel Problem Solving from Nature—PPSN IV*, H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, Eds. Berlin, Germany: Springer, 1996, vol. 1141, Lecture Notes in Computer Science, pp. 890–900.
[43] P. Merz and B. Freisleben, "Genetic local search for the TSP: New results," in *Proc. 1997 IEEE Int. Conf. Evol. Comput.*, T. Bäck, Z. Michalewicz, and X. Yao, Eds. Piscataway, NJ: IEEE Press, 1997, pp. 159–164.
[44] ——, "Memetic algorithms and the fitness landscape of the graph bi-partitioning problem," in *Proc. 5th Int. Conf. Parallel Problem Solving from Nature—PPSN V*, A.-E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, Eds. Berlin, Germany: Springer, 1998, vol. 1498, Lecture Notes in Computer Science, pp. 765–774.
[45] ——, "On the effectiveness of evolutionary search in high-dimensional NK-landscapes," in *Proc. 1998 IEEE Int. Conf. Evol. Computat.*, D. Fogel, Ed. Piscataway, NJ: IEEE Press, 1998, pp. 741–745.
[46] ——, "A comparison of memetic algorithms, tabu search, and ant colonies for the quadratic assignment problem," in *1999 Congr. Evol. Comput. (CEC'99)*, P. Angeline, Ed. Piscataway, NJ: IEEE Press, 1999, pp. 2063–2070.
[47] P. Merz, "Self-adaptive memetic algorithms for combinatrial optimization problems," Ph.D. thesis, Department of Electrical Engineering and Computer Science, University of Siegen, Germany, 1999. in preparation.
[48] E. S. Buffa, G. C. Armour, and T. E. Vollmann, "Allocating facilities with CRAFT," *Harvard Business Review*, vol. 42, pp. 136–158, Mar. 1964.
[49] J. L. Bentley, "Experiments on traveling salesman heuristics," in *Proc. 1st Ann. ACM-SIAM Symp. Discrete Algorithms*, San Francisco, CA, 1990, pp. 91–99.
[50] S. Wright, "The roles of mutation, inbreeding, crossbreeding, and selection in evolution," in *Proc. 6th Congr. Genetics*, vol. 1, 1932, p. 365.
[51] E. D. Weinberger, "Correlated and uncorrelated fitness landscapes and how to tell the difference," *Biol. Cybern.*, vol. 63, pp. 325–336, 1990.
[52] P. F. Stadler, "Toward a theory of landscapes," in *Complex Systems and Binary Networks*, R. Lopéz-Peña, R. Capovilla, R. García-Pelayo, H. Waelbroeck, and F. Zertuche, Eds. New York: Springer Verlag, 1995, vol. 461, Lecture Notes in Physics, pp. 77–163.
[53] ——, "Landscapes and their correlation functions," *J. Math. Chem.*, vol. 20, pp. 1–45, 1996.
[54] E. Angel and V. Zissimopoulos, "Autocorrelation coefficient for the graph bipartitioning problem," *Theor. Comput. Sci.*, vol. 191, pp. 229–243, 1998.
[55] B. Krakhofer and P. F. Stadler, "Local minima in the graph bipartitioning problem," *Europhys. Lett.*, vol. 34, pp. 85–90, 1996.
[56] T. Jones and S. Forrest, "Fitness distance correlation as a measure of problem difficulty for genetic algorithms," in *Proc. 6th Int. Conf. Genetic Algorithms*, L. J. Eshelman, Ed. San Francisco, CA: Morgan Kaufmann, 1995, pp. 184–192.
[57] S. A. Kauffman, *The Origins of Order: Self-Organization and Selection in Evolution*. Oxford, U.K.: Oxford University Press, 1993.
[58] K. D. Boese, "Cost versus distance in the traveling salesman problem," UCLA CS Department, Tech. Rep. TR-950018, Los Angeles, CA, 1995.
[59] C. R. Reeves, "Landscapes, operators and heuristic search," *Annals of Operations Research*, 1999.
[60] P. Merz and B. Freisleben, "Fitness landscapes, memetic algorithms and greedy operators for graph bi-partitioning," *Evol. Comput.*, 2000.
[61] R. E. Burkard, S. Karisch, and F. Rendl. (1991) QAPLIB—A quadratic assignment problem library. *Eur. J. Oper. Res.*, pp. 115–119
[62] T. E. Vollmann and E. S. Buffa, "The facilities layout problem in perspective," *Manage. Sci.*, vol. 12, no. 10, pp. 450–468, 1966.
[63] E. Angel and V. Zissimopoulos, "On the hardness of the quadratic assignment problem with meta-heuristics," Laboratoire de Recherche en Informatique, Université Paris Sud, Tech. Rep., France, 1997.
[64] ——, "On the landscape ruggedness of the quadratic assignment problem," *Theor. Comput. Sci.*, 1999.
[65] Y. Li and P. M. Pardalos, "Generating quadratic assignment test problems with known optimal permutations," *Comput. Optim. Appl.*, vol. 1, pp. 163–184, 1992.
[66] L. J. Fogel, A. J. Owens, and M. J. Walsh, *Artificial Intelligence Through Simulated Evolution*. New York: Wiley, 1966.

[67] I. Rechenberg, *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Stuttgart: Frommann-Holzboog, 1973.
[68] J. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: University of Michigan Press, 1975.
[69] J. R. Koza, *Genetic Programming: On the Programming of Computers by Natural Selection*. Cambridge, MA: MIT Press, 1992.
[70] P. Merz and B. Freisleben, "Fitness landscapes and memetic algorithm design," in *New Ideas in Optimization*, D. Corne, M. Dorigo, and F. Glover, Eds. London, U.K.: McGraw-Hill, 1999.
[71] R. Dawkins, *The Selfish Gene*. Oxford, U.K.: Oxford University Press, 1976.
[72] P. Moscato, "Memetic algorithms: A short introduction," in *New Ideas in Optimization*, D. Corne, M. Dorigo, and F. Glover, Eds. London, U.K.: McGraw-Hill, 1999.
[73] L. J. Eshelman, "The CHC adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination," in *Foundations of Genetic Algorithms*, G. J. E. Rawlings, Ed. San Mateo, CA: Morgan Kaufmann, 1991, pp. 265–283.
[74] P. Merz and B. Freisleben, "A genetic local search approach to the quadratic assignment problem," in *Proc. 7th Int. Conf. Genetic Algorithms*, T. Bäck, Ed. San Francisco, CA: Morgan Kaufmann, 1997, pp. 465–472.
[75] I. M. Oliver, D. J. Smith, and J. R. C. Holland, "A study of permutation crossover operators on the traveling salesman problem," in *Genetic Algorithms Appl.: Proc. 2nd Int. Conf. Genetic Algorithms*. Hillsdale, NJ: Lawrence Erlbaum, 1987, pp. 224–230.
[76] H.-P. Schwefel, *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie*. Basel: Birkhäuser Verlag, 1977, vol. 26, Interdisciplinary Systems Research.
[77] D. E. Goldberg and R. Lingle Jr., "Alleles, loci, and the traveling salesman problem," in *Proc. Int. Conf. Genetic Algorithms Appl.*: Carnegie Mellon Publishers, 1985, pp. 154–159.
[78] É. D. Taillard, "Robust taboo search for the quadratic assignment problem," *Parallel Comput.*, vol. 17, pp. 443–455, 1991.
[79] É. D. Taillard and L. M. Gambardella, "Adaptive memories for the quadratic assignment problem," Tech. Rep. IDSIA-87-97, IDSIA, Lugano, Switzerland, 1997.
[80] É. D. Taillard, "FANT: Fast ant system," Tech. Rep. IDSIA-46-98, IDSIA, Lugano, Switzerland, , 1998.
[81] T. Stützle, "$\mathcal{MAX}$–$\mathcal{MIN}$ ant system for quadratic assignment problems," Tech. Rep. AIDA-97-4, FG Intellektik, TU Darmstadt, Germany, 1997.
[82] V. Maniezzo, M. Dorigo, and A. Colorni, "Algodesk: An experimental comparison of eight evolutionary heuristics applied to the quadratic assignment problem," *Eur. J. Oper. Res.*, vol. 81, pp. 188–204, 1995.
[83] T. Bäck, "Self-adaptation in genetic algorithms," in *Proc. 1st Eur. Conf. Artif. Life. Toward a Practice of Autonomous Syst.*, F. J. Varela and P. Bourgine, Eds. Cambridge, MA: MIT Press, 1991, pp. 263–271.
[84] A. E. Eiben, R. Hinterding, and Z. Michalewicz, "Parameter control in evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 3, no. 2, pp. 124–141, 1999.
[85] B. Freisleben, "Meta-evolutionary approaches," in *Handbook of Evolutionary Computation*, T. Bäck, D. B. Fogel, and Z. Michalewicz, Eds. Oxford: Oxford Univ. Press, 1997, pp. C2.8.1–C2.8.12.

Peter Merz received the M.S. degree in computer science from the University of Siegen, Germany, in 1996. He is currently a Ph.D. candidate in computer science, and works as a Research Assistant in the Department of Electrical Engineering and Computer Science of the University of Siegen since 1996. His research interests include heuristic optimization techniques for combinatorial optimization problems, hybrid evolutionary algorithms, complex adaptive systems, and artificial life.

Bernd Freisleben is currently a Professor of computer science in the Department of Electrical Engineering and Computer Science at the University of Siegen, Germany. He received the Master's degree in computer science from the Pennsylvania State University, USA, in 1981, and the Ph.D. degree in computer science from the University of Darmstadt, Germany, in 1985. His research interests include evolutionary computation, neural networks, network computing, and applications of soft computing.