Chapter 7
Memetic Algorithms and Fitness Landscapes in Combinatorial Optimization

Peter Merz

7.1 Introduction

Combinatorial optimization problems (COPs) arise in many practical applications in the fields of management science, biology, chemistry, physics, engineering, and computer science. Although the search space is comprised of a finite number of candidate solutions, many of these problems are very complex and thus hard to solve. Often, the search space grows exponentially with the problem size rendering enumeration schemes impractical. Moreover, for many problems it has been shown that they are NP-hard, hence no polynomial time algorithm is known to find optimum solutions. Therefore, effective meta-heuristics are required to find (near) optimum solutions in short time. Memetic algorithms are known to perform well for a wide range of combinatorial optimization problems. Still, an open question is when and why they perform so well. After providing an overview and a common outline of memetic algorithms for combinatorial optimization problems in section 2, we introduce the concept of fitness landscapes in section 3 to address these two questions. In Section 4 and 5 we present case studies of the TSP and the BQP, respectively, in which we show and discuss results from the fitness landscape analysis. Furthermore, we discuss the state-of-the-art meta-heuristics for these problems. Section 6 concludes the chapter.

7.2 MAs in Combinatorial Optimization

The travelling salesman Problem (TSP) is one of the best-known combinatorial optimization problems. Often, new new ideas in meta-heuristics have initially been tested on the TSP and were applied afterwards to other combinatorial problems.

Peter Merz
University of Applied Sciences and Arts Hannover,
Department of Computer Science and Business Administration, 30459 Hannover, Germany
e-mail: `peter.merz@fh-hannover.de`

Hence, it is not surprising that the first memetic algorithms have been developed for the TSP. In the late 80s, several attempts have been made to apply evolutionary algorithms to the travelling salesman problem. Especially, when using recombination, many researchers discovered that it is necessary to use some form of local search within the evolutionary framework [334]. One of the reasons why recombination-based evolutionary algorithms fail to perform well on the TSP is that it is not trivial to recombine some of the edges of two or more TSP tours into a single tour such that all edges are from at least one of the parents and the resulting edge set is a valid TSP tour. Most recombination operators introduce implicit mutations by adding random edges of arbitrary length to ensure feasibility [230, 565, 581, 931]. As a consequence, the offspring tend to be much worse than their parents if the parents have a high fitness. Therefore, researchers considered applying local search to remove these arbitrary long edges from the tours.

7.2.1 Combinatorial Optimization

According to [303], a combinatorial optimization problem P is either a minimization problem or a maximization problem, and it consists of

1. a set D_P of instances,
2. a finite set $S_P(I)$ of candidate solutions for each instance $I \in D_P$, and
3. a function m_P that assigns a positive rational number $m_P(I,x)$ called the solution value for x to each instance $I \in D_P$ and each candidate solution $x \in S_P(I)$.

Thus, an optimal solution for an instance $I \in D_P$ is a candidate solution $x^* \in S_P(I)$ such that, for all $x \in S_P(I)$, $m_P(I,x^*) \leqslant m_P(I,x)$ if P is a minimization problem, and $m_P(I,x^*) \geqslant m_P(I,x)$ if P is a maximization problem.

Due to the fact that the set of candidate solutions is finite, an algorithm for finding an optimum solution always exists. This algorithm, referred to as *exhaustive search*, simply evaluates and compares $m_P(I,x)$ for all $x \in S_P(I)$. Unfortunately, the search space of many combinatorial problems grows exponentially with the problem size, i.e., the number of components in a solution vector x. Thus, this complete enumeration scheme becomes impractical. For a large class of combinatorial optimization problems no alternative algorithms running in polynomial time are known. This phenomenon has led to the development of complexity theory [303], and in particular, to the theory of NP-completeness.

Combinatorial optimization can be considered as a special case of discrete optimization. However, in discrete optimization the search space is not always finite. In contrast to integer programming, combinatorial optimization refers to problems on graphs, matroids and other discrete structures.

7.2.2 MA Outline

Beginning with Brady [81], many researchers have made consequent use of local search in their evolutionary algorithms for the TSP [86, 331, 622, 637, 896]. These

Algorithm 12. The Memetic Algorithm

```
begin
    foreach S in Population do S ← LocalSearch(Init());
    while not terminated do
        Offspring ← ;
        for i ← 0 to crossovers do
            A ← Select(Population);
            B ← Select(Population);
            C ← LocalSearch(Recombine(A, B));
            Offspring ← OffSpring + C;
        endfor
        for i ← 0 to mutations do
            A ← Select(Population);
            C ← LocalSearch(Mutate(A));
            Offspring ← OffSpring + C;
        endfor
        Population ← Select(Population, Offspring);
    endw
end
```

approaches can be classified as memetic algorithms although they have not been called so at the time they have been proposed. Some researchers used the term Genetic Local Search [285, 286, 478, 584, 896], others described them as hybrids of evolutionary and local search. Still today, many researchers use the same basic MA framework that is shown in Alg.12. In this framework, local search is consequently applied to all newly created solutions, more precisely to all the members of the initial population created by some initialization operator, and those solutions created by the mutation and recombination operators. In the framework, recombination and mutation are treated independent of each other. In some MAs, mutation is only applied after crossover. However, we concentrate on the framework above since it allows for mutation–only MAs.

When using recombination, selection becomes highly important, since there is a high probability of premature convergence. Due to the fact that local search is expensive, MAs tend to have relatively small population sizes (10–40 individuals). Compared to EAs without local search, the problem of convergence is more severe. When the population only contains very similar solutions, recombination / combined with local search will likely discover the same solutions again and again. It is therefore important to keep diversity in the population. There are several methods to deal with diversity preservation depending on the type of selection: In selection for recombination, one can choose to recombine only those individuals which are sufficiently far away from each other in the search space. Another approach is to consider diversification in the recombination operator itself as has done in HX or DPX [76, 246, 247]. Moreover, in selection for survival duplicates may be removed such that each indivdual is found only once in the population [247] or replacement scheme may be used that replaces similar solutions based on a distance threshold

[285, 286]. Finally, a restart mechanism can be used to diversify the population once convergence has been detected [246, 581].

7.2.3 *Related Meta-Heuristics*

There are several meta-heuristics which are similar to MAs. Most notably, Scatter Search [313] and Iterated/Stochastic Local Search [533]. While the former incorporates a recombination meachanism as and the MA framework above, the latter can be considered as a special case of the MA above. In that case the population is reduced to 1 and only mutation is used (#recombinations=0,#mutations=1), which simplifies the algorithm significantly. Iterated local search (ILS) was also first proposed for the TSP [427], but has been applied later on to various combinatorial problems [533]. The outline of iterated local search is shown in Alg. 13.

Algorithm 13. Iterated Local Search.

```
begin
    S ← Init();
    S ← LocalSearch(S);
    while not terminated do
        T ← Mutate(S);
        T ← LocalSearch(T);
        S ← Select(S, T);
    endw
end
```

Iterated local search is also highly similar to some instances of variable neighborhood search (VNS) [604].

7.3 Why and When MAs Work

Although many different meta-heuristics have been proposed for combinatorial optimization problems, only little is known in which cases they are effective. Moreover, every approach comes with a considerable number of parameters and it is often not known which parameter settings are optimum due to the huge parameter space and the computational time required for testing all possible combinations.

In the case of MAs, it would be highly desirable to have guidelines for the development of MAs for new or untackled combinatorial optimization problems. Important questions that arise are: Which local search operator or neighborhood to choose, how many iterations to apply local search, to use recombination or mutation, how to mutate or recombine and so forth. Fitness landscape analysis has been shown to be valuable when trying to find answers to these questions. We therefore provide a short overview of fitness landscapes and statistics methods to analyse them.

7.3.1 The Concept of Fitness Landscapes

The concept of *fitness landscapes* [941], introduced to illustrate the dynamics of biological evolutionary optimization, has been proven to be very powerful in evolutionary theory. As metioned before, the concept has been shown to be useful for understanding the behavior of combinatorial optimization algorithms and can help in predicting their performance. Regarding 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 $f : S \rightarrow \mathbb{R}$, which assigns a real–valued fitness to each of the points in S, and a distance measure $d : S \times S \rightarrow \mathbb{R}$, which defines the spatial structure of the landscape. A 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 $diam\, G_L$ of the landscape is another important property: it is defined as the maximum distance between any two points in the search space.

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

7.3.2 NK-*Landscapes*

To study rugged fitness landscapes, Kauffman [451, 452] developed a formal model for gene interaction which is called the *NK*-model. In this model, N refers to the number of parts in the system, i.e. genes in a genotype or amino acids in a protein. Each part makes a fitness contribution which depends on the part itself and K other parts. Thus, K reflects how richly cross-coupled the system is; it measures the epistasis, i.e. the richness of interactions among the components of the system.

Each point in the *NK*-fitness landscape is represented by a bit string of length N and can be viewed as a vertex in the N-dimensional hypercube. The fitness f of a point $b = b_1, \ldots, b_N$ is defined as follows:

$$f(b) = \frac{1}{N} \sum_{i=1}^{N} f_i(b_i, b_{i_1}, \ldots, b_{i_K}), \tag{7.1}$$

where the fitness contribution f_i of the gene at locus i depends on the allele (value of the gene) b_i and K other alleles $b_{i_1}, \ldots, b_{i_K}$. The function $f_i : \{0,1\}^{K+1} \rightarrow \mathbb{R}$ assigns a uniformly distributed random number between 0 and 1 to each of its 2^{K+1} inputs. The values for $i_1, \ldots, i_K$ are chosen randomly from $\{1, \ldots, N\}$ or from the left and right of locus i. The former is called the random neighbor model while the latter is called the adjacent neighbor model. Since the random neighbor model is NP-hard and the adjacent model is not [929], we focus on the random case. With the *NK* model, the "ruggedness" of a fitness landscape can be tuned by changing the value of K and thus the number of interacting genes per locus. Low values of K indicate low epistasis and high values of K indicate high epistasis.

7.3.3 Analysis of Fitness Landscapes

7.3.3.1 Autocorrelation Analysis

The local properties of the landscape have a strong influence on the effectiveness of a local search, since in a local search the decision which point has to be visited next is based solely on these local properties. The properties can be analyzed with statistical methods such as autocorrelation/random walk correlation analysis. These methods calculate (or estimate) the correlation of neighboring points in the search space with respect to the local search neighborhood. The *random walk correlation function* [845, 846, 928]

$$r(s) = \frac{1}{\sigma^2(f)\,(m-s)} \sum_{t=1}^{m-s} (f(x_t) - \bar{f})(f(x_{t+s}) - \bar{f}) \tag{7.2}$$

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). A step denotes here a move from the current solution to a neighboring solution in the fitness landscape. Typical random walk correlation functions for the TSP and *NK*-landscapes are displayed in Fig. 7.1.

Based on this correlation function, the correlation length ℓ [846] of the landscape is defined as

$$\ell = -\frac{1}{\ln(|r(1)|)} \tag{7.3}$$

for $r(1) \neq 0$. The correlation length directly reflects the ruggedness of a landscape. The smaller the value for ℓ, the more rugged the landscape. A landscape is said to be

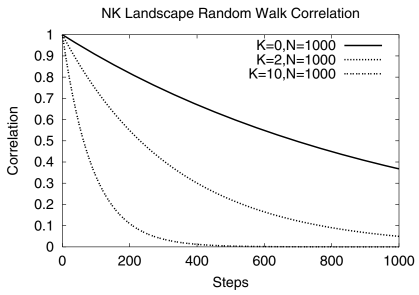

Fig. 7.1. The random walk correlation functions of *NK* landscapes (right) with varying K.

smooth if there is high correlation between neighboring points (correlation length is large), and rugged if there is low or no correlation between neighboring solutions (correlation length is small). It can be observed that higher correlation leads to a higher number of iterations in a local search until a local optimum is reached and may also lead to a higher fitness as Kauffman concludes [451]: On the other hand, if the correlation is low, the local search terminates after few iterations in a local optimum with relatively low fitness [451]. The correlation length as a measure of landscape ruggedness can be utilized to compare different neighborhoods for a problem (assumed that the neighborhoods have the same size). The higher the correlation, the more effective the local search.

Alternative landscapes in combinatorial optimization can be found by allowing for infeasible solutions. However, the problem becomes then to find an appropriate penalty function in order to obtain an effective local search. In the graph bipartitioning problem, local search can be performed by exchanging a vertex from one set with a vertex from the other set. An alternative is to move just one vertex to the other set. As a consequence, both sets can have different sizes and hence the solution may be infeasible. Therefore, it is required to introduce a penalty function to penalize infeasible solutions depending on the absolute difference of the sizes of the two sets. The objective becomes

$$f(V_1,V_2) = |e(V_1,V_2)| + \alpha(|V_1| - |V_2|)^2, \tag{7.4}$$

where V_1, V_2 are the two vertex sets, $e(\cdot,\cdot)$ is the number of edges between the two sets, and α is called the imbalance factor [430]. In [20], the correlation length has been used to determine the perfect imbalance factor α, resulting in the highest random walk correlation. In experiments, it could been verified that the "optimum" imbalance factor leads to the best local search performance. Thus, the correlation length can be used to find the smoothest and hence easiest landscape for a local search for a given problem. In some cases, where the correlation length is problem instance dependent, it may serve as an indicator for the hardness of the instance for a local search. For *NK*-Landscapes as well as for other combinatorial optimization problems such as the quadratic assignment problem, it can be observed that the number of iterations of a local search (the number of moves until a local optimum is reached) decreases for less correlated landscapes and the resulting solution quality becomes worse [451, 581, 587].

7.3.3.2 Fitness Distance Correlation

The effectiveness of the evolutionary meta-search depends on the global properties of the fitness landscape. Since in the MAs discussed in this chapter, the evolutionary variation operators are applied to locally optimum solutions, the distribution of the local optima is the most important global property of a landscape. The distribution can be analyzed with a fitness distance correlation analysis of the local optima – the peaks in the fitness landscape. The fitness distance correlation (FDC) coefficient ρ is defined as

$$\rho(f, d_{opt}) = \frac{\mathrm{Cov}(f, d_{opt})}{\sigma(f)\,\sigma(d_{opt})}, \tag{7.5}$$

where $\mathrm{Cov}(\cdot,\cdot)$ denotes the covariance of two random variables and $\sigma(\cdot)$ denotes the standard deviation. The FDC determines how closely fitness and distance to the nearest optimum in the search space (denoted by d_{opt}) are related. If 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 $\rho = -1.0$ ($\rho = 1.0$) for a maximization (minimization) problem indicates that fitness and distance to the optimum are perfectly related and that search promises to be easy.

The FDC coefficient has been proposed in [435] as a measure for problem difficulty for genetic algorithms. In [14], a counterexample is presented in which a GA performs well on an uncorrelated landscape. Horjik and Manderick argue why FDC is useful for recombination [394]. [832] propose the NKP model which is a superset of the NK model and show that as K increases the correlation goes down but with no statistically significant effect on the mean fitness of the local optima. A summary of landscape metrics and related issues is provided in [438]. These papers, however, concentrate on evolutionary algorithms without local search.

In respect to MA performance, FDC analysis may reveal a correlation between the fitness and the distance of the local optima to the global optimum. The presence of correlation implies that the fitness increases the closer the local optima are to the global optimum. Therefore, an MA can exploit this feature by 'hopping' from one local optimum to the next local optimum with better fitness until the global optimum is reached. If recombination is used for variation, 'jumping' from one peak to another can be achieved if the recombination is respectful, i.e. if the resulting offspring point lies near both parents and has a distance to its parents that is no greater than the distance between the parents themselves. Compared to other forms of variation, the resulting offspring are closer to other local optima with high fitness. In such cases, it is more likely that the offspring is within a suboptimal 'basin of attraction' (with higher fitness than the parents) rather than jumping into an arbitrary direction.

Besides the FDC, fitness distance scatter plots provide useful information about a fitness landscape. In fact, there are cases in which the FDC can be misinterpreted if the plot is not considered. In Fig. 7.2, typical fitness distance plots are provided. The *NK*-landscape in the left ($N = 1024, K = 2$) is correlated but the landscape ($N = 1024, K = 11$) shown on the right is uncorrelated, the local optima appear to be randomly distributed in the search space. The correlated landscape has a massive central structure, meaning that the optimum is more or less central among other near optimum local optima. This phenomenon can be observed for several other combinatorial optimization problems and is also known as the big valley structure (for minimization problems). The structured landscape is well suited for MA based on recombination while the uncorrelated and structured landscape is not. In fact, for the latter it was shown that variation based on mutation is better than recombination, such as uniform crossover [581]. Thus, in uncorrelated landscapes, jumps in random

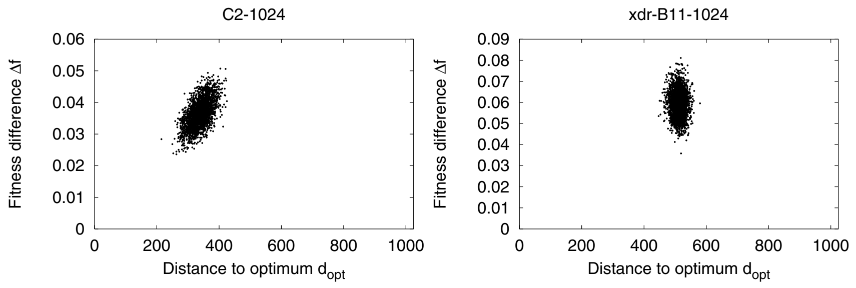

Fig. 7.2. FDC plots for two *NK*-landscapes with $K = 2$ (left) and $K = 11$ (right)

directions with a fixed jump distance are more effective than jumps toward other solutions with high fitness using variable jump distances (respectful recombination).

7.3.3.3 Advanced Fitness Landscape Analysis

Although the analysis techniques described above are valuable, they do not focus on all important aspects of the fitness landscape. Hence, in [583] we proposed some rather simple statistical analysis methods. The first addresses the question how much should be mutated in a mutation-based MA? Intuitively, the idea would be to mutate only the minimum number of components in the solution vector that is sufficient to leave the basin of attraction of the local optimum represented by the current solution. In [583], we computed the average escape rate form local optima depending on the number of mutation steps for various landscapes. Since the higher the number of mutation steps, the higher the chance to escape but also the higher the computational cost in terms of local search iterations to reach a new local optimum, we proposed to calculate the number of local search iterations per escape for various mutation strengths. In Fig.7.3, the escape rates and the number of iterations per escape for *NK*-landscapes with three different values of *K* are shown on the left, and on the right, respectively. Both escape rates and number of iterations per escape for

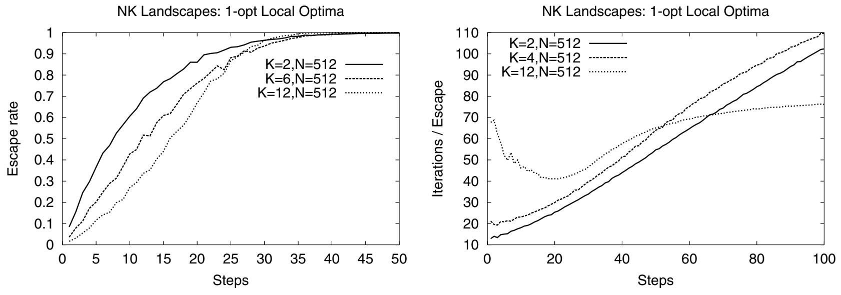

Fig. 7.3. Basins of Attraction of Local Optima in the NK-model

1-opt local optima are shown depending on the number of mutation steps performed to escape. The number of mutation steps is equal to the distance of the mutated solution and the local optimum. The left plot shows that the basins of attraction become larger with increasing K or at least escaping becomes harder: more mutation steps are required to leave the current basin of attraction. This is surprising, since with increasing K the number of local optima increases and we would expect the size of the basins of attraction to decrease. The question arises whether there is an optimum mutation rate in terms of computation costs. At which mutation rate is the number of visited local optima per time unit maximum? The answer is given in the left part of the figure. Clearly, for $K = 12$, there is an optimum at 20 mutation steps. For the landscapes with smaller K, the optimum approaches two mutation steps.

Additionally to this local search escape analysis we proposed random walk analysis starting at local optima. This analysis provides a picture of the search space from a local optimums' perspective. The idea here is that walking away from a local optimum in a random direction may show a different degradation of fitness than walking in the direction of another local optimum. How severe this difference is, depends on fitness distance correlation of the local optima. However, this analysis does not require the knowledge of a global optimum as the FDC analysis does. In Fig.7.4, the results of a random walk analysis for the *NK*-model is shown for different values of K. The random walks in the direction of another local optimum (simulating recombination) are denoted by 'rec', the random walks in a random direction (simulating mutation) or denoted by 'rw'. In the left, the average fitness values of the solutions on a typical random walk path are displayed over the distance from the starting point of the random walk (parent A). As expected, the fitness decreases when moving away from A. When approaching B the fitness increases as expected. The fitness of the solutions halfway on a random walk between solution A and B tell an interesting story. Let C denote such a point with $d(A,C) = d(A,B)/2$. For $K = \{2,6\}$, this fitness is considerably higher than for the solutions on a random walk in an arbitrary direction indicating that recombination produces much better solutions than mutation. For $K = 12$, this effect can not be observed. Here, the fitness halfway on a random walk between A and B is similar to the fitness on an arbitrary random walk, indicating that recombination is not superior to mutation for this landscape. This is not surprising since the fitness distance correlation analysis reveals that the local optima are randomly distributed in the search space. On the right of the figure, the fitness difference of points on directed and undirected random walks is provided depending on the distance to parent A: The fitness difference $f(C_{rec}) - f(C_{rw})$ for points C_{rec} on a directed random walk and points C_{rw} on an undirected random walk depending on the distance $d(A,C) = d(A,B)/2$ are shown The data is collected over a full run of a memetic algorithm. For $K = 2$, the fitness difference and the distance of the parents are correlated (upper right of the figure). The higher the distance, the better recombination performs compared to mutation (upper left of the figure). Recombination is also always superior to mutation in case of $K = 6$. However, The gain achieved with recombination is highest for distances about 100 and decreases with decreasing distance as well as increasing distance. For $K = 12$, the picture changes. Recombination and mutation perform equally well for

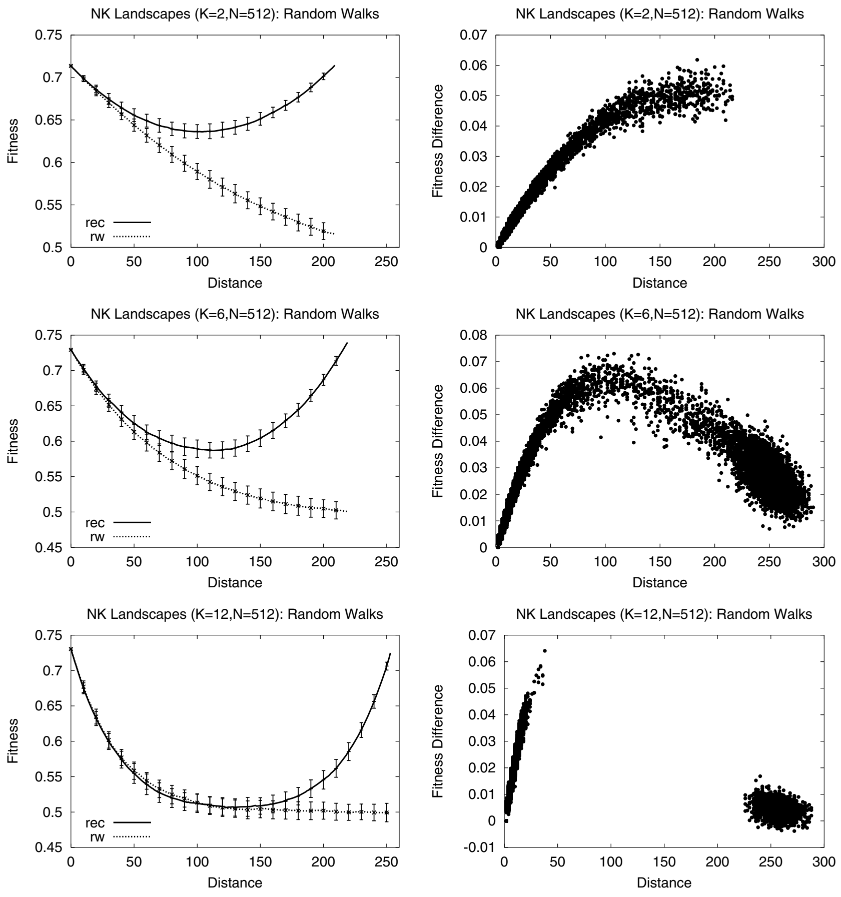

Fig. 7.4. Random Walks Starting from a Local Optimum in the *NK* model.

a distance around 250. This is the case at the beginning of the MA run. Later in the run of the MA, the average distance of the solutions in the population drops due to the effects of mutation with a relatively small mutation rate. For a parents distance smaller than 50, recombination is again superior to mutation.

7.4 Case Study I: The TSP

The travelling salesman problem (TSP) is believed to be the best-known combinatorial optimization problem. Exact methods such as branch & cut as well as many meta-heuristics have been evaluated initially on the TSP. The reason why it has attracted so many researchers is probably that it is very easy to formulate and understand: Given a set of cities and the distances between them, the problem is to find the shortest closed tour that visits each city exactly once. More formally, the tour length

$$l(\pi) = \sum_{i=1}^{n-1} d_{\pi(i),\pi(i+1)} + d_{\pi(n),\pi(1)} \tag{7.6}$$

has to be minimized, where d_{ij} is the distance between city i and city j and π a permutation of $\langle 1,2,\ldots,n\rangle$. Thus, an instance $I = \langle D\rangle$ is defined by a distance matrix $D = (d)_{ij}$, and a solution (TSP tour) is a vector π with $j = \pi(i)$ denoting city j to visit at the i-th step.

A special case of the TSP is the Euclidean TSP. Here, the distance matrix d_{ij} is symmetric, that is $d_{ij} = d_{ji} \quad \forall i,j \in \{1,2,\ldots,n\}$, and the triangle inequality holds: $d_{ij} \leqslant d_{ik} + d_{kj} \quad \forall i,j,k \in \{1,2,\ldots,n\}$. The distance between two cities is defined by the Euclidean distance between two points in the plane. These two assumptions do not lead to a reduction of the complexity; hence the general as well as the euclidian problem is NP-hard. In the following we focus on the Euclidean TSP.

7.4.1 Fitness Landscape

The TSP is also among the first combinatorial optimization problems for which researchers tried to analyze the search space. In order to define the fitness landscape for the TSP, an appropriate distance measure is required.

A suitable distance measure for TSP tours appears to be a function that counts the number of edges different in both tours: Since the fitness of a TSP tour is determined by the sum of the weights of the edges the tour consists of, the distance between two tours t_1 and t_2 can be defined as the number of edges in which one tour differs from the other. Hence

$$d(t_1,t_2) = |\{e \in E \mid e \in t_1 \wedge e \notin t_2\}|. \tag{7.7}$$

This distance measure has been used by several researchers, including [76, 286, 548, 638]. It has been shown that this distance function satisfies all four metric axioms [774].

Alternatively, a distance measure could be defined by counting the number of applications of a neighborhood search move to obtain one solution from the other. In the case of the *2-opt* move, the corresponding distance metric d_{2-opt} is bound by $d \leqslant d_{2-opt} \leqslant 2d$ [548].

7.4.1.1 Autocorrelation Analysis

Stadler and Schnabl [847] performed a landscape analysis of random TSP landscapes considering different neighborhoods: the *2-opt* and the *node exchange* neighborhood. Their results can be summarized as follows.

For the symmetric TSP, both landscapes (based on *2-opt* and *node exchange*) are AR(1) landscapes. The random walk correlation function for random landscapes is of the form

$$r(s) \approx \exp(-s/\ell) = \exp(-b/n \cdot s), \tag{7.8}$$

with n denoting the number of nodes/cities of the problem instance and b denoting the number of edges exchanged between neighboring solutions. Thus, for the *2-opt*

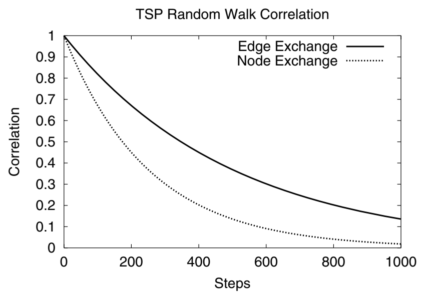

Fig. 7.5. Random walk correlation functions for the TSP based on edge exchange and node exchange.

landscape, the normalized correlation length $\xi = \ell/n$ is $\frac{1}{2}$, for the *node re-insertion* landscape ξ is $\frac{1}{3}$, and for the *node exchange* landscape ξ is $\frac{1}{4}$. This result coincides with experimentally obtained results that *2-opt* local search is much more effective than local search based on *node exchange* or *node re-insertion* [764]. The correlation functions for edge and node exchange are shown in Fig.7.5.

The formula 7.8 implies that a landscape with a strict *3-opt* neighborhood is more rugged than a landscape with a *2-opt* neighborhood. One may conclude that a *2-opt* local search performs better than a *3-opt* local search. However, the opposite is true, since the *3-opt* neighborhood is much greater than the *2-opt* neighborhood and the *3-opt* neighborhood as defined above contains the *2-opt* neighborhood. Therefore, a *3-opt* local search cannot perform worse than a *2-opt* local search in terms of solution quality.

7.4.1.2 Fitness Distance Correlation

The fitness distance correlation of local minima of the TSP has been analyzed in [75, 76] for a single instance and also in [581] for serveral other typical TSP instances. Given the distance measure described above, the results show a strong correlation between tour length and distance to the optimum solution. In fact, the TSP was the first problem to show the deep valley structure, meaning that better local optima tend to be close to the global optimum. Moreover, the global optimum is found in a big valley surounded by the other local optima and the local optima concentrate in a relatively small part of the search space. A radically different structure would be a random distribution of the local optima in the search space. Hence there would be no correlation between distance to the optimum and tour length. In Fig.7.6, the fitness distance plot of a typical TSP instance is shown together with a fitness distance plot for a completely unstructured landscape of the quadratic assignment problem (QAP). The figure indicates that not all combinatorial problems have a 'nice' landscape as the TSP. The findings also provide an explanation of the success of many of the TSP heuristics. Many meta–heuristics implicitly exploit the fact that local optima are close to each other. An obviously reasonable strategy is to hop from one

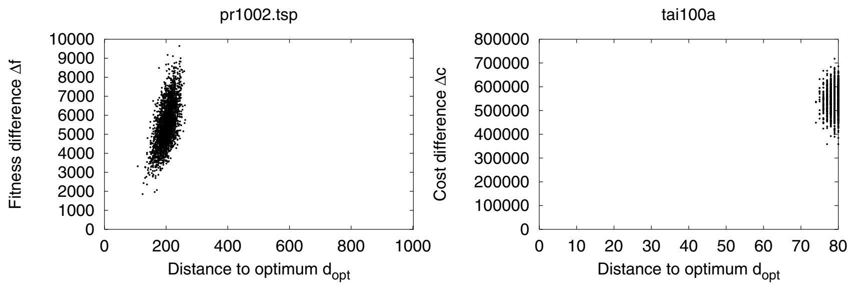

Fig. 7.6. FDC plots for a TSP instance (left) and a QAP instance (right).

local optimum to the next better one until the global optimum is reached. This is, in fact, what MAs and ILS do.

7.4.2 State-of-The-Art Meta-Heuristics for the TSP

The TSP has served as a test-bed for new heuristic approaches including evolutionary algorithms (EA) and memetic algorithms. Consequently, many approaches, both evolutionary and non-evolutionary, have been proposed. Here, we focus on those approaches which are highly effective and scalable. Euclidean TSP instances up to a size of 1,000 can be considered as trivial for most algorithms. In fact, these small problems can usually be solved exactly by Branch & Cut [24] in a few seconds. Therefore, these instances are no longer of interest for heuristics research on the TSP. For instances up to approx. 30,000 cities, very effective heuristics have been proposed most of which are based on the powerful Lin-Kernighan (LK) heuristic [524], a variable *k*-opt local search. An example is Helsgaun's LK implementation (LKH) [380].

Only few evolutionary algorithms can compete with LKH. One of the best evolutionary approaches is the EA of Nagata using EAX crossover [647] and 2-opt local search. This algorithm finds (near) optimal tours up to a size of 33,000 cities, although with a high runtime. Recently, Nguyen *et al.*[664] have proposed a memetic algorithm which utilizes a variant of the MPX crossover operator [637] and a Lin-Kernighan local search variant with 5-opt moves. Results are reported for instances up to 85,900 cities. The authors claim that their algorithm is more effective than LKH. Moreover, the authors describe an approach for solving the World TSP (approx. 2 million cities) by solving and merging subproblems. But results for other instances in the range from 100,000 to 10 million cities are not reported.

For instances larger than 100,000 cities, only few heuristics have been proposed. For these instances, the DIMACS TSP implementation challenge [428] lists several approaches of which the best are based on the LK heuristic: The multi-level algorithm of Walshaw [912] first reduces the size of a TSP instance stepwise and then applies the (chained) LK heuristic to the smaller problems. The results are inferior to the results obtained by directly applied chained LK or iterated LK heuristics. These heuristics are based on the principle of iterated local search [533], an evolutionary heuristic incorporating local search. The idea is to stepwise improve the current best solution by mutating it and subsequently applying local search. The first iterated local search was the iterated Lin-Kernighan (ILK) heuristic by Johnson [427]. Other variants have been proposed such as the chained Lin-Kernighan heuristic [25, 26]. These ILK heuristics have been applied to instances with up to 10 million cities. The only algorithm within the DIMACS challenge not using LK as a subroutine and still being highly effective for large instances is the dynamic programming approach of Balas and Simonetti [39].

Except for the LKH heuristic, none of the mentioned algorithms provides a lower bound on the optimum solution. To the best of our knowledge, the only evolutionary algorithm computing lower bounds is the one proposed in [556]. However, the approach deals with instances below 2,400 cities only.

7.4.2.1 An ILS Approach for Very Large TSP Instances

In [590] we have presented an iterated local search approach that simultaneously improves lower and upper bounds for a TSP instance to provide a gap for the best solution found. The gap determines the maximum deviation from the optimum solution and therefore provides a quality measure for the obtained TSP tour. The approach differs from exact algorithms like Branch & Cut [179] in that no efficient linear programming (LP) solver is required and it differs from approximation algorithms such as PTAS (polynomial time approximation scheme) [27] in that no general performance guarantee is provided. Instead, the quality is proved for each instance and a particular run: a final gap between lower and upper bound of 1% means that the solution found is at most one percent above the optimum (in practice the real gap is much lower).

Our local search is based on the LK heuristic, hence our iterated local search is called iterated LK. The general outline of our iterated LK is shown in Alg. 14. In contrast to other approaches, our ILK incorporates a lower bound computation.

Algorithm 14. The Advanced Iterated Link-Kernighan Heuristic

```
begin
    C ← FindInitialCandidateSet(Instance);
    Tour ← Init();
    Tour ← LocalSearch(C, Tour);
    C ← FindInitialLowerBound(C, TourLength(Tour));
    for iter ← 1 to MaxIter do
        Tbest ← Tour;
        Tour ← Mutate(Tour);
        Tour ← LocalSearch(Tour);
        if TourLength(Tour) < TourLength(Tbest) then Tbest ← Tour;
        if (iter mod 400) = 0 then C ← UpdateLowerBound(C, TourLength(Tbest));
    endfor
end
```

This computation is interleaved with the optimization algorithm as can be seen in the figure: every 400 iterations of the ILK, the lower bound is improved until there appears to be no more improvement possible (the lower bound computation has converged). The lower bound computation possibly modifies the candidate edge set, which is used by the local search to look for improving moves (edge exchanges).

To find initial solutions (*Init()* in the pseudo code), we use the Quick-Boruvka heuristic [26, 428], and the initial candidate set (*FindInitialCandidateSet(Instance)* in the pseudo code) is based on a subgraph containing the two nearest neighbors for each quadrant of a city [26]. This candidate set has the property of being connected.

The mutation operator used in the algorithm is non-sequencial four exchange [524, 588] using a random walk on the candidate set to find edges to be included in the tour. This operator has been proven to be very effective in conjunction with Lin-Kernighan local search [26]. The random walk on the candidate edge set assures that edges with a relatively small length instead of arbitrarily long edges are included.

As mentioned before, we use a variant of the original Lin-Kernighan heuristic for the local search. Compared to the original LK, we use 3-opt moves as submoves instead of 2-opt moves at all levels. We do not use backtracking which simplifies the implementation drastically without affecting the performance. In this aspect our implementation is similar to LKH.

In order to compare with other state-of-the-art approaches, Table 7.1 shows a comparison with the eleven best performing algorithms (out of 90) listed on the DIMACS TSP challenge web page. The summary was produced with the statistics

Table 7.1. Comparison of DIMACS TSP Challenge Results on E1M.0. ILK-PM-.1N denotes our ILK with 1 million iterations and ILK-PM-.1N denotes our ILK with 1,2 million iterations.

% HK	Seconds	Implementation	Reference
0.787	17544.0	ILK-PM-.12N	this paper
0.792	77161.6	ILK-NYYY-N	([663])
0.797	16062.0	ILK-PM-.1N	this paper
0.804	8694.0	ILK-PM-.1N without LB	this paper
0.841	6334.0	ILK-NYYY-Ng	([663])
0.879	42242.5	MLCLK-N	[912]
0.888	3480.2	ILK-NYYY-.5Ng	([663])
0.903	19182.7	BSDP-6	[39]
0.903	19503.1	BSDP-8	[39]
0.903	21358.3	BSDP-10	[39]
0.903	19108.1	CLK-ABCC-N.Sparc	[25]
0.905	19192.3	CLK-ACR-N	[26]
0.910	16008.0	CLK-ABCC-N.MIPS	[25]
0.945	20907.6	MLCLK-.5N	[912]

code from the challenge. Thus the running time reported in the table is normalized to a DEC Alpha processor with 500 MHz in order to allow a comparison of the different approaches. The quality is given as the percentage excess over the Held-Karp (HK) bound. As shown in the table, our algorithm provides a significantly better tour quality than the other approaches. And it does this in a fraction of time of the second best approach which is also an ILK implementation. Note that none of the competitors computes a lower bound. For the 10 million city instance E10M.0, the quality of our approach is 0.75% over the Held-Karp bound compared to the best algorithm of the DIMACS challenge which is 1.63% over the Held-Karp bound! This is due to the fact that the best algorithms for the smaller instances do not scale as well as our approach. While the runtime of our approach without lower bound computation grows linearly with the problem size, the runtime of the others clearly grows faster and and yields in the non-applicability of these algorithms to very large problem instances (>1 million) whereas our approach is still very successful even if the lower bound computation is activated.

More details on the algoritm as well as the results can be found in [590].

7.5 Case Study II: The BQP

In the *unconstrained binary quadratic programming problem* (BQP), a symmetric rational $n \times n$ matrix $Q = (q_{ij})$ is given, and a binary vector of length n is searched for, such that the quantity

$$f(x) = x^t\, Q\, x = \sum_{i=1}^{n} \sum_{j=1}^{n} q_{ij}\, x_i\, x_j, \quad x_i \in \{0,1\}\ \forall\, i = 1,\ldots,n \tag{7.9}$$

is maximized. This problem is also known as the *(unconstrained) quadratic bivalent programming problem*, *(unconstrained) quadratic zero–one programming problem*, or *(unconstrained) quadratic (pseudo-) boolean programming problem* [55]. The general BQP is known to be NP-hard but there are special cases that are solvable in polynomial time [55]. In [926], it has been shown that there are special cases of the BQP, which can be solved efficiently with simple EAs, but there are also cases, for which these EAs have been proven to be ineffective (exponentially growing running time).

The BQP has a large number of applications, for example in capital budgeting and financial analysis problems [506, 571], CAD problems [485], traffic message management problems [300], machine scheduling [10], and molecular conformation [722]. Furthermore, several other combinatorial optimization problems can be formulated as a BQP, such as the maximum cut problem, the maximum clique problem, the maximum vertex packing problem and the maximum independent set problem [414, 707, 708].

There is a close relation between binary quadratic programming and *NK*-landscapes: The objective function of the BQP can be decomposed into n functions.

The fitness of a BQP solution can thus be rewritten as a sum of functions for each site, called the fitness contributions f_i of site i in the genome:

$$f(x) = \sum_{i=1}^{n} f_i(x_i, x_{i_1}, \dots, x_{i_{k(i)}}), \tag{7.10}$$

$$f_i(x) = \sum_{j=1}^{n} q_{ij} x_i x_j. \tag{7.11}$$

Similar to the *NK*-landscapes defined in [452], the fitness contribution f_i of a site i depends on the gene value x_i and of $k(i)$ other genes $x_{i_1}, \dots, x_{i_{k(i)}}$. While for *NK*-landscapes $k(i) = K$ is constant for all i, in the BQP $k(i)$ is defined as the number of non-zero entries in the i-th column of matrix Q. Hence, the degree of epistasis in a BQP instance can be easily determined by calculating the density of the matrix Q. It is defined as the number of non-zero entries divided by the number of total entries in the matrix. Thus, the density is between 0 and 1, where 0 means no epistasis and 1 maximum epistasis (every gene depends on the values of all other genes).

7.5.1 Fitness Landscape

Since the BQP is binary-coded and local search for the BQP is based on the k-opt neighborhood as defined as

$$\mathcal{N}_{k\text{-}opt}(x) = \{x' \in X | d_H(x', x) \leqslant k\} \tag{7.12}$$

where d_H denotes the hamming distance between bit strings and X the set of all bit strings of length n ($X = \{0,1\}^n$), the landscape considered in the search space analysis of the BQP is $\mathcal{L} = (X, f, d_H)$. The graph of this landscape is a hypercube of dimension n in which the nodes represent the (candidate) solutions of the problem. An edge in the graph connects neighboring points in the landscape, i.e. points that have hamming distance 1.

7.5.1.1 Autocorelation Analysis

Since there are no theoretical results on the autocorrelation function or the random walk correlation function for the BQP, experiments have been conducted in [591] to estimate the correlation length of selected landscapes. The instances were taken from ORLIB [54] and [18, 318]. Here, we summarize the findings: Considering all selected instances, the quotient of n/ℓ varies in tight bounds: the lowest value for n/ℓ is 2.36 and the highest is 2.71. Compared to *NK*-landscapes, this is fairly low since in the *NK*-model $n/\ell \approx K + 1$. For the instances denoted glov500, the values are very similar (2.67 ± 0.04) and thus remain constant independent of the density of the problem. For the set kb-g, the values for n/ℓ do change with the density of Q, but the values become smaller with increasing density. This is surprising, since in the *NK*-model, the correlation length decreases with increasing epistasis, and the density can be regarded as a measure of epistasis in the BQP. For the set of instances of size 2500 and a density of 0.1, the values for n/ℓ are constant (about 2.66).

Summarizing, all the instances of the BQP considered here have got a smooth landscape similar to *NK*-landscapes with $K \leqslant 3$.

7.5.1.2 Fitness Distance Correlation Analysis

In a FDC analysis, we studied the correlation of fitness (objective f(x)) and distance to the optimum for local optima with respect to *1-opt* local search. The findings can be summarized as follows. In most cases, the average distance between the local optima and the average distance to the global optimum (best-known solution) are very similar. Moreover, the local optima are located in a small region of the search space: the average distance between the local optima is between a fourth and sixth of the maximum distance (the diameter) between two solutions in the search space. For set glov500, the average distance to the optimum is a sixth of the diameter independent of the density of Q. For set beas2500 the local optima are even closer to the optimum in relation to the maximum distance of two solutions in the landscape: the average distance to other local optima is more than a seventh of the diameter of the landscape. The FDC coefficient varies from -0.56 to -0.81 excluding glov500-4. The FDC coefficient for this instance is -0.31.

In Figure 7.7, some scatter plots are provided in which the distance to the global optimum is plotted against the fitness difference $\Delta f = f(x_{opt}) - f(x_{loc})$ for each local optimum found. The figure indicates that the local optima are even closer to each other than for smooth *NK*-landscapes with $K = 2$, revealing the deep valley/massiv central property.

7.5.1.3 Advanced Fitness Landscape Analysis

In order to analyze the structure of the basins of attraction more closely, we conducted several experiments for the BQP [583]. For each problem instance, 1000 local optima were generated and mutated 100 times with a specified mutation rate.

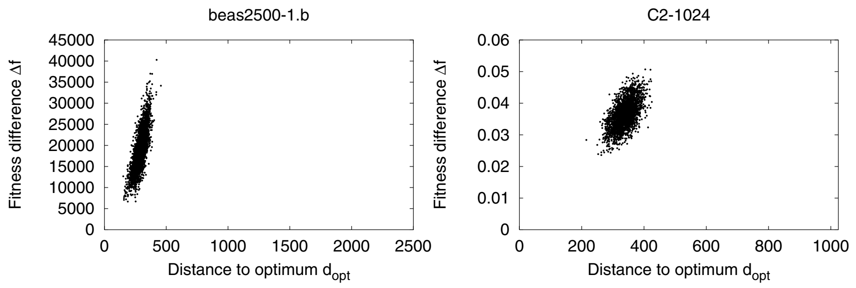

Fig. 7.7. 1-opt Local Optima FDC plots for a BQP instance (left), an *NK*-landscape with $K = 2$ (right)

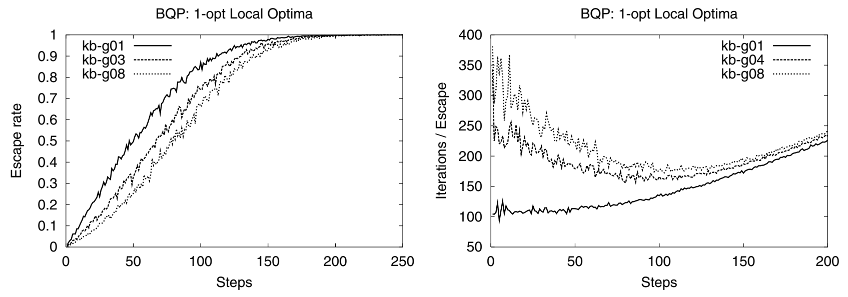

Fig. 7.8. Basins of Attractions of Local Optima: The escape rate (left) and LS iterations/escape

The number of mutation steps was increased from 1 to $n/2$ where n denotes the problem size. Since both problems are binary coded, the number of mutation steps is defined as the number of bit-mutations executed by the mutation operator.

The BQP instances from [319] denoted kb-g01, kb-g02, ... , kb-g10, where the number indicates the density of matrix Q (01 means density 0.1 and 10 denotes density 1.0), were used in the experiments. All instances have a problem size of $n = 1000$. In all cases, 1-opt local search (single bit-flip neighborhood) was used with the best improvement strategy [582, 589]. Selected results are presented in Fig. 7.8 (right). The question arises whether there is an optimum mutation rate in terms of computation costs. At which mutation rate is the number of visited local optima per time unit maximum? The answer is given in Fig. 7.8 (left). In the figure, the number of local search iterations per escape is displayed over the number of mutation steps. For densities greater 0.1, the optimum is around hundred mutation steps and the optimum approaches 2 steps for density 0.1.

To investigate the properties of random walks starting at local optima we conducted several experiments on the problem instances mentioned above. During the run of a memetic algorithm, random walks were performed by selecting two parents A and B and performing a random walk from A to B to simulate recombination as well as a walk with the same length starting at A in an arbitrary direction to simulate mutation. Fig. 7.9 shows the results of the random walk analysis for the BQP on selected instances. Directed random walks have a much higher average fitness than undirected random walks. As the right of the figure indicates, the fitness difference is always positive, independent of the distance between the start and end-points of the random walks. During the whole run of the MA, recombination is clearly superior to mutation since the random walks from one local optimum to the other produce solutions with much higher fitness than random walks starting at the same local optima but in arbitrary direction.

As the FDC analysis reveals, BQP landscapes are structured, even with a high density of matrix Q. Hence, recombination appears to be superior to mutation as the random walk correlation analysis indicates. However, in [591] it has been shown that simple recombination schemes are not very effective in MAs for the BQP. The

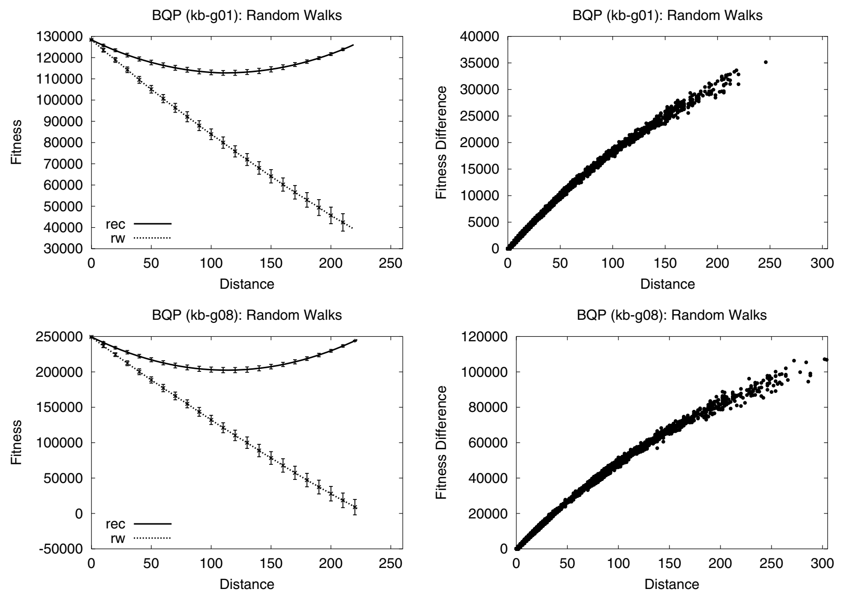

Fig. 7.9. Random Walks Starting from Local Optima of the BQP

reasons become obvious considering the results of the local search escape analysis: the local optima in the BQP have very large basins of attractions leading to the fact that after recombination and local search, one of the parent local optima is rediscovered very often. As a consequence, additional techniques are required to prevent this from happening, as shown in [591].

7.5.2 State-of-the-Art Meta-Heuristics for the BQP

Several (meta–)heuristic approaches have been proposed for the BQP. In the following, we briefly review effective heuristic algorithms capable of finding optimum/best-known or very good near-optimum solutions for the BQP.

Glover, Kochenberger, and Alidaee [319] have proposed a tabu search heuristic for instances of up to 500 variables. Their method consists of a strategic oscillation scheme that alternates between constructive and destructive phases.

Lodi, Allemand, and Liebling [532] proposed a heuristic based on an evolutionary algorithm for the same problem set studied by Glover *et al.* Their heuristic is combined with the local search algorithm that is based on the constructive and destructive phases of the tabu search in [319]. Their crossover operator is similar to uniform crossover [864], utilizing the MinRange algorithm, which is based on the property by Pardalos and Rodgers [706].

Alkhamis, Hasan, and Ahmed proposed a simulated annealing algorithm [11]. Unfortunately, only small problem instances with up to 100 variables were investigated. In [55], Beasley has provided larger BQP test problems with up to 2500 variables as new test problems of the ORLIB [54]. Beasley includes the best-known solutions for each of the provided instances found by tabu search and simulated annealing.

In our genetic local search algorithm [586], a simple local search (called *1-opt*, see below) and a variant of uniform crossover, HUX [246], were employed. For several large instances of [55], they provided new best-known solutions and have shown that their algorithm outperforms the two alternative heuristics reported by Beasley. Furthermore, we developed a greedy heuristic and two local search heuristics called *1-opt* and *k-opt* [589]. We showed that in particular the *k-opt* local search is capable of finding high-quality solutions even for the large-scale problem instances, and they also proposed that these heuristics are well suited as components for metaheuristics, such as MA.

Katayama and Narihisa [446] proposed a new simulated annealing-based heuristic with a reannealing process. The approach was tested on the large instances ranging from 500 to 2500 variables contained in the ORLIB. Although simulated annealing is based on the simple *1-opt* neighborhood structure, better average solution results for the large instances were found as compared to the other heuristics: our genetic local search *et al.* [586] and the heuristics by Beasley [55]. Moreover, the heuristic was considerably faster than the others, and new best-known solutions for several large instances were reported.

7.5.3 A Memetic Algorithm Using Innovative Recombination

In [591], we proposed a memetic algorithm using a new recombination operator that takes the properties of the search space of the BQP into account. Although the landscape is correlated/structured, recombination operators such as HUX or simple uniform crossover are not that effective due to the large basins of attraction of the local optima, as stated above. This is even more true when a powerful variable k-opt local search is used.

The outline of the MA is provided in Alg. 15. The population is initialized (Init()) with the randomized greedy heuristic we proposed in [589]. The local search used is a randomized k-opt local search algorithm proposed in [447, 448], which is based on the k-opt local search proposed in [589]. Similar to the Lin-Kernighan algorithm for the TSP [524], the basic idea of the heuristic is to find a solution by flipping a variable number of k bits in the solution vector per iteration. In each step, a sequence of n solutions is generated by flipping a random bit with positive gain or the bit with the highest associated gain. Furthermore, a candidate set is used to assure that each bit is flipped no more than once. The best solution in the sequence is accepted as the new solution for the next iteration.

To minimize the number of times a local optimum is rediscovered, we have proposed a new variation operator. The basic idea is to utilize a simple local search for introducing new alleles, i.e. alleles not contained in both parents. The crossover can be regarded as innovative, since new alleles are introduced based on the associated gain in fitness. Hence the name *innovative variation*. The operator works as follows:

Algorithm 15. BQP-MA

```
begin
    foreach S in Population do S ← LocalSearch(Init());
    while not terminated do
        Offspring ← {};
        for i ← 0 to crossovers do
            A ← Select(Population);
            B ← Select(Population);
            C ← LocalSearch(Recombine(A, B));
            Offspring ← OffSpring + C;
        endfor
        Population ← Select(Population, Offspring);
        if Converged(Population) then
            foreach S in Population\Best do S ← LocalSearch(Mutate(S));
        endif
    endw
end
```

In the first step, the common and the non-common bits of the parents are identified. Then, the contents of parent I_a are copied to the offspring. Variation is now achieved by alternately flipping non-common bits and common-bits: In a loop, a randomly selected non-common bit is flipped with a positive associated gain, if there is at least one such non-common bit. The common bit with the maximum associated gain is flipped afterwards, even if the gain is negative. If a bit has been flipped, it is removed from the set it was contained in (either the common or non-common bit set). The loop is repeated n times where n is the number of non-common bits.

Mutation is only applied when the population is converged. In such a case, we perform a diversification/restart strategy, which is borrowed from [246], in order diversity the population by moving to other points of the search space if no new best individual in the population was found for more than 30 generations. In response to this requirement, the individuals except for the best one in the population are mutated by flipping randomly chosen $n/3$ bits for each individual of length n. After that, each of the mutated individuals is improved by the randomized *k-opt* local search to obtain a renewal set of local optima and the search is started again with the new, diverse population. The performance of our MA is shown in Table 7.2. We have tested our algorithm on several benchmark instances from the literature. The first set kb-g consists of 10 instances of size $n = 1000$ that have been provided by Kochenberger and have been used in the performance evaluation of scatter search [18]. The densities of instances in the problem set are between 0.1 and 1.0. The last two sets beas1000 ($n = 1000$) and beas2500 ($n = 2500$) were first studied by Beasley [55], each of which consists of ten instances with $dens(Q) = 0.1$.

In [591], a detailed comparison with other approaches for the BQP has been made. Summarizing, The MA approach provides a higher average solution quality than other approaches. CPU times have not been reported for all approaches or are not directly comparable to our results. However, our MA is superior or at least

Table 7.2. Computational results of the MA with innovative variation incorporating the randomized *k-opt* local search algorithm for test problem instances from the literature.

		best	MA with Innovative Variation			
Instance	$dens(Q)$	known	best	avg (quality %)	b/30	t1/s (gens)
kb-g01	0.1	131456	131456	131456.0 (0.000000)	30	6.1 (6)
kb-g02	0.2	172788	172788	172788.0 (0.000000)	30	12.8 (9)
kb-g03	0.3	192565	192565	192565.0 (0.000000)	30	11.4 (4)
kb-g04	0.4	215679	215679	215679.0 (0.000000)	30	42.0 (23)
kb-g05	0.5	242367	242367	242367.0 (0.000000)	30	15.6 (5)
kb-g06	0.6	243293	243293	243293.0 (0.000000)	30	69.4 (30)
kb-g07	0.7	253590	253590	253590.0 (0.000000)	30	45.7 (13)
kb-g08	0.8	264268	264268	264268.0 (0.000000)	30	40.2 (12)
kb-g09	0.9	262658	262658	262618.0 (0.015219)	25	140.1 (40)
kb-g10	1.0	274375	274375	274335.4 (0.014423)	15	143.9 (41)
beas1000-1	0.1	371438	371438	371438.0 (0.000000)	30	6.7 (9)
beas1000-2	0.1	354932	354932	354932.0 (0.000000)	30	7.7 (10)
beas1000-3	0.1	371236	371236	371236.0 (0.000000)	30	5.8 (5)
beas1000-4	0.1	370675	370675	370675.0 (0.000000)	30	6.6 (7)
beas1000-5	0.1	352760	352760	352760.0 (0.000000)	30	11.8 (20)
beas1000-6	0.1	359629	359629	359629.0 (0.000000)	30	11.0 (17)
beas1000-7	0.1	371193	371193	371193.0 (0.000000)	30	9.1 (12)
beas1000-8	0.1	351994	351994	351994.0 (0.000000)	30	28.1 (50)
beas1000-9	0.1	349337	349337	349337.0 (0.000000)	30	6.1 (6)
beas1000-10	0.1	351415	351415	351415.0 (0.000000)	30	7.3 (9)
beas2500-1	0.1	1515944	1515944	1515944.0 (0.000000)	30	59.9 (15)
beas2500-2	0.1	1471392	1471392	1471357.8 (0.002322)	26	165.2 (60)
beas2500-3	0.1	1414192	1414192	1414183.1 (0.000629)	29	87.8 (30)
beas2500-4	0.1	1507701	1507701	1507701.0 (0.000000)	30	42.2 (9)
beas2500-5	0.1	1491816	1491816	1491816.0 (0.000000)	30	76.1 (29)
beas2500-6	0.1	1469162	1469162	1469162.0 (0.000000)	30	78.5 (26)
beas2500-7	0.1	1479040	1479040	1479040.0 (0.000000)	30	92.2 (30)
beas2500-8	0.1	1484199	1484199	1484199.0 (0.000000)	30	47.1 (10)
beas2500-9	0.1	1482413	1482413	1482413.0 (0.000000)	30	140.0 (70)
beas2500-10	0.1	1483355	1483355	1483336.9 (0.001218)	28	108.6 (43)

comparable to other state of the art approaches including tabu search and scatter search. In [700], a multi-start tabu search has been proposed that appears to be similarly effective. Again, a direct comparison is not easy. The author, however, fails to compare with the results from [591]. Instead, older results are considered from [586].

7.6 Conclusion

In this chapter, we have discussed fitness landscape analysis as suitable methodology for discovering search space properties relevant for the development of memetic algorithms. We have shown that some combinatorial optimization problems have structured landscapes that have a deep value/massive central structure. This structure is known to be a reason why MAs perform well on certain combinatorial problems. We have argued that the results from the fitness landscape analysis can be used to design local search (autocorrelation analysis) or can help in deciding to use mutation based or recombination variation (fitness distance analysis). In two case studies, we have demonstrated that MAs are highly effective and belong to state-of-the-art meta-heuristics. For the binary quadratic programming problem, we have shown that an advanced fitness analysis can help in designing a recombination operator by assuring that it has a high chance to leave the basin of attraction of the current local optima. This innovative variation operator increases the effectiveness of the MA considerably.