A Genetic Local Search Algorithm for Solving Symmetric and Asymmetric Traveling Salesman Problems

Bernd Freisleben and Peter Merz
Department of Electrical Engineering and Computer Science (FB 12)
University of Siegen
Hölderlinstr. 3, D–57068 Siegen, Germany
E-Mail: {freisleb,pmerz}@informatik.uni-siegen.de

Abstract— **The combination of local search heuristics and genetic algorithms is a promising approach for finding near-optimum solutions to the traveling salesman problem (TSP). In this paper, an approach is presented in which local search techniques are used to find local optima in a given TSP search space, and genetic algorithms are used to search the space of local optima in order to find the global optimum. New genetic operators for realizing the proposed approach are described, and the quality and efficiency of the solutions obtained for a set of symmetric and asymmetric TSP instances are discussed. The results indicate that it is possible to arrive at high quality solutions in reasonable time.**

I. Introduction

In the *Traveling Salesman Problem* (TSP) [18], [27], a number of cities with distances between them is given and the task is to find the minimum-length closed tour that visits each city once and returns to its starting point. A *symmetric* TSP (STSP) is one where the distance between any two cities A and B is equal to to the distance between B and A; in an *asymmetric* TSP (ATSP) these distances are different.

The TSP has become a standard testbed for combinatorial optimization methods which attempt to find near-optimum solutions to this NP–hard problem [10]. Several such approximation techniques, typically based on deterministic or probabilistic search heuristics, have been proposed in the literature, including the classical *local search* algorithms [27], *simulated annealing* [17], *threshold accepting* [8], *tabu search* [7], *elastic nets* [6], *neural networks* [1], *genetic algorithms* [11], [14], and *ant colonies* [9].

Although some of these approaches, when applied individually, have proven to be successful in producing quite good results, top quality solutions seem to require the combined efforts of several methods, particularly for large TSP instances. For example, near-optimum or optimal tours have been obtained by combinations of local search algorithms and simulated annealing [16], [20] and by combinations of local search algorithms and genetic algorithms (GA) [12], [23], [25], [29].

In this paper, we present a new combined local search/genetic algorithm approach to the TSP. The basic idea is to use a simple nearest-neighbor tour construction heuristic for creating the initial population of a GA, apply a local search algorithm to this initial population for producing a population of local optima, and let a GA operate on the search space of local optima to determine the global optimum.

The general approach is applied to both the symmetric and asymmetric TSP, but the GA operators are adapted to the individual characteristics of each of the two problem classes. A new crossover operator has been developed for enabling the GA to perform particular "jumps" within the search space of local minima. Its design is based on the observation that the solution space of a typical TSP has a "globally convex" or "big valley" character, such that the optimal solution can be found near the center of a single valley of near-optimum solutions with approximately equal distance to each other [4]. Furthermore, there is a novel mutation operator which performs random jumps within the neighborhood of a local optimum, and a new replacement scheme which attemps to avoid the emergence of highly similar or identical individuals. The feasibility of the approach is demonstrated by presenting the quality and runtime performance of the solutions obtained for several STSP and ATSP instances.

The paper is organized as follows. Section 2 describes the combined local search/genetic algorithm approach for the symmetric TSP, and section 3 is devoted to the asymmetric case. In section 4, performance results for several TSP instances are presented. Section 5 concludes the paper and outlines areas for future research.

II. The Genetic Algorithm for the STSP

The desire to improve the rather poor performance achievements of pure GA approaches in TSP applications has motivated several researchers to build additional heuristic elements into GAs. The proposals for such "heuristic" GAs include tour construction heuristics for generating the initial population of a GA [13], tour improvement heuristics for producing local optima [15], [24], and special heuristic crossover operators tailored to the characteristics of the TSP [14], [21], [23], [28], [29]. Since these studies seem to indicate that the only way to obtain results comparable to good conventional local search techniques is to incorporate TSP specific knowledge into a GA, our approach is based on rigorously using heuristic information at various steps of the GA optimization process. The proposed algorithm for solving the symmetric TSP is

0-7803-2902-3/96/ $4.00 © 1996 IEEE

```
procedure STSP-GA;
  begin
    initialize population P with Nearest-Neighbor heuristic;
    foreach individual i ∈ P do Lin-Kernighan-Opt(i);
    repeat
      for i = 0 to #crossovers do
        select two parents i_a, i_b ∈ P randomly;
        i_c := DPX-STSP(i_a, i_b);
        Lin-Kernighan-Opt(i_c);
        with predefined probability do Mutation-STSP(i_c);
        replace an individual of P by i_c;
      end;
    until converged;
  end;
```

Fig. 1. The GA for the symmetric TSP

shown in Fig. 1.

The first step of the algorithm is to create an initial population P of tours, i.e. permutations of the N integers representing the cities contained in a given STSP. A simple *nearest-neighbor* tour construction heuristic [27], applied to p different randomly chosen starting points, is used for that purpose.

Each of the p individuals of the initial GA population P is then exposed to the Lin-Kernighan (LK) tour improvement heuristic [19]. The result of the LK local search is a population of p individuals which represent locally optimal solutions. The LK heuristic, which is an efficient realization of a k-Opt procedure with k variable in each step [19], has been selected due to its effectiveness in producing high quality solutions; it may be regarded as the "uncontested champion" of local search techniques for the symmetric TSP for the past twenty years.

The GA starts operating on this population by selecting two individuals at random as the inputs to the crossover operator. Selection methods based on the fitness of the individuals are not appropriate, because the high degree of similarity of such individuals would possibly lead to an offspring identical to one of its parents.

After selection, a new crossover operator, called *distance preserving crossover* (DPX), is applied. The functionality of the DPX is motivated by Boese's analysis of the TSP search space [4]. Defining the distance between two tours as the number of edges that are contained in the first but not in the second tour, he observed that the average distance between locally optimal tours is similar to the average distance between a locally optimal tour and the global optimum. Consequently, in order to allow the GA to perform jumps in the search space which most likely lead to the global optimum, the aim of the DPX is to produce an offspring which has the same distance to each of its parents as one parent to the other. Its functionality is shown in Fig. 2.

The DPX works as follows: the contents of the first parent is copied to the offspring and all edges that are not

```
procedure DPX-STSP(i_a, i_b);
  begin
    i_c := i_a;
    delete all edges in i_c that are not contained in i_b;
    greedy_reconnect-STSP(i_c);
  end;
```

Fig. 2. The DPX crossover for the STSP

in common with the other parent are deleted. The resulting parts of the broken tour are reconnected without using any of the edges which are contained in only one 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 endpoints of the remaining tour fragments is taken and the edge (i, k) is added to the tour, provided that (i, k) is not contained in one of the two parents. This process continues until all fragments have been reconnected. In order to illustrate the DPX operator, let us consider an example (see Fig. 3).

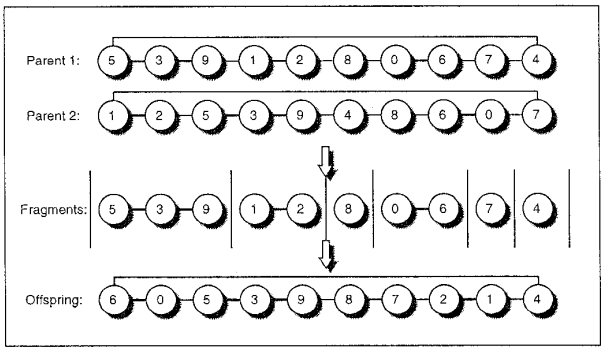

Fig. 3. Illustration of the DPX crossover

Suppose that the two parents shown in Fig. 3 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 Fig. 3 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 cities which are either start or endpoints of not yet visited tour fragments, i.e. {5,9,1,2,4}, is determined. City 8 and city 7 are not contained in this set, because it is not desirable to reinsert edge $(0,8)$ or edge $(0,7)$, since they are contained in either 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 candidate cities is {2,8,7}. The process 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.

Since the offsprings produced by the DPX are in general not local optima, the *LK* heuristic is applied to each offspring in the next step of the algorithm to again obtain a locally optimal tour.

Mutation is then performed with a small probability on the locally optimized offspring. The mutation operator developed for that purpose modifies the tour by a random *non-sequential 4-change*, also called a *double bridge kick move* [19], [20]. The non-sequential 4-change as the mutation operator can be seen as a random transformation of a tour T into a tour T' with the property that the distance between T and T' is 4. The 4-change has been selected, because it is the smallest possible non-sequential change, i.e. it does not change the tour length considerably and thus will bring us not too far away from a good solution. Furthermore, since *LK* performs only sequential changes, i.e. the endpoint of a removed edge is the startpoint of a newly inserted edge, the probability of eliminating the effect of the 4-change by two or more *LK* runs (and therefore falling back into the previous local minimum) is low. As part of the mutation operator, the tour modified by the non-sequential 4-change is again transformed into a local minimum by applying the *LK* heuristic, as shown in Fig. 4.

```
procedure Mutation-STSP(i);
  begin
    non-sequential 4-change(i);
    Lin-Kernighan-Opt(i);
  end;
```

Fig. 4. The mutation operator for the STSP

Finally, the resulting locally optimal tour replaces an individual of the current population before the next iteration of the algorithm is executed.

The replacement scheme is important to maintain a sufficient degree of diversity within the population, which in turn is required to avoid premature convergence of the GA. In an environment where the search space of local optima is investigated, the problem of premature convergence is even more severe. Our replacement scheme to maintain the population diversity works as follows. First, the individual of the current population which is most similar (in terms of the distance) to the offspring to be included is identified. If the distance between them is below a predefined small threshold, this individual is replaced by the new offspring, unless this individual is the currently best individual, in which case it is only replaced if the new offspring has a higher fitness; otherwise the individual with the worst fitness in the current population will be replaced.

The proposed GA approach may be considered as a generalization of other (non-GA) optimization methods which have been shown to produce high quality solutions to the STSP. For example, setting the number of generations to 0 is equivalent to performing a *multi-start local search* [5], where the individual with the highest fitness represents the solution. Furthermore, the *Large Step Markov Chains* approach [20] and the *Iterated-LK* technique are special cases of our GA approach, since they are obtained by setting the population size and the mutation rate to 1, and the crossover rate to 0. Thus, our proposal promises to be more powerful than the these approaches, which are among the best heuristic approaches suggested for solving the STSP.

III. The Genetic Algorithm for the ATSP

Since the ATSP represents a generalization of the STSP, it is more difficult to solve. This is documented by the fact that optimal solutions are currently only available for instances with up 400 cities [26]. Although it is possible to transform (the directed graph of) a given ATSP into (an undirected graph of) an equivalent STSP, local search heuristics such as 2-Opt usually fail to produce satisfactory results on the resulting undirected graph [27]. Therefore, in order to solve the ATSP directly, the operators of the general GA appproach presented in the previous section must be adapted to the properties of the ATSP. The proposed algorithm for solving the asymmetric TSP is shown in Fig. 5.

The approach for solving the ATSP starts with p nearest-neighbor runs to create p individuals for the initial population. The *fast-3-Opt* tour improvement heuristic [2] is subsequently used to transform the individuals of the initial population into local optima. The reason why *LK*, used in the STSP, does not produce satisfactory results is that it includes 2-Opt moves; 2-Opt yields directional changes of tour fragments which in case of the ATSP may alter the tour length unpredictably (but which are irrelevant in the STSP). Since 3-Opt heuristics take a tour fragment and reinsert it at another position without reversing the order in which the cities are visited, they are more suitable as local search procedures for the ATSP, because unpredictable tour length changes are avoided. The *fast-3-Opt* variant

```
procedure ATSP-GA;
  begin
    initialize population P with Nearest-Neighbor heuristic;
    foreach individual i ∈ P do fast-3-Opt(i);
    repeat
      for i = 0 to #crossovers do
        select two parents i_a, i_b ∈ P randomly;
        i_c := DPX-ATSP(i_a, i_b);
        fast-3-Opt(i_c);
        with predefined probability do Mutation-ATSP(i_c);
        replace an individual of P by i_c;
      end;
    until converged;
  end;
```

Fig. 5. The GA for the asymmetric TSP

used in our approach is based on two fixed radius nearest neighbor searches to determine the most suitable cities for attempting to perform only two particular 3-Opt moves out of the eight possible 3-Opt moves.

The algorithm then proceeds analogously to the one for the symmetric TSP. After random selection of two individuals from the population of locally optimal tours, the DPX for the ATSP, as shown in Fig. 6, is applied.

```
procedure DPX-ATSP(i_a, i_b);
  begin
    i_c := i_a;
    delete all edges in i_c that are not contained in i_b;
    greedy_reconnect-ATSP(i_c);
  end;
```

Fig. 6. The DPX crossover for the ATSP

It is identical to the DPX for the STSP, except for the greedy reconnection procedure which has been suitably modified to account for the fact that the ATSP is based on a directed graph. After having performed the crossover, the fast-3-Opt described above is used to convert the resulting offspring into a local optimum.

The mutation operator has also been modified for the ATSP, as shown in Fig. 7.

```
procedure Mutation-ATSP(i);
  begin
    let k be a random number between 4 and 7;
    remove k randomly choosen edges from i;
    greedy_reconnect-ATSP(i);
    fast-3-Opt(i);
  end;
```

Fig. 7. The mutation operator for the ATSP

It is based on removing k randomly chosen edges from the tour ($4 \leq k \leq 7$, k random), and using a greedy reconnection procedure similar to the one for the crossover to connect each tour fragment with the nearest not yet reconnected fragment. In other words, a variable k-change with k between 4 and 7 is performed; such a k-change is more powerful than the non-sequential 4-change used in the STSP. Finally, a fast-3-Opt is applied to the resulting mutated tour in order to obtain a local minimum.

The repetition of the GA cycle, including the replacement of an individual of the current population by the offspring produced is performed as described in the algorithm for the STSP.

IV. Implementation and Results

The algorithms were implemented in *C++* on a DEC Alpha workstation under OSF/1 [22], using some of the features provided by the GALIB library (version 2.3.2) [30].

Detailed information about the development of the solution qualities in each of the experiments conducted are given elsewhere [3]. Table I summarizes the final results obtained by running the STSP-GA on several symmetric TSP instances containing between 51 and 1577 cities, taken from the TSPLIB [26]. In the table, *eval* denotes the total number of evaluations of individuals, *gen* denotes the number of generations performed, *best* shows the length of the best tour found together with its deviation from the known optimum in percent, *average* displays the same information for the average of r runs ($r = 20$ for eil51.tsp, kroA100.tsp, d198.tsp, att532.tsp, $r = 10$ for rat783.tsp, fl1577.tsp), and *time* shows the computation times on a DEC AlphaStation (175 MHz). Since for the last problem instance (fl1577.tsp) the optimal tour length is still unknown, the deviation from the lower bound (included in the TSPLIB) is given instead. The number of evaluations *eval* is determined by

$$eval = p + p/2 \cdot gen$$

where p represents the population size ($p = 10$ for eil51.tsp, kroA100.tsp, d198.tsp, $p = 20$ for att532.tsp, rat783.tsp, fl1577.tsp). In the experiments, the following parameters were used: (a) crossover rate: 0.5 (i.e. $0.5 \cdot p$ new offsprings

TABLE I

THE RESULTS FOR SYMMETRIC TSP INSTANCES

STSP Results				
problem	eval/gen	best (quality)	average (quality)	time (hh:mm:ss)
eil51.tsp	100/18	426 (0.00 %)	426.0 (0.00 %)	00:00:06
kroA100.tsp	50/8	21282 (0.00 %)	21282.0 (0.00 %)	00:00:11
d198.tsp	100/18	15780 (0.00 %)	15780.0 (0.00 %)	00:04:13
att532.tsp	1000/98	27686 (0.00 %)	27699.2 (0.05 %)	01:41:16
rat783.tsp	1200/128	8806 (0.00 %)	8809.5 (0.04 %)	04:08:45
fl1577.tsp	1200/128	22286 (0.37 %)	22306.8 (0.46 %)	34:24:67

TABLE II

THE RESULTS FOR ASYMMETRIC TSP INSTANCES

ATSP Results				
problem	eval/gen	best (quality)	average (quality)	time (hh:mm:ss)
p43.atsp	60/1	2810 (0.00 %)	2810.00 (0.00 %)	00:00:10
ry48p.atsp	1000/48	14422 (0.00 %)	14440.0 (0.12 %)	00:00:30
ft70.atsp	1000/48	38673 (0.00 %)	38683.8 (0.03 %)	00:10:39
kro124p.atsp	1000/48	36230 (0.00 %)	36235.3 (0.01 %)	00:01:55
ftv170.atsp	4000/198	2755 (0.00 %)	2766.1 (0.40 %)	00:03:31

per generation), and (b) mutation rate: 0.2 (i.e. 20% of the population per generation are mutated).

The results shown in Table I demonstrate that the proposed approach is capable of producing high quality solutions for each of the TSP instances investigated. For all the instances, the optimal solutions have been found within a relatively small number of evaluations/generations, and as indicated by the average quality, the optimal solutions are found in the majority of runs. The best solution quality for fl1577.tsp shown in Table I is 0.37% above the lower bound; running the GA for about 200 generations leads to values of 0.31% and 0.38% for the best and average qualities, respectively. However, in this case about 58 hours of computation time are required.

In addition to the information shown in Table I, it is interesting to consider the increase of quality when executing the individual steps of the algorithm, as shown in Table III.

TABLE III

NEAREST-NEIGHBOR (NN) AND LIN-KERNIGHAN-OPT (LK)

STSP		
problem	*NN*	*NN-LK*
eil51.tsp	22.54 %	2.11 %
kroA100.tsp	16.60 %	0.45 %
d198.tsp	11.57 %	4.91 %
att532.tsp	18.48 %	1.49 %
rat783.tsp	21.27 %	2.19 %
fl1577.tsp	21.34 %	3.45 %

The length of the best tours created by the nearest-neighbor construction heuristic (*NN*) is on the average around 20% above the optimum. The subsequent use of the Lin-Kernighan heuristic (*LK*) yields solutions which are about 2–3% above the optimum, with computation times for NN–LK of up to 36 seconds (for fl1577.tsp). Thus, the GA is responsible for the smallest but hardest part of the optimization process, in order to reach the values shown in Table I.

Table II shows the results obtained by running the ATSP-GA on several asymmetric TSP instances containing between 43 and 170 cities, again taken from the TSPLIB [26]. In this case, the computation times have been measured on a DEC AlphaStation with 266 MHz. The population size for all instances is $p = 40$, and $r = 20$ runs have been used to determine the average values.

The optimal solutions for all asymmetric instances have been found; the average values demonstrate that the optima were obtained in most of the experiments. In contrast to the STSP where the computation times strongly increase with increasing problem sizes, the problem size does not necessarily constitute the only relevant factor for the asymmetric instances; the characteristics of the problem instances do have a large impact on the computation times, as exhibited by instance ft70.atsp which requires a longer time to compute than the larger instances.

It is worth mentioning that the performance spectrum produced by the nearest-neighbor search and the fast-3-Opt is wider than that generated by the heuristics for the STSP; Table IV shows the values obtained. The computation times for the heuristics in all experiments were less than 1 second. The results indicate that the GA for the ATSP is responsible for delivering higher tour length re-

TABLE IV
NEAREST-NEIGHBOR (NN) AND FAST-3-OPT

ATSP		
problem	*NN*	*NN-fast-3-Opt*
p43.atsp	0.50 %	0.21 %
ry48p.atsp	6.56 %	2.94 %
ft70.atsp	9.30 %	2.87 %
kro124p.atsp	17.20 %	5.32 %
ftv170.atsp	31.36 %	11.58 %

ductions than its counterpart for the STSP.

V. CONCLUSIONS

In this paper, an approach for approximately solving symmetric and asymmetric traveling salesman problems (TSP) has been presented. The approach is based on the combination of local search heuristics and genetic algorithms; local search techniques were used to efficiently find local optima in the TSP search space, and genetic algorithms were used to search the space of local optima to find the global optimum. The performance results presented for several symmetric and asymmetric TSP instances have shown that the approach is able to produce high quality solutions in reasonable time. The results obtained are superior to those published for any GA approaches known to us, and are comparable to those of top quality non-GA heuristic techniques.

There are several issues for future research. First, it would be interesting to investigate the properties of promising alternatives for some of the components used in the algorithms, such as a a modified Lin-Kernighan heuristic for the asymmetric TSP (instead of fast-3-Opt). Second, the efficiency of the implementation must be increased to reduce the computation times; in particular, a parallel version of the proposal would be desirable. Third, further tests of the algorithms on other possibly more complex TSP instances are required to provide a detailed assessment of the merits of the proposed approach.

REFERENCES

[1] E. H. L. Aarts and H. P. Stehouwer, "Neural Networks and the Travelling Salesman Problem," in *Proc. Int. Conf. on Artificial Neural Networks*, pp. 950–955, Springer-Verlag, 1993.

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

[3] H. Bersini, M. Dorigo, L. Gambarella, S. Langerman, and G. Seront, "First International Contest on Evolutionary Optimisation," *Proc. IEEE Int. Conf. on Evolutionary Computation*, Nagoya, Japan, 1996.

[4] K. Boese, "Cost versus Distance in the Traveling Salesman Problem," Tech. Rep. TR-950018, UCLA CS Department, 1995.

[5] K.D. Boese and S. Muddu, "A New Adaptive Multi-Start Technique for Combinatorial Global Optimizations," *Operations Research Letters 16*, pp. 101–113, 1994.

[6] R. Durbin, R. Szeliski, and A. Yuille, "An Analysis of the Elastic Net Approach to the Travelling Salesman Problem," *Neural Computation*, Vol. 1, pp. 348–358, 1989.

[7] L. Fiechter, "A Parallel Tabu Search Algorithm for Large Traveling Salesman Problems," *Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science*, 51, pp. 243–267, 1994.

[8] B. Freisleben and M. Schulte. "Combinatorial Optimization with Parallel Adaptive Threshold Accepting". In *Proc. 1992 European Workshop on Parallel Computing*, Barcelona, pp. 176–179. IOS Press, 1992.

[9] L. M. Gambardella and M. Dorigo, "Ant-Q: A Reinforcement Learning Approach to the Traveling Salesman Problem," in *Proc. 12th Int. Conf. on Machine Learning*, pp. 252–260, Morgan Kaufmann, 1995.

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

[11] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1989.

[12] M. Gorges-Schleuter, "ASPARAGOS: An Asynchronous Parallel Genetic Optimization Strategy," in *Proc. of the 3rd Int. Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, 1989.

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

[14] L. Homaifar, C. Guan, and G. Liepins, "A New Approach to the Traveling Salesman Problem by Genetic Algorithms," in *Proc. 5th Int. Conf. on Genetic Algorithms*, pp. 460–466, Morgan Kaufmann Publishers, 1993.

[15] P. Jog, J. Y. Suh, and D. van Gucht, "The Effects of Population Size, Heuristic Crossover and Local Improvement on a Genetic Algorithm for the Travelling Salesman Problem," in *Proc. 3rd Int. Conf. on Genetic Algorithms*, pp. 110–115, Morgan Kaufmann Publishers, 1989.

[16] D. S. Johnson, "Local Optimization and the Traveling Salesman Problem," in *Annual Int. Colloquium on Automata, Languages and Programming*, pp. 446–461, 1990.

[17] P. van Laarhoven and E. H. L. Aarts, *Simulated Annealing: Theory and Applications*. Kluwer Academic Publishers, 1987.

[18] E. Lawler, J. K. Lenstra and D. B. Shmoys, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. Wiley, 1985.

[19] S. Lin and B. Kernighan, "An Effective Heuristic Algorithm for the Travelling Salesman Problem," *Operations Research*, Vol. 21, pp. 498–516, 1973.

[20] O. Martin, S. W. Otto, and E. W. Felten, "Large-Step Markov Chains for the Traveling Salesman Problem," *Complex Systems*, Vol. 5, pp. 299–326, 1991.

[21] K. Mathias and D. Whitley, "Genetic Operators, the Fitness Landscape and the Traveling Salesman Problem," in *Parallel Problem Solving from Nature II*, (R. Männer and B. Manderick, eds.), pp. 219–228, Elsevier, 1992.

[22] P. Merz, "*Genetic Algorithms for Combinatorial Optimization Problems* (in German)". Master's Thesis, Department of Electrical Engineering and Computer Science, University of Siegen, Germany, 1996.

[23] H. Mühlenbein, "Parallel Genetic Algorithms, Population Genetics and Combinatorial Optimization," in *Proc. 3rd Int. Conf. on Genetic Algorithms*, (J. D. Schaffer, ed.), pp. 416–421, Morgan Kaufmann Publishers, 1989.

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

[25] H. Mühlenbein, M. Gorges-Schleuter, and O. Krämer, "Evolutionary Algorithms in Combinatorial Optimization," *Parallel Computing*, Vol. 7, pp. 65–88, 1988.

[26] G. Reinelt, "TSPLIB— A Traveling Salesman Problem Library," *ORSA Journal on Computing*, Vol. 3, No. 4, pp. 376–384, 1991.

[27] G. Reinelt, *The Traveling Salesman: Computational Solutions for TSP Applications*. Vol. 840 of *Lecture Notes in Computer Science*, Springer-Verlag, 1994.

[28] A. Y. C. Tang and K. S. Leung, "A Modified Edge Recombination Operator for the Travelling Salesman Problem," in *Parallel Problem Solving from Nature III*, (H.-P. Schwefel and R. Männer, eds.), pp. 180–188, Springer-Verlag, 1994.

[29] N. L. J. Ulder, E. H. L. Aarts, H. J. Bandelt, P. J. M. van Laarhoven, and E. Pesch, "Genetic Local Search Algorithms for the Traveling Salesman Problem," in *Parallel Problem Solving from Nature I*, (H. Schwefel and R. Männer, eds.), pp. 109–116, Springer-Verlag, 1990.

[30] M. Wall, "GALIB 2.3.2," http://lancet.mit.edu/ga, 1995.