Memetic Algorithms for the Traveling Salesman Problem

Article *in* Complex Systems · July 2002

Source: CiteSeer

2 authors, including:

Peter Merz
Hochschule Hannover
84 PUBLICATIONS **3,432** CITATIONS

SEE PROFILE

Memetic Algorithms for the Traveling Salesman Problem

Peter Merz[*]

Department of Computer Science,
University of Tübingen,
Sand 1, D–72076 Tübingen, Germany

Bernd Freisleben[†]

Department of Mathematics and Computer Science,
University of Marburg,
Hans-Meerwein-Straße,
D–35032 Marburg, Germany

Memetic algorithms (MAs) have been shown to be very effective in finding near-optimum solutions to hard combinatorial optimization problems. In this paper, the fitness landscapes of several instances of the traveling salesman problem (TSP) are investigated to illustrate why MAs are well-suited for finding near-optimum tours for the TSP. It is shown that recombination-based MAs can exploit the correlation structure of the landscape. A comparison of several recombination operators—including a new generic recombination operator—reveals that when using the sophisticated Lin–Kernighan local search, the performance difference of the MAs is small. However, the most important property of effective recombination operators is shown to be respectfulness.

In experiments it is shown that our MAs with generic recombination are among the best evolutionary algorithms for the TSP. In particular, optimum solutions could be found up to a problem size of 3795, and for larger instances up to 85,900 cities, near-optimum solutions could be found in a reasonable amount of time.

1. Introduction

The traveling salesman problem (TSP) is one of the best-known combinatorial optimization problems. It can be stated as follows: Given n cities and the geographical distance between all pairs of these cities, the task is to find the shortest closed tour in which each city is visited

[*]Electronic mail address: peter.merz@ieee.org.
[†]Electronic mail address: freisleb@informatik.uni-marburg.de.

exactly once. More formally, the tour length

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

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 step i.

In recent years, the exact solution of large TSP instances has made enormous progress due to the improvement of branch and cut algorithms. Furthermore, the TSP has been widely used as a problem for testing new heuristic algorithms and general purpose optimization techniques. As a result, highly effective heuristics have been proposed that are capable of solving TSPs with thousands of cities.

In this paper, memetic algorithms (MAs) [79] for the TSP are introduced which have been shown to belong to the best heuristics currently available for the TSP. These algorithms are similar to evolutionary algorithms, but have more in common with cultural than biological evolution. The MAs considered in this paper are hybrid evolutionary algorithms incorporating local search.

Firstly, a landscape analysis is performed to identify properties of TSP instances which can be exploited by MAs. It will be shown that although all TSP instances share certain characteristics, there are some landscapes that differ significantly from others, leading to a different performance of heuristic approaches. However, the analysis reveals that respectful recombination is capable of exploiting the distribution of local minima in the TSP landscape.

Secondly, a new generic greedy recombination operator is proposed and used to identify important properties of recombination operators in MAs for the TSP. Various recombination operators are compared in experiments and it is shown that many operators show similar performance, when a sophisticated local search heuristic is employed. On the other hand, it is shown—as expected due to the results of the landscape analysis—that recombination needs to be respectful to be highly effective.

Finally, it is demonstrated that the successor of the already published MA [35, 71, 72] is capable of finding optimum solutions for problems up to 3795 cities in a small amount of time and is thus superior to any other evolutionary algorithm for the TSP known to the authors. In additional experiments it is shown that with this new approach problems of up to 85,900 cities can be tackled.

The paper is organized as follows. In section 2, simple construction heuristics and improvement heuristics for the TSP are described. In section 3, a fitness landscape analysis is performed for the TSP and

several types of TSP instances are discussed. Section 4 presents a MA for solving combinatorial optimization problems in general terms, and the evolutionary operators and MA components specially designed to solve the TSP. The results of the MA with different evolutionary operators for selected instances is provided and a comparison of the MA with the iterated Lin–Kernighan approach is conducted on large instances of up to 85,900 cities. Section 5 concludes the paper and outlines areas for future research.

2. Heuristics for the traveling salesman problem

For decades, the TSP has served as an initial proving ground for new ideas to solve combinatorial optimization problems. Besides the fast development in solving TSP instances to optimality, enormous progress has been made in the field of heuristics.

Most of the earliest algorithms belong to the class of construction heuristics. Examples of this class are nearest neighbor heuristics and insertion heuristics, for which a detailed description and comparison can be found in [53, 97]. Another intuitive approach is the greedy heuristic, also known as the multi-fragment heuristic [5, 53]. Furthermore, there are more elaborate tour construction heuristics, such as the Christofides algorithm [16] which is based on spanning trees, or the savings heuristic (also known as the Clarke and Wright algorithm) originally developed for the vehicle routing problem [17]. However, these heuristics perform poorly in comparison to local search heuristics which belong to the class of improvement heuristics. But, instead of applying a local search to randomly generated solutions, a local search can be applied to solutions generated by a (randomized) tour construction heuristic. Surprisingly, the best performing construction heuristic is not the best suited for combining with local search, as shown by several researchers independently [7, 53, 97]. For example, in [53] it is shown that although the savings heuristics performs better than the greedy and nearest neighbor heuristics, in combination with 2-opt or 3-opt local search it performs the worst (even worse than the local search applied to random solutions). In fact, the best suited construction heuristic is the greedy heuristic, as shown in [7, 53]. It appears that the starting tour for a local optimization must contain a number of exploitable defects, that is, some rather long edges, and if a tour is too good it may not have these.

Since greedy and local search heuristics are among the most efficient algorithms for the TSP with short running times and thus the most interesting for incorporation into MA, these two types of algorithms are described in the following paragraphs in more detail. Many other heuristics have been proposed for the TSP, including simulated annealing [50, 111], tabu search [28], ant colonies [22, 36, 106], artificial neural networks [23, 76, 92], search space smoothing [43], perturba-

tion [18], and evolutionary algorithms [30, 32, 49, 85, 108, 110, 112, 114].

2.1 The greedy heuristic

Although the nearest neighbor heuristic can be regarded as a greedy heuristic, the term is usually used for the following variant of the greedy algorithm.

This heuristic can be viewed as considering the edges of the graph in increasing order of length, and adding any edge that will not make it impossible to complete a tour. Thus, the algorithm builds up a TSP tour for N cities (a cycle of length N in a graph) by adding one edge at a time, starting with the shortest edge, and repeatedly adding the shortest remaining available edge. In the algorithm, an edge is referred to as available if it is not yet in the tour and if adding it would not create a degree-3 vertex or a cycle of length less than N.

While in the nearest neighbor heuristic partial tours maintain a single TSP fragment, the greedy heuristic employs a set of fragments. Therefore, the greedy heuristic is also known under the name multi-fragment heuristic [7].

The implementation sketched above requires $O(N^2 \log N)$ time. However, using appropriate data structures, the running time of the algorithm can be reduced considerably. As shown in [7], using K-d trees to calculate nearest neighbors [6], and using a priority queue to store available candidate edges, the expected running time is reduced to $O(N \log N)$ for uniform data (euclidean TSP with points uniformly distributed in the unit square).

2.2 Local search

Local search algorithms for the TSP are based on simple tour modifications. A local search algorithm is specified in terms of a class of operations called moves that can be used to transform one tour to another. We can view local search as a neighborhood search process where each tour has an associated neighborhood of tours, that is, those that can be reached by a single move. The search algorithm repeatedly moves to a better neighbor until no better neighbors exist. Moves proposed for the TSP can be divided into node exchange operators, node insertion operators, and edge exchange operators.

Viewing a TSP tour as a sequence of cities which defines the order in which to visit the cities, the node exchange operator simply exchanges two nodes in the sequence.

Node re-insertion operators work by deleting a node from a tour and inserting it at another position in the tour. Variations of this scheme exist in which two nodes are re-inserted (edge insertion) [97] or up to three nodes are re-inserted (Or-opt) [90].

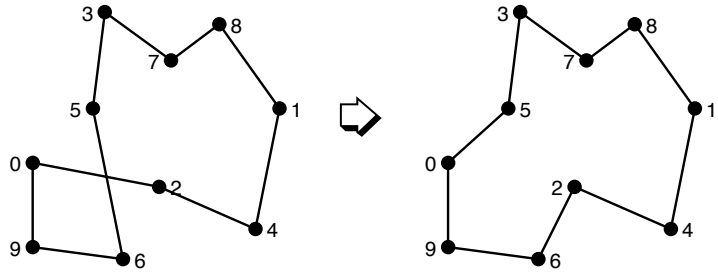

Figure 1. Neighborhood search by exchange of two edges (2-opt).

2.2.1 2-opt and 3-opt local search

Among simple local search algorithms, the most famous are 2-opt and 3-opt [65] which are examples of edge exchange algorithms. The 2-opt algorithm was first proposed by Croes in [19], although the basic move had already been suggested by Flood in [29]. This move deletes two edges, thus breaking the tour into two paths, then reconnects those paths in the other possible way as shown in Figure 1. In the example, the edges $(0, 2)$ and $(5, 6)$ are exchanged with the edges $(0, 5)$ and $(2, 6)$.

Analogously, in 3-opt, up to three edges are exchanged. With 3-opt, the neighborhood size increases from $O(N^2)$ to $O(N^3)$ compared to the other local searches. In practice, however, the running time for searching the neighborhood can be reduced for euclidean instances so that 3-opt is applicable even to large instances. For example, nearest neighbor lists are used to restrict the number of candidate edges for the replacement of edges in an exchange [53, 97]. Furthermore, the concept of "don't look bits" proposed by Bentley in [5] reduces the search time for an improving move considerably. Compared to other neighborhood search algorithms, the first improving move is accepted in this scheme rather than the best.

It has been shown that edge exchange local search is much more effective than node re-insertion or node exchange [97]. Generally, the higher the k of a k-opt local search, the better the resulting tours. However, since the neighborhood size grows exponentially with k, only small k turn out to be practical. Lin and Kernighan have shown that a subset of the k-opt neighborhood can be searched very efficiently with a considerable decrease in tour length compared to 2- or 3-opt.

2.2.2 The Lin–Kernighan algorithm

For over a decade and a half, from 1973 to about 1989, the world champion heuristic for the TSP was generally recognized to be the local search algorithm of Lin and Kernighan (LK) [66]. This algorithm is both

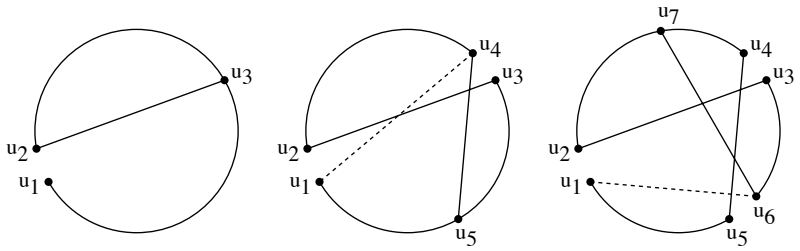

Figure 2. An edge exchange in the LK heuristic.

a generalization of 3-opt and an outgrowth of ideas the same authors had previously applied to the graph partitioning problem [59].

The basic idea of the LK algorithm is to build up complex moves by combining simple submoves to exchange a variable number of edges. The submoves usually employed are 2-opt and 3-opt moves although variants exist where node re-insertion and 2-opt has been used [97]. To illustrate the behavior of the heuristic, an example of an edge exchange is shown in Figure 2. (In the figure, a TSP tour is displayed as a circle and the length of the edges do not resemble their length in the TSP graph.) Briefly, the LK heuristic can be described as follows. In each step, we have a situation where the tour is broken up at one node forming a 1-tree (a spanning tree with an extra edge) as shown on the left of the figure. This 1-tree can be easily transformed into a feasible TSP tour by breaking up one edge of the degree-3 vertex and connecting the two degree-1 vertices. Consider now the example in which an improving k-exchange is searched beginning with node u_1. First, the edge (u_1, u_2) is replaced by a shorter edge (u_2, u_3). Now, the algorithm considers closing up a tour by connecting the predecessor of u_3 called u_4 with u_1 and thus replacing edge (u_3, u_4) with edge (u_4, u_1). In this case, we made a 2-change since we replaced the edges (u_1, u_2) and (u_4, u_3) with (u_2, u_3) and (u_4, u_1). Alternatively, we can replace the edge (u_3, u_4) with (u_4, u_5) resulting in a new 1-tree. Once again, we may close up a tour by connecting u_6 with u_1 or continue searching by connecting u_6 to another node u_7 as shown in the right of the figure. Thus, the heuristic performs sequential changes of edges until no further exchanges are possible or favorable to find the best k-change in an iteration. The number of exchanges that are tried is bound by the gain criterion which is fulfilled if the gain of replacing k edges with new edges without closing up the tour is above zero. The change made in an iteration is the one with the highest gain when closing up the tour. If the search for an improving k-change fails, several levels of backtracking are considered. For example, alternatives for (u_2, u_3) at the first level and alternatives for (u_4, u_5) at the second level are considered. A more detailed description of the LK

algorithm would go beyond the scope of this paper and can be found in the original paper by Lin and Kernighan [66].

A major drawback of the LK heuristic besides the high effort needed for its implementation is its rather long running time. Therefore, several improvements to the original algorithm have been made, such as candidate lists based on nearest neighbors and don't look bits [53]. Furthermore, efficient data structures have been proposed to perform the submoves since they consume most of the running time of the algorithm, especially for large TSP instances ($N > 1000$) [33, 68].

2.3 Evolutionary algorithms

Inspired by the power of natural evolution, several researchers independently studied evolutionary algorithms (EAs) keeping in mind the idea that engineering problems could be solved by simulating natural evolution processes. Several EAs; for example, evolution strategies (ES), evolutionary programming (EP), and genetic algorithms (GAs), have been proposed since the early 1960s in which a population of candidate solutions is evolved subject to replication, variation, and selection.

In the past few years there has been an enormous amount of research in evolutionary computation with increasing interaction among the researchers of the various methods. The boundaries between GAs, EP, and ES have been broken down to some extent, and EAs have been developed that combine the advantages of the approaches. The field for applications of EAs has been drastically extended, including the evolution of computer programs known under the name genetic programming [61, 62], or the implementation of machine learning in classifier systems [12, 47]. Other extensions to the basic concepts have been made such as co-evolution [46, 91] or the hybridization of traditional problem-specific methods with EAs [21, 75].

2.3.1 Outline of evolutionary algorithms

Without referring to a particular algorithm, a general template of an EA is shown in Figure 3. All mentioned variants of EAs (GAs, EP, and ES) are special cases of this scheme. First, an initial population is created randomly, usually with no fitness or structural bias. Then, in the main loop, a temporary population is selected from the current population utilizing a selection strategy. Afterwards, the evolutionary operators mutation and/or recombination are applied to some or all members (individuals) of the temporary population. Usually, the main loop is repeated until a termination criterion is fulfilled (a time limit is reached or the number of generations evolved exceeds a predefined limit). The newly created individuals are evaluated by calculating their fitness. Before a new generation is processed, the new population is selected from the old and the temporary population. Now, the algorithm can continue by building a new temporary population. Besides the way

```
procedure EA;

    begin
        t := 0;
        initializePopulation(P(0));
        evaluate(P(0));
        repeat
            P' := selectForVariation(P(t));
            recombine(P');
            mutate(P');
            evaluate(P');
            P(t + 1) := selectForSurvival(P(t), P');
            t := t + 1;
        until terminate = true;
    end;
```

Figure 3. The EA pseudo code.

the methods encode the candidate solutions of the problem to solve, they differ in the order and rate in which the variation operators are applied and in the type of selection strategy they use.

2.3.2 Evolutionary algorithms for the traveling salesman problem

Various attempts have been made to apply EAs to the TSP. For example, evolutionary programming has been applied to the TSP by Fogel using node re-insertion as the mutation operator [31] and random 2-opt moves (random exchanges of two edges) [30]. Evolution strategies have been applied to the TSP by Herdy in [45] and Rudolph in [100]. While Herdy conducted experiments with node exchange, node re-insertion, and the edge exchange operator (two and three edges), Rudolph chose a real vector representation for the TSP and applied the ES on continuous variables. The majority of publications, however, deal with representations and/or recombination operators for GAs for the TSP.

Besides the most commonly used path representation [20, 37, 38, 89] in which a tour is coded as a vector of discrete variables of length N that provides the order in which to visit the cities and is thus a permutation π of the set $\{1, \ldots N\}$, other representations have been proposed such as the adjacency representation [41], the adjacency matrix representation [49], the precedence matrix representation [32], the ordinal representation [41], and the edge list representation in combination with the path representation [114].

There is an enormous number of recombination operators for the TSP, most of which have the disadvantage that they do not scale well or they are only effective in combination with additional heuristic operators. The reason will be illustrated by an example.

The partially-mapped crossover (PMX) was introduced by Goldberg and Lingle in [37]. It performs a sequence of swapping operations to create offspring. Firstly, a mapping section is chosen at random. In the example below, the mapping section is marked by "|":

Partially-mapped crossover												
Parent A	0	9	6	|	5	3	7	8	|	1	4	2
Parent B	0	5	3	|	7	4	1	8	|	2	6	9
Offspring A'	0	9	6	|	7	4	1	8	|	5	3	2
Offspring B'	0	1	4	|	5	3	7	8	|	2	6	9

Secondly, in parent A, cities 5 and 7 are swapped, then 3 and 4, and at last cities 5 and 1. Now, the mapping section is equal to the mapping section of parent B. Thirdly, parent B has to be transformed analogously by a sequence of swaps, these are: 5 and 7, 3 and 4, 6 and 5, and 1 and 7. The resulting offspring are shown above. Both offspring A' and B' contain three and two edges not shared by the parents, respectively. For example, offspring A' is a feasible solution but does not consist entirely of genetic material from its parents: the edges $(6, 7)$, $(5, 8)$, and $(2, 3)$ are not contained in either of the parents. Figure 4 displays the tours of parent A, parent B, and offspring A'. The tour lengths are 1707.96, 1834.27, and 2251.00, for parent A, parent B, and child A', respectively. Here, the crossover has disruptive effects: although only three edges are included that are not contained in the parents, the tour length is considerably longer than the lengths of the parent tours. Note that the edges of the parents in the example above can be recombined to the solution of length 1507.94.

The introduction of foreign edges into the child is referred to as implicit mutation and has a high impact on the effectiveness of recombination operators. If the number of foreign edges gets too high, the GA degrades to pure random search. But even a small number of foreign edges can prevent a GA from finding (near) optimum solutions, since these edges can be arbitrarily long and thus may have a large impact on

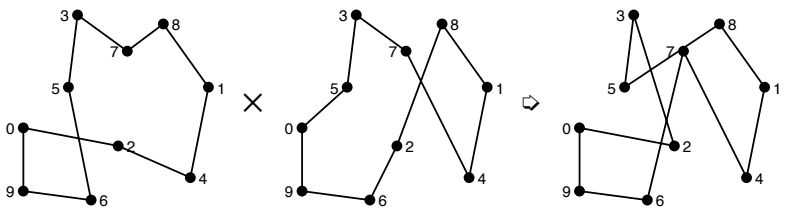

Figure 4. Crossover of TSP tours using PMX.

the objective value. In other words, the objective values of parents and offspring may not be highly correlated.

The phenomenon of implicit mutation during recombination can be observed by almost all recombination operators for the TSP. In [114], Whitley *et al.* argue that it is essential to focus attention on edges rather than preserving the positions of nodes. They developed the edge recombination operator which is aimed at preserving as many edges from the parents as possible while keeping the recombination process simple. Several variants of the edge recombination operator have been proposed [24, 69, 105], none of which guarantees that no implicit mutation occurs.

Grefenstette concludes from his studies [42]:

> Finally, it's widely recognized that GAs are not well suited to performing finely tuned local search... Once the high performance regions of the search space are identified by the GA, it may be useful to invoke a local search routine to optimize the members of the final population.

As a consequence, many researchers incorporated greedy choices into their recombination operators and/or used a local improvement technique to achieve better results [42, 51, 64, 107]. The use of local search after the application of a recombination operator can compensate for the disruptive effects of implicit mutations. In some cases, implicit mutations have a positive effect on the performance of the local search, and in some situations they do not. Thus, it is important that implicit mutations can be controlled in some way. Besides the number of foreign edges introduced during recombination, another aspect appears to be important: which edges are inherited from parents and which are not. More formally, recombination operators can be classified according to Radcliffe and Surry [93, 94] as follows.

Respectful
: The alleles that are identical in both parents are preserved in the offspring, that is, all edges found in both parent tours (common edges) are found in the offspring tour.

Assorting
: The offspring contain only alleles from either one of the parents, that is, all edges in the child tour are found in at least one of the parent tours, thus no implicit mutation occurs.

While respectful recombination can be easily achieved by a recombination operator for the TSP, assorting recombination is hardly accomplished. Note that for binary representations a respectful recombination is also assorting.

Beginning with Brady in [13], many researchers have made consequent use of local search in their EAs for the TSP. These hybrid EAs

are called memetic algorithms (MAs) [79, 84]. They differ from other hybrid evolutionary approaches in that all individuals in the population are local optima, since after each mutation or recombination, a local search is applied. In the following, some of these approaches are briefly described. They have been shown to be among the best heuristic techniques for the TSP developed so far.

2.3.3 Memetic algorithms for the traveling salesman problem

One of the earliest evolutionary approaches for the TSP using local search is the EA in [13]. In this approach, the solutions produced by crossover are optimized with a local search called quenching since it can be regarded as a zero temperature simulated annealing. In this MA, TSP tours are encoded using the path representation. A subpath in one parent is sought that has a corresponding subpath in the other parent containing the same cities. If the subpath is shorter than the corresponding subpath, this subpath in the other parent is replaced:

Brady's crossover												
Parent A	4	2	0	|	8	9	5	3	7	|	6	1
Parent B	2	|	9	8	3	7	5	|	0	1	6	4
Offspring A'	4	2	0	|	9	8	3	7	5	|	6	1

In the above example, the sum $d_{89} + d_{95} + d_{53} + d_{37}$ is greater than the sum $d_{98} + d_{83} + d_{37} + d_{75}$, hence the subpath 9,8,3,7,5 from B is copied over to A (see A') overwriting path 8,9,5,3,7. The path in parent B remains unchanged. Brady reports in [13] that for a 64-city problem it was best to search for subpaths of length between 8 and 32. A disadvantage of this approach is that it is quite expensive to search for possible crossover points.

With this scheme, only up to two foreign edges are copied to the parents. In the above example, the edges are (0,9) and (5,6).

Brady's algorithm can be regarded as the first MA proposed for the TSP.

The asynchronous parallel genetic optimization strategy (ASPARAGOS) [38, 39] has been the best evolutionary strategy for the TSP for years. In this approach, offspring are generated using maximum preservative crossover (MPX). Mutation is applied afterwards followed by a 2-repair, a variant of 2-opt local search focusing on newly introduced edges.

The MPX proposed in [39, 85] has similarities with the traditional two-point crossover. To construct an offspring tour, a subpath between two randomly chosen crossover points is copied from the first parent to the offspring. The partial tour is extended by copying edges from the second parent afterwards. If the subpath cannot be extended in this way to retain a feasible solution, the edges from the first parent are checked.

If there is no such edge from the first parent that can be used to extend the tour, a previously unvisited city is added from the second parent which comes next after the end point in the string. The table below shows an example.

MPX crossover										
Parent A	4	2	**0**	**8**	**9**	**5**	3	7	6	1
Parent B	2	9	8	3	7	5	0	1	6	4
Offspring C	**0**	**8**	**9**	**5**	7	3	1	6	4	2

In the example, the bold subpath from parent A is copied to the offspring. The offspring is extended by appending cities 7 and 3 so that the edges $(5, 7)$ and $(7, 3)$ contained in parent B are copied over. Edge $(3, 8)$ cannot be inserted since city 8 is already contained in offspring C. Looking at parent A, we see that both edges $(3, 5)$ and $(3, 7)$ cannot be used to extend the tour further. Hence, the city next to city 3 in parent B is identified: city 1. After adding city 1 to the partial offspring, the algorithm proceeds by inserting the remaining edges from parent B: edges $(1, 6)$, $(6, 4)$, and $(4, 2)$. The edge from the last to the first node is also contained in tour A, so we got only one foreign edge in offspring C.

Initially, a slightly different crossover had been used in ASPARAGOS [38] that is identical to the order crossover operator [20] except that a subpath of the second parent is inverted before crossing over. In the literature, this operator has been called the Schleuter crossover [24, 69] to avoid confusion with the MPX described above.

As shown in [24, 69], the edge recombination operators are superior to MPX in a GA without local search, and have a smaller failure rate (number of introduced foreign edges) than MPX. But when local search is added to the algorithm, the picture changes and MPX becomes superior to the edge recombination operators. As with tour construction heuristics in combination with local search, in the case of evolutionary variation operators the best stand alone operator does not necessarily perform the best in combination with local improvement [24, 69]. A major difference between ASPARAGOS and other EAs is that the algorithm is asynchronous and parallel. In contrast to traditional GAs, there is no discrete generation model, that is, there are no well distinguished (time-stepped) generations. Instead, selection for variation (mating selection) and selection for survival (replacement) are performed asynchronously. Furthermore, the population is spatially structured and consists of overlapping demes (local subpopulations). Mating (recombination) happens only between individuals within a deme (neighborhood). Therefore, no central control is needed and only local interactions occur. Thus, the algorithm is robust, is well suited for small populations, and can be executed on parallel hardware. The term PGA [86] is often used for such a model with population structures.

Fine-grained PGAs for the TSP have also been studied in [52], and a variant of ASPARAGOS has been proposed in [14] called the insular genetic algorithm. A modified version of ASPARAGOS has been proposed in [40] called ASPARAGOS96 with a hierarchical population structure and slightly modified MPX and mutation.

The strategic edge crossover (SEX) introduced in [80] by Moscato and Norman is similar to the edge recombination operator [114] in that an edge list is utilized to find tour segments consisting only of parent edges. These tour segments are closed up to a subtour eventually introducing foreign edges and finally, the subtours are combined into a single tour by Karp's patching heuristic [55]. Later, Holstein and Moscato developed in [48] another recombination operator, which first copies all common edges to the offspring and is therefore respectful. Secondly, from the parent tours, edges are chosen in order of increasing length ensuring that the TSP constraints are obeyed. Finally, the resulting tour segments are connected with a nearest neighbor heuristic.

The name genetic local search (GLS) was first used by Ulder *et al.* in [109] to describe an EA with recombination and consequently applied local search. Within this scheme, all individuals of the population represent local optima with respect to the chosen local search. In [109], the population model of a GA has been used instead of a model with a structured population and asynchronous application of the variation operators. The recombination operator used was MPX, and opposed to 2-repair, 2-opt and the LK local search were incorporated.

In [15], Bui and Moon also propose a GLS algorithm with LK as the local search procedure. They developed a k-point crossover operator with an additional repair mechanism to produce feasible offspring.

The approach described in the present paper as published in [34, 35, 71] is also a GLS and uses LK local search and a new recombination operator called the distance preserving crossover (DPX). This algorithm has won the first international contest on evolutionary optimization (ICEO) at the IEEE International Conference on Evolutionary Optimization [8, 34].

In [112] Walters developed a two-point crossover for a nearest neighbor representation and a repair mechanism called directed edge repair (DER) to achieve feasibility of the solutions. He uses 3-opt local search to improve the solutions further. Brood selection is incorporated to select the best of the children produced by crossover.

In [56] Katayama and Narihisa proposed an EA with LK and small populations (just two individuals) and a heuristic recombination scheme. Their approach is similar to the iterated LK heuristic but additional diversification is achieved by the recombination of the current solution and the best solution found. The results presented for large TSP instances are quite impressive.

2.3.4 Other highly effective evolutionary algorithms for the traveling salesman problem

There are some other highly effective EAs for the TSP which do not belong to the class of MAs but are worth mentioning.

In [87] Nagata and Kobayashi devised an EA that uses the edge assembly crossover to produce offspring. In this recombination operator, children are constructed by first creating an edge set from the edges contained in the parents (E-set) and then producing intermediate children for which the subtour constraint is generally not fulfilled. In order to obtain feasible offspring, subtours are merged in a greedy fashion based on the minimum spanning tree defined by the disjoint subtours.

In [108] Tao and Michalewicz proposed an EA which is very easy to implement. The operator used in the algorithm is called inver-over since it can be regarded as a mixture of inversion and crossover. The operator is similar to the LK heuristic since a variable number of edges are exchanged. Thus, it is more a local search utilizing a population of solutions than an EA utilizing local search.

In [77, 78] Möbius *et al.* proposed a physically inspired method for the TSP called thermal cycling with iterative partial transcription (IPT). To a population of solutions called "archive," a heating phase (similar to simulated annealing with nonzero temperature) and a quenching phase (local search) is repeatedly applied. After quenching, IPT is used to further improve the solutions in the archive. IPT can be regarded as a form of recombination in which some of the alleles of one parent are copied to the other, explicitly maximizing the fitness of the resulting individual.

Several other approaches have been published for solving the TSP. However, only a few of them are suited for solving large TSP instances ($\gg$1000 cities) like the ones discussed here. It is meaningless to test an approach on just small TSP instances, since (*a*) there are exact methods for solving small instances to optimality in a few seconds, (*b*) simple local search algorithms are much faster than most EAs and produce comparable or better results, and (*c*) the behavior of an algorithm on small instances cannot be used to predict its behavior on large instances.

3. Fitness landscape analysis

The concept of a "fitness landscape" [115], introduced to illustrate the dynamics of biological evolutionary optimization, has been proven to be very powerful in evolutionary theory. The concept has furthermore been shown to be useful for understanding the behavior of combinatorial optimization heuristics and can help in predicting their performance. For example, in [60] Kirkpatrick and Toulouse analyzed the search space of the TSP to explain the performance of simulated annealing. In their work on *NK*-landscapes, Kauffman and Levin recognized the impor-

tance of correlated landscapes for population based approaches [58]. Taking these studies into account, Moscato stressed the importance of the correlation of local optima for MAs [79]. Also based on a fitness lanscape analysis, Boese developed a heuristic for the TSP aimed at exploiting the properties of local optima [10, 11].

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 $f : S \to \mathbb{R}$ 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 $E = \{(s, s') \in S \times S \mid d(s, s') = d_{\min}\}$, 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 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 n = diam G_L.

3.1 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, that is, 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 in [113] measure the ruggedness of a fitness landscape.

Weinberger suggested performing random walks to investigate the correlation structure of a landscape. The random walk correlation

function [102, 103, 113]

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

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).

Based on this correlation function, the correlation length ℓ [103] of the landscape is defined as

$$\ell = -\frac{1}{\ln(|r(1)|)} \tag{3}$$

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, that is, 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 [113].

A ruggedness measure similar to the correlation length ℓ has been proposed in [1] 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 have shown in [63] 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 [54] as a measure for problem difficulty of GAs. The FDC coefficient ϱ is defined as

$$\varrho(f, d_{\text{opt}}) = \frac{\text{Cov}(f, d_{\text{opt}})}{\sigma(f)\,\sigma(d_{\text{opt}})}, \tag{4}$$

and determines how closely fitness and distance to the nearest optimum in the search space denoted by d_{opt} are related. ($\text{Cov}(x, y)$ denotes the covariance of x and y and $\sigma(x)$ denotes the standard deviation of x.) 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 through solutions with increasing fitness. A value of $\varrho = -1.0$ ($\varrho = 1.0$) for a maximization (minimization) problem

indicates that fitness and distance to the optimum are perfectly related and that search promises to be easy. A value of $\varrho = 1.0$ ($\varrho = -1.0$) means that with increasing fitness the distance to the optimum increases, too. To gain insight in the global structure of the landscape, a fitness distance analysis (FDA) can be performed for locally optimum solutions of a given problem instance, since the correlation of local optima has a large influence on population-based search, as is is the case for MAs [79].

Thus, it can be determined whether there is a structure in the distribution of locally optimum solutions which can be exploited by a metaheuristic 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 [10] for the TSP, Reeves for a flow-shop scheduling problem [95], and Merz and Freisleben for the graph bipartitioning problem [74].

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.

3.2 The fitness landscape of the traveling salesman problem

Several researchers have studied the fitness landscape of the TSP to find more effective search techniques. Even a theoretical analysis exists that coincides with conclusions drawn from experiments.

3.2.1 Distances between traveling salesman tours

Besides landscape analysis, distance functions for solution vectors of optimization problems are important in a number of EA techniques, such as mechanisms for preventing premature convergence [25] or the identification of multiple solutions to a given problem [98]. Furthermore, they can be used to observe the dynamic behavior of the EA (or CCVA [26]) and to guide the search of the EA [34].

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

$$d(t_1, t_2) = |\{e \in E \mid e \in t_1 \wedge e \ni t_2\}|. \tag{5}$$

This distance measure has been used by several researchers, including [11, 35, 67, 86]. Recently, it has been shown that this distance function satisfies all four metric axioms [99].

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 $d_{2\text{-opt}}$ is bound by $d \le d_{2\text{-opt}} \le 2d$ [67].

With this distance measure, the neighborhoods based on edge exchange can be defined as

$$\mathcal{N}_{k\text{-opt}}(t) = \{t' \in T : d(t, t') \le k\}, \tag{6}$$

with T denoting the set of all tours of a given TSP instance. Note that the node exchange neighborhood is a small subset of the 4-opt neighborhood, and the node (re)insertion neighborhood is a subset of the 3-opt neighborhood since 4 edges and 3 edges are exchanged, respectively.

3.2.2 Autocorrelation analysis for the traveling salesman problem

In [104] Stadler and Schnabl 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

$$r(s) \approx \exp(-s/\ell) = \exp(-b/n \cdot s), \tag{7}$$

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 landscape, the normalized correlation length $\xi = \ell/n$ is 1/2, for the node re-insertion landscape ξ is 1/3, and for the node exchange landscape ξ is 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 [97]. Equation (7) 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. Obviously, only neighborhoods with the same size should be compared in terms of the correlation length.

In the case of an asymmetric TSP, the above equation holds, too, with the exception that there is no 2-opt move if the distance matrix is asymmetric. Reversing a subpath in an asymmetric TSP tour leads generally to a k-change depending on the length of the subpath. Stadler and Schnabl have shown in [104] that such reversals yield a random

walk correlation function of the form

$$r(s) \approx \frac{1}{2}\delta_{0,s} + \frac{1}{2}\exp(-4s/n), \tag{8}$$

where δ denotes the Dirac function. $\delta_{0,s}$ is defined as

$$\delta_{0,s} = \begin{cases} 1 & \text{if} \quad s = 0, \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

in the discrete case.

As the results for the symmetric and the asymmetric TSP show, landscape ruggedness in terms of the random walk correlation function does not depend on the TSP instance itself and can therefore not be used to compare the "hardness" of TSP instances.

3.2.3 Fitness distance correlation analysis for the traveling salesman problem

The correlation of fitness of local optima and distance to the optimum solution has already been studied by Boese in [9, 10] in order to derive a suitable search strategy for the TSP. However, he concentrated in his studies [10] on a single TSP instance contained in TSPLIB [96], a publicly accessible library of TSP instances.

To obtain more general information, additional instances have been analyzed for which the results are presented in the following. The instances are selected to cover different problem sizes as well as problem characteristics. The first three instances denoted mpeano7, mnpeano7, and David5 are fractal instances based on L-systems (such as the Koch curve) with known optimum tours described in [81, 82, 88]. The number in the name denotes the order of the fractal.

The other nine instances are chosen from TSPLIB. The first instance denoted ts225 is known to be hard to solve exactly by branch and cut algorithms [2] although it has a small number of cities. Instance pcb442 is a printed circuit board production instance with a regular location of the nodes. The instances att532, pr1002, and pr2392 are instances derived from real city locations. rat783 is an instance with a random distribution of the cities in a rectangular area. dsj1000 denotes an instance with clustered cities. And finally, the instances fl1400 and fl1577 are printed circuit board drilling problems. The latter of the two has been the smallest unsolved problem in TSPLIB for a long time. Recently, it could be solved to optimality, however. In Figure 5, some characteristic instances are displayed.

To obtain insight into the structure of the fitness landscapes of these instances, experiments have been conducted in which the (cor-)relation of fitness and distance to the optimum of locally optimum solutions has been investigated. For instances with more than one known optimum solution, the distances to the nearest optimum was considered. For example, the number of optima found in experiments for the instances

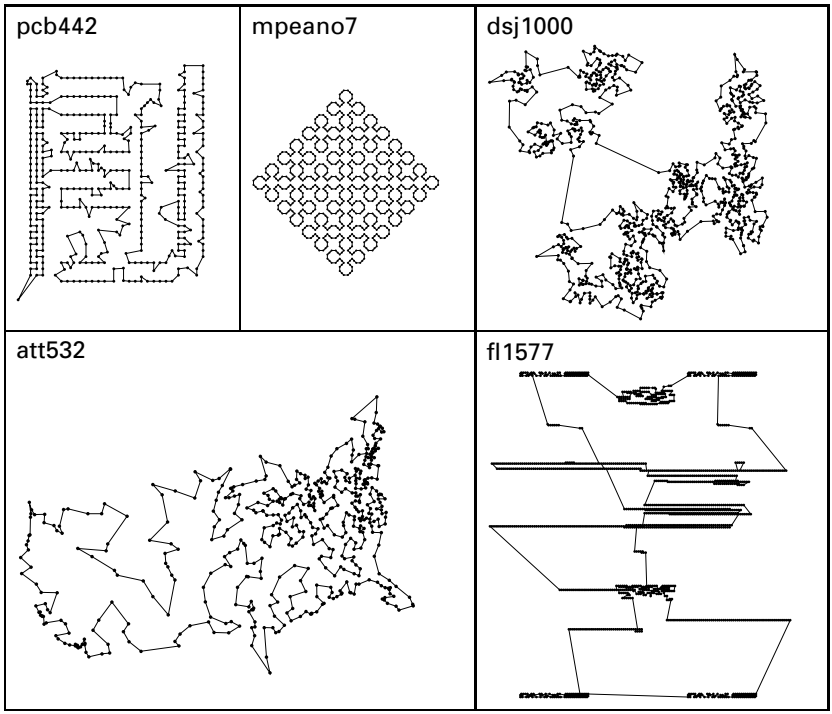

Figure 5. Optimum tours of five TSP instances.

ts225, rat783, and fl1400, is 147, 17, and 7, respectively. For the first two instances, the average distance between the optima is 25.8 and 9.5, respectively. The optima found for instance fl1400 have an average distance of 336.6. It is assumed that all fl instances have a high number of global optima. Since just one global optimum was known to the authors at the beginning of the experiments, no other global optima have been considered in the analysis.

In a first series of experiments, the local optima were produced by a fast 3-opt local search applied to randomly generated solutions. The results are presented in Table 1. In the first column, the name of the instance is displayed, and in the second column the problem size n is given. In columns three through seven, the minimum distance of the local optima to a global optimum ($\min d_{\text{opt}}$), the average distance of the local optima to the global optimum ($\overline{d}_{\text{opt}}$), the average distance between the local optima ($\overline{d}_{\text{loc}}$), the number of distinct local optima ($N_{3-\text{opt}}$) out of 2500, and the fitness distance correlation coefficient (ϱ) are provided, respectively. Additionally, the normalized average distance, that is, the average distance of the local optima to the global optimum divided by

Instance	n	$\min d_{\text{opt}}$	$\overline{d}_{\text{opt}}$	$\overline{d}_{\text{loc}}$	$N_{\text{3-opt}}$	ϱ
mnpeano7	724	20	85.32 (0.12)	138.53	2500	0.50
mpeano7	852	0	1.93 (0.01)	3.83	840	0.40
David5	1458	0	29.98 (0.02)	57.55	2498	0.56
ts225	225	19	33.90 (0.15)	35.07	2496	0.18
pcb442	442	63	105.95 (0.24)	109.74	2500	0.48
att532	532	36	106.48 (0.20)	123.17	2500	0.57
rat783	783	83	151.82 (0.19)	184.77	2500	0.68
dsj1000	1000	122	207.93 (0.21)	239.87	2500	0.36
pr1002	1002	123	203.00 (0.20)	242.16	2500	0.57
fl1400	1400	504	574.85 (0.41)	561.26	2500	0.07
fl1577	1577	152	239.90 (0.15)	260.10	2500	0.27
pr2392	2392	283	430.04 (0.18)	496.62	2500	0.63

Table 1. Results of the fitness distance analysis for 3-opt solutions of the TSP.

the maximum distance in the search space n is shown in column four in parentheses. In cases of more than one known global optimum, the distance to optimum means the distance to the nearest optimum.

In cases of the fractal instances mpeano7 and David5, the optimum could be found with fast 3-opt. The average distance to the optimum is very small compared to the maximum distance in the search space, and the locally optimum solutions are close together. mpeano7 appears to have a small number of local optima since in the analysis only 840 distinct local optima could be found. For the problems contained in TSPLIB, the normalized average distance to the optimum is about 0.2, with one exception: for fl1400 the value is about 0.4. Thus, all TSPLIB instances have local optima with a significantly higher distance to the optimum than the fractal instances. For all instances, the average distances between the local optima are similar to the average distance to the optimum. The correlation coefficient is high for the instances based on real city instances, and clusters seem to affect correlation negatively. For the random instance rat783, the correlation coefficient is highest, and it is lowest for the drilling problems and ts225.

For the same set of instances, a second series of experiments has been conducted. In these experiments, the local optima were generated with the LK heuristic rather than with 3-opt. The results are displayed in Table 2. The local optima generated by the LK heuristic show the same properties as those obtained by 3-opt. The correlation coefficients are slightly higher for almost all TSPLIB instances, and in cases of the fractal instances they are close to 1. Fitness distance plots for some of the instances are provided in Figure 6. The distance to the optimum is plotted against the fitness (cost) difference between the locally optimum

Instance	n	$\min d_{\text{opt}}$	$\overline{d}_{\text{opt}}$	$\overline{d}_{\text{loc}}$	N_{LK}	ϱ
mnpeano7	724	0	20.94 (0.03)	39.09	118	0.99
mpeano7	852	0	13.56 (0.02)	25.99	87	0.99
David5	1458	0	3.82 (0.01)	7.55	137	0.94
ts225	225	20	33.60 (0.15)	34.98	2497	0.21
pcb442	442	61	105.92 (0.24)	109.82	2500	0.50
att532	532	47	106.29 (0.20)	122.71	2500	0.54
rat783	783	75	151.38 (0.19)	184.51	2500	0.70
dsj1000	1000	105	208.19 (0.21)	240.01	2500	0.36
pr1002	1002	108	202.15 (0.20)	241.77	2500	0.60
fl1400	1400	511	575.23 (0.41)	560.71	2500	0.06
fl1577	1577	151	238.95 (0.15)	259.55	2500	0.34
pr2392	2392	310	429.35 (0.18)	496.47	2500	0.64

Table 2. Results of the fitness distance analysis for LK solutions of the TSP.

solutions and the fitness of the global optimum ($\Delta f = c(\pi_{\text{loc}}) - c(\pi_{\text{opt}})$). The instance mpeano7 shows perfect correlation between the fitness difference and the distance to the optimum. The local optima form a straight line originating from the optimum. The plot for ts225 looks quite different: for some fitness values, there are several local optima while for most fitness values there is not even a single one, leading to large gaps in fitness of the local optima. Problems att532, rat783, and pr2392 exhibit a "cloud" of local optima in their scatter plots. The means of the points are oriented along a straight line. The clustered instance dsj1000 is similar but there is no orientation towards the optimum. This phenomenon becomes more apparent in the problems fl1400 and fl1577. The means of the points are distributed parallel to the Δf-axis.

The analysis has shown that local optima in the TSP are found in a small region of the search space: on the average, more than 3/4 of the edges are common to all local optima with one exception, fl1400. Furthermore, fitness and distance to the optimum are correlated for most instances, and the average distance between the local optima is similar to the distance to the optimum. Thus, the global optimum appears to be more or less central among the local optima. Boese calls the structure of the TSP landscape the big valley structure, since local optima are closer together if they are closer to the optimum, and the smaller the tour length (cost), the closer they are to the optimum. However, the analysis has also shown that not all instances exhibit this structure as, for example, ts225. Furthermore, the analysis indicates that problems from application domains such as the drilling problems are harder to solve than randomly generated instances with uniform distribution. The

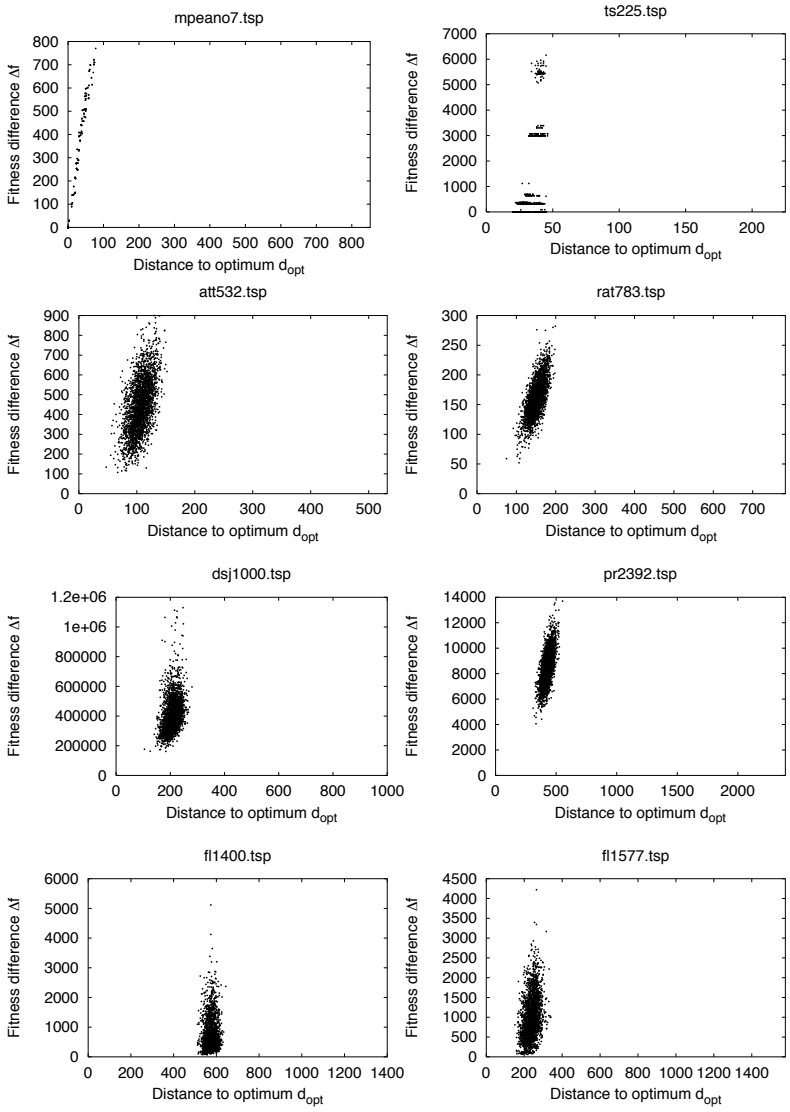

Figure 6. Fitness distance scatter plots produced with LK.

fractal instances on the other hand are very easy to solve. They are not well suited as benchmark problems for highly effective heuristics since they do not have the same characteristics as the instances arising in TSP applications. The big valley structure can be well exploited by an MA with recombination since good solutions are more likely to be found near other local optima, and most recombination operators produce solutions that lie "between" other solutions (respectful recombination).

Furthermore, an EA usually increases fitness of the solutions contained in the population while simultaneously decreasing the distance between the solutions.

The TSP instances considered in the analysis can be regarded as very well suited for MAs with respectful recombination. Many other combinatorial optimization problems do not share the properties of TSP landscapes. For example, the local optima of graph bipartitioning instances have an average distance to the optimum or best-known solution that lies slighty below the maximum distance (diameter) of solutions in the search space [73]. In some landscapes of the quadratic assignment problem, the local optima are randomly distributed over the search space (no correlation of fitness and distance to the optimum), again with an average distance near the diameter of the landscape [74]. The correlation length, on the other hand, depends in some problems on the problem instance itself, as can be observed for the quadratic assignment problem [73], for *NK*-landscapes, and for the binary quadratic programming problem [70].

4. Effective memetic algorithms for the traveling salesman problem

The proposed MAs for the TSP are similar to the EA outlined above: A population of locally optimum solutions is evolved over time by applying evolutionary variation operators (mutation and recombination operators). To ensure that the individuals in the population are local optima, after each application an evolutionary variation operator, local search is applied. This includes the initialization phase of the population in which solutions are constructed from scratch: A local search procedure is applied to these solutions so that even the first generation consists exclusively of local optima.

The problem-specific parts of the algorithm comprise initialization, local search, and the evolutionary variation operators.

4.1 Initialization and local search

To initialize the population of the MA, a local search procedure is applied to solutions constructed by the randomized greedy heuristic described above. However, the randomization technique proposed by Johnson *et al.* in [53], is not well suited for initialization of MA since the resulting solutions are very similar. Therefore, a variant is used: Before the greedy construction scheme is applied, $n/4$ edges are inserted in the tour solution randomly by selecting the edge to the nearest or second nearest unvisited neighbor of a randomly chosen unvisited city. The edge to the nearest city is selected with a probability of 0.66 and the edge to the second nearest city is selected with probability 0.33. After an edge has been inserted, the endpoints of the edge are marked

as visited to guarantee that the partial solution will not become an infeasible solution.

Since the LK heuristic is the best local search heuristic proposed for the TSP, it is used in our algorithm. In some cases, the simpler fast 3-opt heuristic is used when it is more efficient to use a fast but less elaborate local search.

4.2 Variation operators

Mutation operators used in simple EAs are not suited for use in MAs, since subsequently applied local search procedures will usually revert the changes made. For example, the inversion operator randomly exchanging two edges is ineffective when 2-opt, 3-opt, or LK local search is used. Therefore, in MAs alternative mutation operators are required.

4.2.1 The mutation operator

The mutation operators of our algorithms are based on edge exchange. There are two variants, one of which produces arbitrary exchanges of a predefined number of k edges, and the other one which produces nonsequential edge exchanges. The smallest of such an exchange is displayed in Figure 7 and involves four edges [66]. It stands in contrast to the sequential edge exchanges performed by the LK heuristic as described above. Since the LK heuristic performs sequential changes, the probability is minimized that LK reverses mutation if nonsequential edge exchanges are utilized. In the effective iterated LK heuristic, the nonsequential four-change is used as a mutation operator to escape from the basins of attraction of local optima.

4.2.2 The distance preserving crossover operator

In the case of recombination, previously published operators for EAs without local search can be used in MAs, but as shown in [24, 69], there may be others that are better suited for use with MAs. These operators may be ineffective when used without local search.

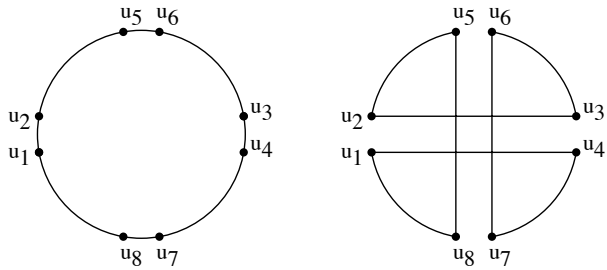

Figure 7. The nonsequential four-change.

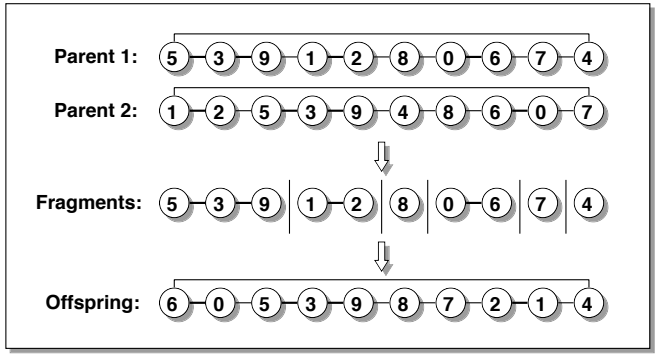

Figure 8. The DPX recombination operator for the TSP.

The distance preserving crossover (DPX) proposed in [34, 35] is such an operator that is only useful in combination with local search. In contrast to other recombination operators such as the edge recombination operators [105, 114], it forces the inclusion of foreign edges in the offspring instead of preventing them.

DPX tries to generate an offspring that has equal distance to both of its parents, that is, its aim is to achieve that the three distances between offspring and parent 1, offspring and parent 2, and parent 1 and parent 2 are identical. It works as follows. The content of the first parent is copied to the offspring, and all edges that are not in common with the other parent are deleted. The resulting parts of the broken tour are reconnected without using the nonshared edges of the parents. A greedy reconnection procedure is employed to achieve this: if the edge (i, j) has been destroyed, the nearest available neighbor k of i among the remaining tour fragments is taken and the edge (i, k) is added to the tour, provided that (i, k) is not contained in the two parents. In order to illustrate the DPX operator, let us consider an example.

Suppose that the two parents shown in Figure 8 are given, then copying parent 1 to the offspring and deleting the edges not contained in both parents leads to the tour fragments 5 3 9 - 1 2 - 8 - 0 6 - 7 - 4. The greedy reconnection procedure fixes the broken connections by producing the offspring shown in Figure 8 as follows. First, a city is chosen randomly as the starting point for the reconnection. Let us assume that the city to begin with is city 6, then the other endpoint (city 0) of the fragment containing city 6 is considered and its nearest neighbor in the set of available cities {5,9,1,2,4} is determined. The set of available cities only contains the start and endpoints of unvisited tour fragments. City 8 and city 7 are not contained in this set, because it is not desirable to re-insert edge $(0, 8)$ or edge $(0, 7)$, since they are contained in parent 1 or parent 2, respectively. Let us assume that in the

example the nearest neighbor to city 0 is city 5, so city 0 is connected to city 5, and the end of the connected fragment (city 9) is considered. At this point, the set of available cities is {2,8,7}. The procedure is repeated until all fragments have been reconnected. Note that the distance d between the offspring and both parent 1 and parent 2 is identical to the distance between the two parents ($d = 6$), hence the name distance preserving crossover.

In some rare cases, it is necessary to introduce backtracking into the greedy reconnection procedure to fulfill the distance criterion. For example, if in the example above the edges $(2, 0)$, $(6, 4)$, $(7, 8)$, $(7, 9)$, and edge $(1, 8)$ are inserted due to the nearest neighbor relations, the remaining edge to close the tour is edge $(4, 5)$. Since this edge is contained in parent A, the resulting child will not fulfill the distance criterion: the distance to parent A becomes 5 and the distance to parent B becomes 6. In such a situation, a backtracking mechanism trying alternative edges in preceding steps has to be employed. However, in the MA used in the experiments, the DPX operator does not include backtracking since the extra computation time for backtracking is not worthwhile. The cases for which the distance criterion is not obeyed are extremely rare and experiments have shown that the use of backtracking in the DPX has no influence on the overall performance of the MA. Thus, the easier to implement "one-pass" DPX is used in the MA experiments.

4.2.3 The generic greedy recombination operator

To study the important characteristics of recombination operators for the TSP, a new recombination operator is proposed in the following that utilizes the greedy construction scheme of the greedy heuristic described above. The generic greedy recombination operator (GX) consists of four phases. In the first phase, some or all edges contained in both parents are copied to the offspring tour. In the second phase, new short edges are added to the offspring that are not contained in one of the parents. These edges are selected randomly among the shortest edges emanating from each node. These edges are with high probability contained in (near) optimum solutions and are thus good candidates for edges in improved tours. In a third phase, edges are copied from the parents by making greedy choices. Here, edges may be included that are not common to both of the parents. Edges are inserted in order of increasing length and only candidate edges are considered, that is, edges that do not violate the TSP constraints. In the fourth and last phase, further edges are included utilizing the greedy construction scheme of the greedy heuristic described above until the child consists of n edges and is thus a feasible TSP tour. The operator is motivated by the facts that (*a*) in the case of an MA framework, implicit mutations can have a positive effect on the subsequent local search, as shown in [24, 69], (*b*) respectfulness is considered an important property of recombination

```
function GX(a,b in X; cRate, nRate, iRate: Real): X;

  begin
    let x = {};
    let rem = n;
    /* Copy common edges */
    foreach edge e in a do
      if (e in b and cRate < random[0,1)) then
        add e to x;
        rem := rem – 1;
      end;
    end;
    /* Insert new edges */
    for k := 1 to (rem * nRate) do
      i := n * random[0,1);
      j := select from (the 5 nearest neighbors of i)
      with (i, j) feasible and (i, j) not in a or b;
      add edge (i, j) to x;
      rem := rem – 1;
    end;
    /* Inherit edges from parents */
    for k := 1 to (rem * iRate) do
      parent := select randomly from (a, b);
      if (parent has candidate edges) then
        e := select from (2 shortest candidates);
        add e to x;
        rem := rem – 1;
      end;
    end;
    /* greedy completion */
    while (rem > 0) do
      e := select from (2 shortest candidates);
      add e to x;
      rem := rem – 1;
    end;
    return x;
  end;
```

Figure 9. The GX recombination operator for the TSP.

operators, (*c*) innovation is an important aspect in MAs due to small population sizes, and (*d*) the greedy heuristic is more effective than the nearest neighbor heuristic in constructing a feasible TSP tour. The pseudo code of the recombination operator is provided in Figure 9.

The GX operator has three parameters. The common edges inheritance rate (*cRate*) that determines the probability that a common edge is added to the child and is thus a control parameter for the first phase.

With a rate of 1.0, respectful recombination is achieved, all other rates lead to unrespectful recombination. The second phase is controlled by the new edges introduction rate (*nRate*) that determines the number of new edges to insert. A rate of 0.5, for example, determines that half of the remaining edges to insert after phase one are new edges that are short but not contained in one of the parent solutions. The number of edges to inherit from the parents including edges not common to both parents is determined by the inheritance rate (*iRate*). In the last phase, edges in increasing length are chosen that may or may not be found in the parents. Note that the recombination operator proposed in [48] is very similar to GX with *cRate* = 1.0, *nRate* = 0.0, and *iRate* = 1.0. However, in the last phase of GX, the greedy heuristic is used instead of the nearest neighbor heuristic, which is more effective.

4.2.4 Local search and recombination

In MAs, recombination operators are desired that are efficient in combination with the local search that is applied after a child has been generated. Thus, it makes sense to tune the local search for its operation after recombination. The landscape analysis has shown that there is correlation between tour length and distance to the optimum of local minima and that a local optimum with high fitness (short tour length) is contained near other local optima with high fitness. Therefore, it makes sense to restrict a local search after recombination to search only the region around or between the two parents. This can be accomplished by fixing all common edges that have been included in the child in the first step of recombination. The edges that are common to both parents can be regarded as the "building blocks" of the evolutionary search and should be found in good offspring tours. Fixing these edges prevents the local search from replacing these edges by others and reduces the running time of the local search considerably. The fixing of edges reduces the problem size for the local search since fixed edges are not considered during the search for edge exchanges. After the local search has been terminated, all edges are marked as not fixed.

The landscape analysis has shown that less than one fourth of the edges in the local optima are different from the optimum tour. Thus, in the first generation of an MA, a local search operates on a problem with a dimensionality of one fourth of the original one if the fixing of edges is performed during recombination. Since with ongoing evolution, the distance between the members of the population diminishes, the size of the problem becomes smaller for the local search in each generation. This leads to a significantly reduced running time for the local search.

4.2.5 Selection and restarts

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 [101]. Additionally, duplicates will be replaced by other solutions, so that each phenotype exists only once in the new population.

The population size of an MA is typically small compared to GAs: a size of 10 up to 40 is common in an MA, since the computational complexity of the local search does not allow evolution of much larger populations within 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 in [27] 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 have 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.

4.3 Implementation details

In the implementation of the algorithms for the TSP described in this paper, a nearest neighbor list of size $m = 100$ for each node is maintained, which is initialized by nearest neighbor queries on a two-dimensional binary search tree [6]. In the local search procedures, a data structure for maintaining don't look bits is incorporated, with the local search for the initial population starting with all don't look bits set to zero. After recombination has been performed, only the don't look bits of the nodes that are incident to the edges not shared by both parents are cleared. Similarly, after mutation, only nodes incident to the edges newly included in the tour have their don't look flags set to zero. This focuses the search of the hill-climber to the promising regions of the search space

and also reduces the time for checking the interesting members of the neighborhood.

Additionally, in the algorithm for the TSP, data structures have been incorporated to deal with large instances of up to 100,000 cities. Since for large instances it is not possible to store the entire distance matrix in main memory the euclidean distances are computed online. This is a rather expensive operation, so a distance cache of size $3 \cdot n$ is maintained, where the first n entries are used to cache the distances of the edges in the current tour and the remaining $2 \cdot n$ entries are organized as described in [6]. The average hit rate of the cache varies between 80% and 95%.

Another target for optimizations is the LK heuristic itself. Most of the computation time is spent in submoves that will be reversed later in the algorithm. Hence, it is profitable to distinguish between tentative and permanent moves. Applegate and Cook have proposed a segment tree data structure for efficiently managing tentative moves, as described in [33]. Instead of using a segment tree, the algorithms described here operate on a segment list that represents a tentative tour. Operations performing a flip on this tentative tour are highly optimized, such that a high performance gain compared to the simple array representation can be achieved. The running times for all operations are in $O(1)$, since the data structure is limited to perform 20 flips only. In practice, this has proven to be sufficient.

4.4 Performance evaluation

Several experiments have been conducted to evaluate the performance of MAs for the TSP. All experiments described in the following were conducted on a PC with a Pentium III Processor (500 MHz) under Linux 2.2. All algorithms were implemented in C++.

4.4.1 Comparison of recombination operators

In a first set of experiments, several recombination operators for the TSP were tested under the same conditions on three selected TSP instances contained in TSPLIB: att532, pr1002, and fl1577. To get a clear picture of the operator effectiveness, no additional mutation was performed and the restart mechanism was disabled during the runs. Furthermore, a fast 2-opt local search was used in the MAs that is not as effective as 3-opt local search or the LK heuristic to reduce the strong influence of the (sophisticated) local search. The recombination operators MPX, DPX, and GX were studied with various parameter settings. The population was set to $P = 100$ in all runs, and the variation operator application rate was set to 0.5, that is, 50 offspring were produced per generation. The results of the experiments are summarized in Table 3. For each instance/operator, the average number of generations, the shortest tour length found, and the percentage access over the optimum solution value is provided. For the GX operator, the values for *cRate*, *nRate*, and *iRate*

Operator	att532		pr1002		fl1577	
DPX	1565	0.386%	664	2.778%	653	0.292%
MPX	2691	0.311%	3404	1.023%	1240	0.444%
GX-Params						
1/0.5/1	868	0.158%	759	0.719%	624	0.206%
1/0.5/0.75	929	0.148%	733	1.133%	713	0.205%
1/0.5/0.5	923	0.142%	808	0.801%	682	0.214%
1/0.5/0.25	892	0.137%	832	0.648%	641	0.245%
1/0.25/0	928	0.139%	1223	0.628%	690	0.250%
1/0.25/0.75	1091	• 0.120%	1430	0.633%	769	0.206%
1/0.25/0.5	1065	0.131%	1422	0.595%	684	0.282%
1/0.25/0.25	998	0.135%	1334	• 0.565%	696	0.261%
1/0/1	956	0.280%	1321	0.901%	736	0.335%
1/0/0.75	1071	0.152%	1481	0.714%	735	• 0.174%
1/0/0.5	1035	0.142%	1434	0.735%	744	0.283%
1/0/0.25	1006	0.186%	1412	0.749%	719	0.347%
0.75/0.25/0	233	1.084%	229	3.948%	227	1.932%
0.75/0.25/0.75	269	1.968%	288	4.007%	269	1.897%
0.75/0.25/0.5	254	1.363%	258	3.972%	254	1.838%
0.75/0.25/0.25	243	1.049%	240	3.991%	239	1.800%
0.75/0/1	407	0.661%	422	1.734%	270	1.503%
0.75/0/0.75	517	0.309%	705	• 1.024%	620	• 0.316%
0.75/0/0.5	457	• 0.221%	558	1.232%	398	0.747%
0.75/0/0.25	415	0.233%	435	1.386%	298	1.093%
0.5/0.25/0	156	2.558%	179	3.998%	161	2.143%
0.5/0.25/0.75	191	2.699%	224	4.007%	187	2.143%
0.5/0.25/0.5	172	2.630%	201	4.007%	178	2.139%
0.5/0.25/0.25	162	2.483%	187	4.007%	170	2.143%
0.5/0/1	195	1.285%	216	2.954%	174	1.999%
0.5/0/0.75	403	0.667%	455	• 1.535%	363	• 0.751%
0.5/0/0.5	293	• 0.551%	316	1.627%	242	1.263%
0.5/0/0.25	220	0.754%	227	2.559%	192	1.706%
ILS	61365	0.331%	126457	0.633%	150797	0.540%
NS4	744	0.629%	1438	1.111%	1633	0.247%
Time:	60 sec.		120 sec.		200 sec.	

Table 3. Comparison of MA recombination strategies for the TSP using 2-opt.

are provided in the form *cRate/nRate/iRate*. For example, a parameter setting of 1/0.25/0.75 means that the common inheritance rate *cRate* was set to 1.0, the new edges introduction rate *nRate* was set to 0.25, and the inheritance rate *iRate* was set to 0.75. The dot in each column block indicates the best result within this block.

For all three instances, MPX and DPX are outperformed by GX for some of the parameter settings: all GX variants with a common inheritance rate of 1.0 and a new edge introduction rate of 0.25 perform better than MPX and DPX. However, the best parameter setting for GX is a different one for each of the instances implying that there is no "golden rule" leading to the best recombination strategy for all TSP instances! For example, the best setting for fl1577 is 1/0/0.75 but all other combinations with *nRate* set to 0.0 do not perform as good as the GX variants with *nRate* set to 0.25. Furthermore, it becomes apparent that respectfulness is a very important property of recombination operators since all GX versions with a common inheritance rate less than 1 perform significantly worse than the respectful greedy recombination operators. However, choosing a high inheritance rate can compensate the phenomenon to an extent since the common edges of the parents have a chance to be included in the offspring in the third phase of the generic recombination. Additionally, iterated 2-opt local search (ILS) and an MA with the nonsequential four-change mutation and no recombination has been applied to the three instances. The mutation-based algorithms perform relatively good but cannot compete with the greedy recombination MAs. The correlation structure of the landscape can be exploited by a recombination-based MA. For the instance fl1577, the MA with NS4 performs much better than ILS indicating that for this type of landscape search from multiple points (population-based search) is more promising.

In the second experiment, the fast 2-opt local search has been replaced by the LK heuristic. The population size was set to 40, the variation operator application rate was set to 0.5; that is, 20 offspring were produced per generation, and restarts were enabled with a diversification rate of 0.3. The results obtained from experiments with MAs using DPX, MPX, respectful GX, and nonsequential four-change mutation (NS4) in comparison to the iterated LK heuristic (ILK) are displayed in Table 4. For each instance/operator pair, the average number of generations and the percentage access over the optimum solution value is provided. For the GX operator, the values for *nRate* and *iRate* are provided in the form *nRate/iRate*. *cRate* was set to 1.0 in all experiments. The dot in each row indicates the best result for an instance.

Here, the performance differences of the MAs are in most cases not significant. For the problems rat783 and pr1002 all algorithms perform well with only small differences, except for the MA with MPX recombination in the case of pr1002. Surprisingly, this MA performs

	rat783	pr1002	fl1577	pr2392	pcb3038
ILK	0.018 %	0.065 %	0.158 %	0.215 %	0.135 %
DPX	0.004 %	0.023 %	• 0.028 %	0.068 %	0.113 %
MPX	• 0.001 %	0.169 %	0.142 %	0.054 %	0.128 %
NS4	0.010 %	0.020 %	0.181 %	0.119 %	0.171 %
GX 1.0/1.0	0.007 %	0.036 %	0.055 %	0.042 %	0.132 %
GX 1.0/0.75	0.026 %	0.022 %	0.058 %	0.053 %	0.211 %
GX 1.0/0.5	0.008 %	0.011 %	0.045 %	0.050 %	0.171 %
GX 1.0/0.25	0.006 %	0.013 %	0.051 %	0.047 %	0.146 %
GX 0.5/0.5	0.006 %	0.009 %	0.042 %	0.037 %	0.112 %
GX 0.5/0.75	0.007 %	0.031 %	0.048 %	0.055 %	0.175 %
GX 0.5/0.5	0.008 %	0.005 %	0.046 %	0.051 %	0.143 %
GX 0.5/0.25	0.009 %	0.011 %	0.037 %	0.044 %	0.136 %
GX 0.25/0	0.002 %	0.017 %	0.044 %	0.022 %	0.125 %
GX 0.25/0.75	0.012 %	0.003 %	0.041 %	0.031 %	0.151 %
GX 0.25/0.5	0.006 %	0.002 %	0.036 %	0.025 %	0.111 %
GX 0.25/0.25	0.005 %	0.002 %	0.040 %	0.023 %	• 0.111 %
GX 0.0/1.0	0.008 %	0.006 %	0.052 %	• 0.020 %	0.123 %
GX 0.0/0.75	0.003 %	• 0.000 %	0.043 %	0.027 %	0.115 %
GX 0.0/0.5	0.011 %	0.008 %	0.052 %	0.029 %	0.122 %
GX 0.0/0.25	0.004 %	0.002 %	0.050 %	0.035 %	0.123 %
Time:	80 sec.	200 sec.	300 sec.	400 sec.	800 sec.

Table 4. Comparison of MA recombination strategies for the TSP using LK.

significantly worse than the other algorithms. For fl1577, the MAs with DPX and GX outperform all other competitors, with the MA using DPX being the best. For pr2392, all recombination-based algorithms perform similarly, but the MAs with mutation and ILK perform significantly worse. In the case of pcb3038, the largest instance considered, all results lie close together. The MAs with DPX and MPX outperform ILK and the MA with NS4. In the greedy recombination MAs, high differences can be observed. The best results are obtained with a new edge introduction rate of 0.25. The results show no clear tendency and often the values lie too close together to be significantly different. However, in none of the cases did ILK or the MA with mutation outperform the MA using DPX or the best greedy recombination. The performance differences between mutation and recombination operators have become more apparent using 2-opt local search. For larger instances, this may also be observed for MAs with the LK heuristic.

In an additional experiment, the combination of recombination and mutation operators in MAs has been investigated. In the same experimental setup as before, the MAs with DPX and MPX recombination

	att532	rat783	pr1002	fl1577	pr2392	pcb3038
DPX	0.030 %	0.004 %	0.023 %	0.028 %	0.068 %	0.113 %
DPX, $m = 0.1$	0.017 %	0.001 %	0.012 %	0.027 %	0.021 %	0.099 %
DPX, $m = 0.5$	0.017 %	0.007 %	0.000 %	0.041 %	0.043 %	0.106 %
MPX	0.021 %	0.001 %	0.169 %	0.142 %	0.054 %	0.128 %
MPX, $m = 0.1$	0.013 %	0.000 %	0.041 %	0.146 %	0.053 %	0.094 %
MPX, $m = 0.5$	0.025 %	0.005 %	0.054 %	0.138 %	0.047 %	0.103 %
Time:	60 sec.	80 sec.	200 sec.	300 sec.	400 sec.	800 sec.

Table 5. Comparison of MAs with recombination and mutation (NS4).

have been run with the nonsequential four change mutation operator. The results are provided in Table 5. The table contains the results achieved with DPX and MPX without mutation as well as the results for a mutation operator application rate of $m = 0.1$ and $m = 0.5$. The number of offspring per generation produced by mutation is $m \cdot P$. The results show a clear tendency: in the majority of runs, additional mutation improves the results. Furthermore, it is shown that the mutation application rate of $m = 0.1$ is preferable.

4.4.2 Results on small and medium traveling salesman problem instances

Using a mutation application rate of $m = 0.1$, the MAs have been run on a variety of problem instances contained in TSPLIB to show the robustness and scalability of the memetic approach. Table 6 shows the results for five instances up to a problem size of 1002. The population size was set to $P = 40$ in all runs, the recombination application rate was set to 0.5, and the diversification rate to 0.1. Two MAs were run on each instance, the first one with DPX recombination and the second one with GX recombination. In the latter, *cRate* was set to 1.0, *nRate* was set to 0.1 (which appears to be a good compromise between 0.25 and 0.0), and *iRate* was set to 0.5. The programs were terminated as soon as they reached an optimum solution. In the table, the average number of generations (gen) and the average running time of the algorithms (t in s) in seconds is provided. In 30 out of 30 runs, the optimum could be found for all instances in less than two minutes. The average running time for rat783 is much lower than for att532 which is not surprising since the landscape of the random instance rat783 has a higher fitness distance correlation coefficient. In most cases, the MA with greedy recombination appears to be slightly superior to the MA with DPX. For larger instances, the average time to reach the optimum as well as the deviation of the running time increases dramatically. Thus, the MAs were run on the larger instances with a predefined time limit. Table 7 summarizes the results for the MA with GX recombination. The population size was set to $P = 100$ for pr2392 and pcb3038, since

Instance	Op	gen	quality	N_{opt}	t in s
lin318	DPX	19	42029.0 (0.000%)	30/30	8
	GX	13	42029.0 (0.000%)	30/30	8
pcb442	DPX	824	50778.0 (0.000%)	30/30	147
	GX	286	50778.0 (0.000%)	30/30	68
att532	DPX	560	27686.0 (0.000%)	30/30	127
	GX	289	27686.0 (0.000%)	30/30	106
rat783	DPX	122	8806.0 (0.000%)	30/30	26
	GX	136	8806.0 (0.000%)	30/30	35
pr1002	DPX	333	259045.0 (0.000%)	30/30	112
	GX	182	259045.0 (0.000%)	30/30	98

Table 6. Average running times of two MAs to find the optimum.

Instance	gen	quality	sdev.	N_{opt}	t in s
pr2392	2407	378032.6 (0.000%)	0.8	27/30	2588
pcb3038	5248	137702.6 (0.006%)	6.4	3/30	6955
fl3795	341	28794.7 (0.079%)	21.3	1/30	7212

Table 7. Performance of MA using GX on large TSP instances.

smaller population sizes led to poorer performance. Due to the long running time of the LK heuristic, the population size for fl3795 was set to $P = 40$. In the table, the average number of generations evolved by the MA (gen), the average final tour length, the percentage access over the optimum solution value (in parentheses), the standard deviation of the final tour length (sdev.), the number of times the optimum was found (N_{opt}), and the running time in seconds (t in s) is provided.

The running times presented here can only be indirectly compared with results of alternative approaches found in the literature, since different hardware/software platforms have been used. However, it appears that the MA presented here outperforms other approaches. With ASPARAGOS96 [40], an average tour length of 8809 (0.03%) could be found in approximately three hours on a 170 MHz SUN UltraSparc for rat783, and an average final tour length of 28,820 (0.34%) for fl3795 in approximately 17 hours. These results are significantly worse in both running times and solution quality. With the edge assembly crossover [87], the running time for finding the optimum for rat783 was 3013 seconds on a PC with a 200 MHz Intel Pentium processor, which is much slower even taking the performance differences of the processors into account. The running time to reach a solution quality of 0.006% for pr2392 was 33,285 seconds which is worse than the MA presented here in both quality and time.

The physically inspired IPT approach [78] outperforms the MA on problem fl3795, for which it requires 6050 seconds on a HP K460 Server with 180 MHz PA8000 processors to find the optimum solution. However, the MA is superior on the instances att532, rat783, and pr2392 in terms of average solution quality. For the latter instance, IPT required 9380 seconds to reach an average final tour length of 378,158 (0.033%).

The genetic iterated local search approach (GILS) [56] is similar to the MA presented in this paper. Due to the different hardware platform and different running times, a comparison is not possible. GILS delivers very impressive results for instance pr2392: an average quality of 0.006%—the optimum is found 3 out of 10 times—is achieved in 1635 seconds on a Fujitsu S-4/5 workstation (microSPARCII 110 MHz). The average final quality for att532 and rat783 is 0.056% and 0.022% found in 113 and 103 seconds, respectively. However, the MA is able to find the optimum for fl3795 while the optimum could not be found after 26,958 seconds by the GILS.

All other heuristics proposed for the TSP, such as simulated annealing [50, 111], tabu search [28], ant colonies [22, 36, 106], artificial neural networks [23, 76, 92], search space smoothing [43], and perturbation [18] have been applied only to rather small problems from TSPLIB or to randomly generated problems. None of these heuristics has been applied to TSPLIB instances with between 3000 and 4000 cities.

The branch and cut approach by Applegate *et al.* in [3, 4] required 80,829 seconds for pcb3038, and 69,886 seconds for fl3795 on a Compaq XP1000 (500 MHz) machine, which is more than two times faster than a 500 MHz Pentium III.

4.4.3 Results on large traveling salesman problem instances

Finally, the MA has been applied to the largest instances in TSPLIB. For these instances, there are no published results of other evolutionary methods known to us. Table 8 shows the tour length of the optimum solutions as well as the computation times required by branch and cut to find the optimum [4] on a Pentium II (600 MHz) machine. For the three largest problems, the optimum solutions are not known. Therefore, the bounds in which the optimum is known to lie is provided instead of the optimum value itself.

To demonstrate the applicability of the algorithms to very large instances, the MA has been applied to the seven problems listed in Table 8. With the same parameters as above, but with termination before the third restart, the MAs were run with a population size P of 10, 20, and 40. The results are presented in Table 9. For each population size (P) and each instance, the average number of generations (gen), the average final tour length and percentage access over the optimum or the lower bound (quality), the standard deviation (sdev.), and the average time (t) in seconds of 10 runs is displayed.

Instance	Optimum/Bounds	Time to find optimum Pentium II 600 MHz
fnl4461	182566	≈ 108044 sec
pla7397	23260728	≈ 867661 sec
rl11849	923288	≈ 313 days
usa13509	19982859	≈ 8 years
d18512	[645198, 645255]	– open –
pla33810	[66005185, 66059941]	– open –
pla85900	[142307500, 142409553]	– open –

Table 8. The largest instances in TSPLIB.

P	Instance	gen	quality	sdev.	t in s
10	fnl4461	291	183762.7 (0.655%)	192.1	105
	pla7397	887	23328499.5 (0.291%)	21931.7	802
	rl11849	314	931333.5 (0.871%)	1417.2	417
	usa13509	466	20186311.8 (1.018%)	17135.1	790
	d18512	379	653474.3 (1.283%)	381.3	930
	pla33810	1386	66575838.8 (0.864%)	57687.2	3443
	pla85900	2216	143596390.7 (0.906%)	103234.6	12314
20	fnl4461	528	183366.3 (0.438%)	163.7	294
	pla7397	1155	23307621.7 (0.202%)	14120.4	1860
	rl11849	536	928115.5 (0.523%)	795.8	1006
	usa13509	1082	20125182.2 (0.712%)	27980.9	2422
	d18512	1226	650803.2 (0.869%)	477.8	2873
	pla33810	3832	66321344.7 (0.479%)	45162.4	11523
	pla85900	9069	142986675.5 (0.477%)	79510.3	52180
40	fnl4461	856	183047.1 (0.263%)	82.2	742
	pla7397	1185	23294046.2 (0.143%)	12538.2	3789
	rl11849	861	926253.7 (0.321%)	605.5	2503
	usa13509	1936	20057767.0 (0.375%)	10176.8	6638
	d18512	2091	649354.6 (0.644%)	501.6	7451

Table 9. Performance of MA using GX on the largest instances in TSPLIB.

The results show that a running time smaller than an hour is sufficient to reach a quality of less than 1% for all problems except the largest one. For the latter, the running time increases to 12,000 seconds. Increasing the population size increases the final solution quality, but running times increase drastically. In the extreme case of the largest problem, the running times grow 4.2 times from 12,314 to 52,180 seconds. In most other cases the running time grows less than 3 times. It can be observed

that the pla-problems are better solved than the other instances with respect to the solution quality.

5. Conclusions

In this paper, the fitness landscape of various (euclidean) traveling salesman problem (TSP) instances has been investigated. The autocorrelation analysis described in this paper is well suited for finding the most effective family of local search algorithms, but it does not allow predicting the performance of meta-heuristics based on local search. Therefore, a fitness distance correlation analysis of local optima has been conducted on various TSP landscapes. It has been shown that there are different types of landscapes although the majority of instances have common characteristics: locally optimum tours have more than 3/4 of the edges in common. Thus, the local minima are contained in a small fraction of the search space. Fractal instances like the ones studied in this work are artificial, they have highly correlated landscapes and are therefore easily solved by well-known improvement heuristics. Although they are of interest in the analysis of heuristics [82], they are not well suited for testing highly effective heuristic approaches for the TSP. Random instances in which the cities are uniformly distributed have higher correlated local optima with respect to fast 2-opt and Lin–Kernighan (LK) local search than others based on real city coordinates. The local optima of instances in which the cities form clusters—as found in the application of drilling holes in printed circuit boards—have even lower correlation of tour length and distance to the global optimum. These instances belong to the hardest type of instances from the viewpoint of heuristics for the TSP.

The high correlation of tour length and distance to the optimum of the local optima in the TSP landscape is an indicator for a good performance of recombination-based search algorithms, since recombination is capable of exploiting this correlation in an efficient way. However, for the TSP, an effective combination of local search and mutation exists—iterated local search. In an extensive study, several recombination operators, including a newly proposed generic greedy recombination operator (GX), are compared against each other in a memetic algorithm (MA) framework. The MAs show significant performance differences if a simple fast 2-opt local search is employed. For MAs with the sophisticated LK local search, the results lie much closer together. The study has shown that respectfulness is the most important property of a recombination operator. Furthermore, the MA with the newly proposed GX operator has been shown to outperform all its competitors: MAs with DPX or MPX recombination, MAs with nonsequential four change mutation, and iterated local search.

MAs with DPX and GX recombination and mutation have been applied to various instances contained in TSPLIB to show robustness

and scalability of the approach. While for problems with up to 1000 cities the optimum could be found in all runs in an average time of less than two minutes on a state of the art personal computer, for the larger instances much more time was required to find the optimum solution. However, for a problem size up to 3795, the optimum could be found in less than two hours. Compared to other proposed approaches, the MA appears to be superior in average solution quality and running times. Finally, the MA with GX has been applied to very large instances of up to 85,900 cities and is thus the first meta-heuristic known to us which can tackle very large problems.

There are several issues for future research. Most importantly, a parallel implementation of the MA is desired to reduce the running time on large instances ($n > 10{,}000$) and to allow performing more experiments. Regional parallelization models of evolutionary algorithms (EAs) using subpopulations and migration may enhance the overall performance of the memetic approach. Sophisticated data structures are required to solve very large problems, and we believe that performance can be increased significantly if a strong effort is made to tune the LK local search, as recent developments suggest [44].

References

[1] E. Angel and V. Zissimopoulos, "Autocorrelation Coefficient for the Graph Bipartitioning Problem," *Theoretical Computer Science*, **191** (1998) 229–243.

[2] D. Applegate, R. Bixby, V. Chvátal, and B. Cook, "Finding Cuts in the TSP (A preliminary report)," Technical Report 95-05 (DIMACS, 1995).

[3] D. Applegate, R. Bixby, V. Chvátal, and W. Cook, "On the Solution of Traveling Salesman Problems," *Documenta Mathematica*, **Extra Volume ICM III** (1998) 645–656.

[4] D. Applegate, R. Bixby, V. Chvátal, and W. Cook, "Concorde Benchmarks on TSPLIB Instances," W. M. Keck Center for Computational Discrete Optimization, Rice University, Houston, USA, 2000. http://www.keck.caam.rice.edu/concorde/bench.html.

[5] J. L. Bentley, "Experiments on Traveling Salesman Heuristics," in *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*, 1990.

[6] J. L. Bentley, "*K*-d-Trees for Semidynamic Point Sets," in *Proceedings of the Sixth Annual ACM Symposium on Computational Geometry*, 1990.

[7] J. L. Bentley, "Fast Algorithms for Geometric Traveling Salesman Problems," *ORSA Journal on Computing*, **4**(4) (1992) 387–411.

[8] H. Bersini, M. Dorigo, S. Langerman, G. Seront, and L. Gambardella, "Results of the First International Contest on Evolutionary Optimisation (1st ICEO)," in *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation* (IEEE Press, Piscataway, NJ, 1996).

[9] K. D. Boese, *Models for Iterative Global Optimization*, PhD thesis (University of Carlifornia, Los Angeles, USA, 1996).

[10] K. Boese, "Cost versus Distance in the Traveling Salesman Problem," Technical Report TR-950018 (UCLA Computer Science Department, Los Angeles, CA, 1995).

[11] K. Boese, A. Kahng, and S. Muddu, "A New Adaptive Multi-start Technique for Combinatorial Global Optimizations," *Operations Research Letters*, **16** (1994) 101–113.

[12] L. B. Booker, D. E. Goldberg, and J. H. Holland, "Classifier Systems and Genetic Algorithms," *Artificial Intelligence*, **40** (1989) 235–282.

[13] R. M. Brady, "Optimization Strategies Gleaned from Biological Evolution," *Nature*, **317** (1985) 804–806.

[14] H. Braun, "On Solving Traveling Salesman Problems by Genetic Algorithms," in *Parallel Problem Solving from Nature—Proceedings of First Workshop, PPSN 1*, edited by H.-P. Schwefel and R. Männer (Springer-Verlag, Berlin and Dortmund, 1–3 October, 1991).

[15] T. G. Bui and B. R. Moon, "A New Genetic Approach for the Traveling Salesman Problem," in *Proceedings of the First IEEE Conference on Evolutionary Computation* (IEEE Press, 1994).

[16] N. Christofides, "The Traveling Salesman Problem," in *Combinatorial Optimization*, edited by N. Christofides, A. Mingozzi, P. Toth, and C. Sandi (Wiley and Sons, 1979).

[17] G. Clarke and J. W. Wright, "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points," *Operations Research*, **12** (1964) 568–581.

[18] B. Codenotti, G. Manzini, L. Margara, and G. Resta, "Perturbation: An Efficient Technique for the Solution of Very Large Instances of the Euclidean TSP," Technical Report TR-93-035 (International Computer Science Institute, Berkeley, CA, 1993).

[19] G. A. Croes, "A Method for Solving Traveling Salesman Problems," *Operations Research*, **5** (1958) 791–812.

[20] L. Davis, "Applying Adaptive Algorithms to Epistatic Domains," in *Proceedings of the International Joint Conference on Artificial Intelligence* (Morgan Kauffman, 1985).

[21] L. Davis, *Genetic Algorithms and Simulated Annealing* (Pitman, London, 1987).

[22] M. Dorigo and L. M. Gambardella, "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem," *IEEE Transactions on Evolutionary Computation*, **1**(1) (1997) 53–66.

[23] R. Durbin, R. Szeliski, and A. Yuille, "An Analysis of the Elastic Net Approach to the Traveling Salesman Problem," *Neural Computation*, **1** (1989) 348–358.

[24] J. Dzubera and D. Whitley, "Advanced Correlation Analysis of Operators for the Traveling Salesman Problem," in *Parallel Problem Solving from Nature—Proceedings of the Third Workshop, PPSN III*, edited by H.-P. Schwefel and R. Männer (Springer-Verlag, Berlin and Dortmund, 1994).

[25] L. J. Eshelman and J. D. Schaffer, "Preventing Premature Convergence in Genetic Algorithms by Preventing Incest," in *Proceedings of the Fourth International Conference on Genetic Algorithms* (Morgan Kaufmann, 1991).

[26] L. Eshelman, K. Mathias, and J. D. Schaffer, "Convergence Controlled Variation," in *Foundations of Genetic Algorithms 4*, edited by R. K. Belew and M. D. Vose (Morgan Kaufmann, 1997).

[27] L. Eshelman, "The CHC Adaptive Search Algorithm: How to Have Safe Search When Engaging in Nontraditional Genetic Recombination," in *Foundations of Genetic Algorithms*, edited by G. J. E. Rawlings (Morgan Kaufmann, 1991).

[28] C.-N. Fiechter, "A Parallel Tabu Search Algorithm for Large Traveling Salesman Problems," *Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science*, **51** (1994) 243–267.

[29] M. M. Flood, "The Traveling–Salesman Problem," *Operations Research*, **4** (1956) 61–75.

[30] D. B. Fogel, "Applying Evolutionary Programming to Selected Traveling Salesman Problems," *Cybernetics and Systems*, **24** (1993) 27–36.

[31] D. B. Fogel, "An Evolutionary Approach to the Traveling Salesman Problem," *Biological Cybernetics*, **60** (1988) 139–144.

[32] B. R. Fox and M. B. McMahon, "Genetic Operators for Sequencing Problems," in *Foundations of Genetic Algorithms*, edited by G. J. E. Rawlings (Morgan Kaufmann, 1991).

[33] M. L. Fredman, D. S. Johnson, L. A. McGeoch, and G. Ostheimer, "Data Structures for Traveling Salesmen," *Journal of Algorithms*, **18** (1995) 432–479.

[34] B. Freisleben and P. Merz, "A Genetic Local Search Algorithm for Solving Symmetric and Asymmetric Traveling Salesman Problems," in *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation*, edited by T. Bäck, H. Kitano, and Z. Michalewicz (IEEE Press, Piscataway, NJ, 1996).

[35] B. Freisleben and P. Merz, "New Genetic Local Search Operators for the Traveling Salesman Problem," in *Proceedings of the Fourth International Conference on Parallel Problem Solving from Nature—PPSN IV*, edited by H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel (Springer, Berlin, 1996).

[36] L. M. Gambardella and M. Dorigo, "Ant-Q: A Reinforcement Learning Approach to the Traveling Salesman Problem," in *Proceedings of the Twelfth International Conference on Machine Learning* (Morgan Kaufmann, 1995).

[37] D. E. Goldberg and J. R. Lingle, "Alleles, Loci, and the Traveling Salesman Problem," in *Proceedings of an International Conference on Genetic Algorithms and their Applications* (Carnegie Mellon Publishers, 1985).

[38] M. Gorges-Schleuter, "ASPARAGOS: An Asynchronous Parallel Genetic Optimization Strategy," in *Proceedings of the Third International Conference on Genetic Algorithms*, edited by J. D. Schaffer (Morgan Kaufmann, 1989).

[39] M. Gorges-Schleuter, *Genetic Algorithms and Population Structures—A Massively Parallel Algorithm*, PhD thesis (Universität Dortmund, Germany, 1990).

[40] M. Gorges-Schleuter, "Asparagos96 and the Traveling Salesman Problem," in *Proceedings of the 1997 IEEE International Conference on Evolutionary Computation* (IEEE Press, 1997).

[41] J. Grefenstette, R. Gopal, B. Rosimaita, and D. V. Gucht, "Genetic Algorithms for the Traveling Salesman Problem," in *Proceedings of an International Conference on Genetic Algorithms and their Applications* (Carnegie Mellon Publishers, 1985).

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

[43] J. Gu and X. Huang, "Efficient Local Search With Search Space Smoothing: A Case Study of the Traveling Salesman Problem (TSP)," *IEEE Transactions on Systems, Man, and Cybernetics*, **24** (1994) 728–735.

[44] K. Helsgaun, "An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic," *European Journal of Operational Research*, **126**(1) (2000) 106–130.

[45] M. Herdy, "Application of the Evolutionsstrategie to Discrete Optimization Problems," in *Parallel Problem Solving from Nature*, edited by H.-P. Schwefel and R. Männer (Springer, 1991).

[46] W. D. Hillis, "Co-evolving Parasites Improve Simulated Evolution as an Optimization Procedure," in *Artificial Life II*, edited by C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen (Addison-Wesley, 1992).

[47] J. Holland, *Adaptation in Natural and Artificial Systems* (University of Michigan Press, 1975).

[48] D. Holstein and P. Moscato, "Memetic Algorithms using Guided Local Search: A Case Study," in *New Ideas in Optimization*, edited by D. Corne, M. Dorigo, and F. Glover (McGraw-Hill, London, 1999).

[49] A. Homaifar, S. Guan, and G. E. Liepins, "A New Approach to the Traveling Salesman Problem by Genetic Algorithms," in *Proceedings of the Fifth International Conference on Genetic Algorithms* (Morgan Kaufmann, 1993).

[50] C.-S. Jeong and M.-H. Kim, "Fast Parallel Simulated Annealing for Traveling Salesman Problem on SIMD Machines with Linear Interconnections," *Parallel Computing*, **17**(2–3) (1991) 221–228.

[51] P. Jog, J. Y. Suh, and D. V. Gucht, "The Effects of Population Size, Heuristic Crossover and Local Improvement on a Genetic Algorithm for the Travelling Salesman Problem," in *Proceedings of the Third International Conference on Genetic Algorithms* (Morgan Kaufmann, 1989).

[52] P. Jog, J. Y. Suh, and D. V. Gucht, "Parallel Genetic Algorithms Applied to the Traveling Salesman Problem," *SIAM Journal on Optimization*, **1**(4) (1991) 515–529.

[53] D. S. Johnson and L. A. McGeoch, "The Traveling Salesman Problem: A Case Study," in *Local Search in Combinatorial Optimization*, edited by E. H. L. Aarts and J. K. Lenstra (Wiley and Sons, New York, 1997).

[54] T. Jones and S. Forrest, "Fitness Distance Correlation as a Measure of Problem Difficulty for Genetic Algorithms," in *Proceedings of the Sixth International Conference on Genetic Algorithms*, edited by L. J. Eshelman (Morgan Kaufmann, 1995).

[55] R. M. Karp, "A Patching Algorithm for the Nonsymmetric Traveling Salesman Problem," *SIAM Journal on Computing*, **8**(4) (1979) 561–573.

[56] K. Katayama and H. Narihisa, "Iterated Local Search Approach using Genetic Transformation to the Traveling Salesman Problem," in *GECCO-1999: Proceedings of the Genetic and Evolutionary Computation Conference*, edited by W. Banzhaf (Morgan Kauffman, 1999).

[57] S. A. Kauffman, *The Origins of Order: Self-Organization and Selection in Evolution* (Oxford University Press, 1993).

[58] S. A. Kauffman and S. Levin, "Towards a General Theory of Adaptive Walks on Rugged Landscapes," *Journal of Theoretical Biology*, **128** (1987) 11–45.

[59] B. Kernighan and S. Lin, "An Efficient Heuristic Procedure for Partitioning Graphs," *Bell Systems Journal*, **49** (1972) 291–307.

[60] S. Kirkpatrick and G. Toulouse, "Configuration Space Analysis of Travelling Salesman Problems," *Journal de Physique*, **46** (1985) 1277–1292.

[61] J. R. Koza, *Genetic Programming: On the Programming of Computers by Natural Selection* (MIT Press, Cambridge, 1992).

[62] J. R. Koza, *Genetic Programming II: Automatic Discovery of Reusable Programs* (MIT Press, Cambridge, 1994).

[63] B. Krakhofer and P. F. Stadler, "Local Minima in the Graph Bipartitioning Problem," *Europhysics Letters*, **34** (1996) 85–90.

[64] G. E. Liepins and M. R. Hilliard, "Greedy Genetics," in *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms* (Lawrence Erlbaum, 1987).

[65] S. Lin, "Computer Solutions of the Travelling Salesman Problem," *Bell System Technical Journal*, **44** (1965) 2245–2269.

[66] S. Lin and B. Kernighan, "An Effective Heuristic Algorithm for the Traveling Salesman Problem," *Operations Research*, **21** (1973) 498–516.

[67] K.-T. Mak and A. J. Morton, "Distances between Traveling Salesman Tours," *Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science*, **58** (1995) 281–291.

[68] F. Margot, "Quick Updates for *p*-opt TSP Heuristics," *Operations Research Letters*, **11** (1992) 45–46.

[69] K. Mathias and D. Whitley, "Genetic Operators, the Fitness Landscape and the Traveling Salesman Problem," in *Proceedings of the Second International Conference on Parallel Problem Solving from Nature—PPSN 2*, edited by R. Männer and B. Manderick (Elsevier Science Publishers, 1992).

[70] P. Merz, *Memetic Algorithms for Combinatorial Optimization Problems: Fitness Landscapes and Effective Search Strategies*, PhD thesis (Department of Electrical Engineering and Computer Science, University of Siegen, Germany, 2000).

[71] P. Merz and B. Freisleben, "Genetic Local Search for the TSP: New Results," in *Proceedings of the 1997 IEEE International Conference on Evolutionary Computation*, edited by T. Bäck, Z. Michalewicz, and X. Yao (IEEE Press, Piscataway, NJ, 1997).

[72] P. Merz and B. Freisleben, "Fitness Landscapes and Memetic Algorithm Design," in *New Ideas in Optimization*, edited by D. Corne, M. Dorigo, and F. Glover (McGraw-Hill, London, 1999).

[73] P. Merz and B. Freisleben, "Fitness Landscape Analysis and Memetic Algorithms for the Quadratic Assignment Problem," *IEEE Transactions on Evolutionary Computation*, **4**(4) (2000) 337–352.

[74] P. Merz and B. Freisleben, "Fitness Landscapes, Memetic Algorithms and Greedy Operators for Graph Bi-Partitioning," *Evolutionary Computation*, **8**(1) (2000) 61–91.

[75] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs* (Springer, Berlin, 1996).

[76] C. Ming and L. Minghui, "Kohonen Neural Network-based Solution of TSP," *Mini-Micro Systems*, **15**(11) (1994) 35–39.

[77] A. Möbius, A. Diaz-Sanchez, B. Freisleben, M. Schreiber, A. Fachat, K. Hoffmann, P. Merz, and A. Neklioudov, "Two Physically Motivated Algorithms for Combinatorial Optimization: Thermal Cycling and Iterative Partial Transcription," *Computer Physics Communications*, **121–122**(1–3) (1999) 34–36.

[78] A. Möbius, B. Freisleben, P. Merz, and M. Schreiber, "Combinatorial Optimization by Iterative Partial Transcription," *Physical Review E*, **59**(4) (1999) 4667–4674.

[79] P. Moscato, "On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms," Technical Report C3P Report 826 (Caltech Concurrent Computation Program, California Institue of Technology, 1989).

[80] 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*, edited by M. Valero, E. Onate, M. Jane, J. L. Larriba, and B. Suarez (IOS Press, Amsterdam, 1992).

[81] P. Moscato and M. Norman, "Arbitrarily Large Planar ETSP Instances with Known Optimal Tours," *Pesquisa Operacional*, **15** (1995) 89–96.

[82] P. Moscato and M. Norman, "On the Performance of Heuristics on Finite and Infinite Fractal Instances of the Euclidean Traveling Salesman Problem," *INFORMS Journal on Computing*, **10**(2) (1998) 121–132.

[83] P. Moscato and F. Tinetti, "Blending Heuristics with a Population-based Approach: A Memetic Algorithm for the Traveling Salesman Problem," Technical Report CeTAD (CeTAD, Universitad Nacional de La Plata, 1994).

[84] P. Moscato, "Memetic Algorithms: A Short Introduction," in *New Ideas in Optimization*, edited by D. Corne, M. Dorigo, and F. Glover (McGraw-Hill, London, 1999).

[85] H. Mühlenbein, M. Gorges-Schleuter, and O. Krämer, "Evolution Algorithms in Combinatorial Optimization," *Parallel Computing*, **7** (1988) 65–88.

[86] H. Mühlenbein, "Evolution in Time and Space—The Parallel Genetic Algorithm," in *Foundations of Genetic Algorithms*, edited by G. J. E. Rawlins (Morgan Kaufmann Publishers, 1991).

[87] Y. Nagata and S. Kobayashi, "Edge Assembly Crossover: A High-power Genetic Algorithm for the Traveling Salesman Problem," in *Proceedings of the Seventh International Conference on Genetic Algorithms*, edited by T. Bäck (Morgan Kaufmann, 1997).

[88] M. G. Norman and P. Moscato, "The Euclidean Traveling Salesman Problem and a Space-Filling Curve," *Chaos, Solitons and Fractals*, **6** (1995) 389–397.

[89] 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 and their Applications: Proceedings of the Second International Conference on Genetic Algorithms* (Lawrence Erlbaum, 1987).

[90] I. Or, *Traveling Salesman-Type Combinatorial Problems and Their Relation to the Logistics of Regional Blood Banking*, PhD thesis (Northwestern University, Evanston, 1976).

[91] J. Paredis, "Coevolutionary Computation," *Artificial Life*, **2**(4) (1995) 355–375.

[92] J.-Y. Potvin, "The Traveling Salesman Problem: A Neural Network Perspective," *ORSA Journal on Computing*, **5** (1993) 328–348.

[93] N. Radcliffe and P. Surry, "Fitness Variance of Formae and Performance Prediction," in *Proceedings of the Third Workshop on Foundations of Genetic Algorithms*, edited by L. Whitley and M. Vose (Morgan Kaufmann, San Francisco, 1994).

[94] N. Radcliffe and P. Surry, "Formal Memetic Algorithms," in *Evolutionary Computing: AISB Workshop*, edited by T. Fogarty (Springer-Verlag, Berlin, 1994).

[95] C. R. Reeves, "Landscapes, Operators and Heuristic Search," *Annals of Operations Research*, **86** (1999) 473–490.

[96] G. Reinelt, "TSPLIB—A Traveling Salesman Problem Library," *ORSA Journal on Computing*, **3**(4) (1991) 376–384.

[97] G. Reinelt, *The Traveling Salesman: Computational Solutions for TSP Applications*, volume 840 of *Lecture Notes in Computer Science*, (Springer-Verlag, Berlin, 1994).

[98] S. Ronald, "Finding Multiple Solutions with an Evolutionary Algorithm," in *Proceedings of the 1995 IEEE International Conference on Evolutionary Computation* (IEEE Press, 1995).

[99] S. Ronald, "Distance Functions for Order-Based Encodings," in *Proceedings of the 1997 IEEE International Conference on Evolutionary Computation* (IEEE Press, 1997).

[100] G. Rudolph, "Global Optimization by Means of Distributed Evolution Strategies," in *Parallel Problem Solving from Nature—Proceedings of the First Workshop, PPSN 1*, edited by H. P. Schwefel and R. Männer (Springer, 1991).

[101] H.-P. Schwefel, *Numerische Optimierung von Computer–Modellen mittels der Evolutionsstrategie*, volume 26 of *Interdisciplinary Systems Research* (Birkhäuser Verlag, Basel, 1977).

[102] P. F. Stadler, "Towards a Theory of Landscapes," in *Complex Systems and Binary Networks*, edited by R. Lopéz-Peña, R. Capovilla, R. García-Pelayo, H. Waelbroeck, and F. Zertuche (Springer-Verlag, Berlin, New York, 1995).

[103] P. F. Stadler, "Landscapes and their Correlation Functions," *Journal of Mathematical Chemistry*, **20** (1996) 1–45.

[104] P. F. Stadler and W. Schnabl, "The Landscape of the Travelling Salesman Problem," *Physics Letters A*, **161** (1992) 337–344.

[105] T. Starkweather, S. McDaniel, K. Mathias, D. Whitley, and C. Whitley, "A Comparison of Genetic Sequencing Operators," in *Proceedings of the Fourth International Conference on Genetic Algorithms* (Morgan Kaufmann, 1991).

[106] T. Stützle, *Local Search Algorithms for Combinatorial Problems—Analysis, Improvements, and New Applications*, PhD thesis (FB Informatik, TU Darmstadt, 1998).

[107] J. Y. Suh and D. V. Gucht, "Incorporating Heuristic Information into Genetic Search," in *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms* (Lawrence Erlbaum, 1987).

[108] G. Tao and Z. Michalewicz, "Inver-over Operator for the TSP," in *Proceedings of the Fifth International Conference on Parallel Problem Solving from Nature—PPSN V*, edited by A.-E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel (Springer, 1998).

[109] N. L. J. Ulder, E. H. L. Aarts, H. J. Bandelt, P. J. M. van Laarhoven, *et al.*, "Genetic Local Search Algorithms for the Traveling Salesman Problems," in *Proceedings of the First International Conference on Parallel Problem Solving from Nature—PPSN I*, edited by H. P. Schwefel and R. Männer (Springer-Verlag, Berlin and Dortmund, 1991).

[110] C. L. Valenzuela, "Evolutionary Divide and Conquer (II) for the TSP," in *Proceedings of the Genetic and Evolutionary Computation Conference*, edited by W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith (Morgan Kaufmann, 1999).

[111] P. J. M. van Laarhoven and E. H. L. Aarts, *Simulated Annealing: Theory and Applications* (Kluwer Academic Publishers, 1987).

[112] T. Walters, "Repair and Brood Selection in the Traveling Salesman Problem," in *Proceedings of the Fifth International Conference on Parallel Problem Solving from Nature—PPSN V*, edited by A.-E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel (Springer, 1998).

[113] E. D. Weinberger, "Correlated and Uncorrelated Fitness Landscapes and How to Tell the Difference," *Biological Cybernetics*, **63** (1990) 325–336.

[114] D. Whitley, T. Starkweather, and D. Fuquay, "Scheduling Problems and Traveling Salesman: The Genetic Edge Recombination Operator," in *Proceedings of the Third International Conference on Genetic Algorithms* (Morgan Kaufmann, 1989).

[115] S. Wright, "The Roles of Mutation, Inbreeding, Crossbreeding, and Selection in Evolution," in *Proceedings of the Sixth Congress on Genetics* (Brooklyn, New York, 1932).