New Genetic Local Search Operators for the Traveling Salesman Problem

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. In this paper, an approach is presented to incorporate problem specific knowledge into a genetic algorithm which is used to compute near-optimum solutions to traveling salesman problems (TSP). The approach is based on using a tour construction heuristic for generating the initial population, a tour improvement heuristic for finding local optima in a given TSP search space, and new genetic operators for effectively searching the space of local optima in order to find the global optimum. The quality and efficiency of solutions obtained for a set of TSP instances containing between 318 and 1400 cities are presented.

1 Introduction

The *traveling salesman problem* (TSP) is a well-known NP-hard combinatorial optimization problem [15] in which a salesman tries to find the shortest closed tour to visit a set of N cities under the condition that each city is visited exactly once. Since the computation times for finding optimal solutions grow exponentially with N, many heuristic approaches have been proposed in order to compute near-optimum solutions in reasonable time. Among these are special purpose heuristics applicable only to the TSP and general methods useful in a variety of optimization problems, such as *simulated annealing* [14], *threshold accepting* [3] and *genetic algorithms* [5]. In many cases, the search efficiency of the general methods is strongly increased when some form of domain knowledge about the TSP is provided to them.

In this paper, we present means of incorporating TSP domain knowledge into a genetic algorithm (GA) and propose an approach in which *local search* (also called *local hill-climbing*) techniques and GAs are combined. The basic idea of the approach is to let the GA perform its search in the space of local minima rather than in the search space of all feasible tours. New genetic operators which are adapted to the properties of the local minima search space and which are thus different from commonly known genetic operators for the TSP, will be presented to realize the approach.

The paper is organized as follows. Section 2 reviews proposals for integrating TSP domain knowledge into a GA. Section 3 presents the new GA approach for solving the TSP. Section 4 presents computational results. Section 5 concludes the paper and outlines areas for future research.

2 Incorporating TSP Domain Knowledge into GAs

Many studies have indicated that pure GA approaches are outperformed by the conventional heuristics developed for the TSP, particularly when the problem size increases. The desire to improve the performance of GAs in TSP applications has motivated several researchers to build additional heuristic elements into GAs [21, 27]. The main possibilities to augment GAs with such heuristics are as follows:

- Instead of generating random tours which form the individuals of the initial population of a GA, it is possible to use a tour construction heuristic, such as the *nearest neighbor* (NN) heuristic or various *insertion* heuristics [15, 24]. Since an entire population has to be created, the heuristic in use must be able to produce different tours. For example, the nearest neighbor heuristic allows to construct N different tours, where N is the number of cities; each city can act as the starting point for the heuristic.
- Genetic recombination offers a large potential for using heuristic elements. For example, instead of determining new edges for producing a valid offspring in the crossover operator on a purely random basis, it might be desirable to prefer short edges. It is also possible to replace mutation by one of the well-known tour improvement heuristics such as *2-opt*, *3-opt* [16] or the *Lin-Kernighan* (LK) heuristic [17]. Independent of mutation, tour improvement heuristics may alternatively be applied to some or all individuals of the current population.
- Tour improvement heuristics can also be used for postprocessing: the resulting population or just the best individual may be the target of an additional run of a tour improvement algorithm.

Fig. 1 summarizes the possibilities of incorporating TSP heuristics into a GA by showing a general "template" for constructing such a *heuristic GA*.

Step 1: create the initial population by a tour construction heuristic.
Step 2: apply a tour improvement heuristic to initial population.
Step 3: selection: select parents for mating.
Step 4: recombination: perform heuristic crossover.
Step 5: apply a tour improvement heuristic to offspring.
Step 6: mutation: mutate individuals with a given probability.
Step 7: replacement: replace some parents with new offsprings.
Step 8: if not converged go to Step 3.
Step 9: perform postprocessing by applying a tour improvement heuristic

Fig. 1. A heuristic GA template

Although the integration of TSP domain knowledge into GAs according to Fig. 1 seems to be a promising approach, several previous attempts to use TSP heuristics at particular steps of the template were rather discouraging. For example, Grefenstette [9] used a heuristic crossover and got solutions as far as 25%

above the optimum for a 50-cities TSP. In [8] he reports results of experiments with tour construction heuristics for generating the initial population as well as using a heuristic crossover and local hill-climbing as the mutation operator. These results were better than corresponding ones obtained with "blind" genetic search, but still worse than those produced by a simple 2-opt tour improvement heuristic. Suh and van Gucht [25] have applied 2-opt to some individuals of a GA population and reached a quality of 1.73% above the optimum for a 100-cities problem. In [11], Jog, Suh and van Gucht present better solutions for the same 100-cities problem: using 2-opt and *Or*-opt [15] as the tour improvement operator leads to a quality of 1.37% above the optimum on the average. However, the results obtained with these heuristic GA approaches are still rather poor when compared to conventional heuristics. For example, a nearest neighbor heuristic combined with a 2-opt improvement algorithm (implemented as part of our work) applied to same 100-cities instance as the one mentioned above is able to find a tour with a better quality (0.50% above the optimum) than the average quality of all heuristic GA approaches mentioned.

In [29], Whitley et al. show that their *edge recombination crossover* allows to solve a 30-cities problem to optimality and is superior to other "blind" recombination operators such as *PMX*, *cycle* and *order crossover*. The edge recombination operator has subsequently been improved by Mathias and Whitley [18] who called their enhanced versions *edge-2* and *edge-3*; edge-3 is superior to edge-2 and designed for larger TSP instances. The value obtained by using edge-3 in the 532-cities problem of Padberg and Rinaldi [22] has been reported to be 28979 (4.67%) on the average and 28752 (3.85%) as the best solution [18]. A modified version of edge-2, called *edge-NN*, has been shown by Tang and Leung [26] to be slightly superior to edge-3. Edge-NN incorporates greedy choices into the recombination step and additionally improves individuals by 2– and 3-changes. The tour lengths that have been achieved by edge-NN for the 532-cities problem are 27949 (0.95%) as the shortest tour and 28255 (2.06%) as the average value.

Mühlenbein [19, 21] has developed a parallel GA which is quite different to the ones described above. In this GA, each individual can be seen as an active entity which chooses itself a partner for mating and improves during its lifetime (i.e. local hill-climbing is performed). A new crossover operator, called MPX (*maximum preservative crossover*), has been proposed for recombination. The population structure also differs from the one of a traditional GA, as described by Gorges-Schleuter [6, 7]. In this approach, the improvement rate is set to 1, hence all individuals are local minima; the tour improvement operator is 2-opt. Compared to the above results, the solutions are much better: for the 532-cities problem a best tour length of 0.10% and an average tour length of 0.19% above the optimum has been obtained [6]. Mathias and Whitley [18] have shown that MPX is the most suitable crossover for the TSP, since it outperforms edge-3 and several others. Ulder et al. [27] have modified the parallel GA of Mühlenbein by replacing 2-opt with the LK heuristic, which is known to be the best improvement heuristic for the TSP. Not surprisingly, the results are better than those obtained using 2-opt. For the 532-cities instance an average tour length of

0.17% above the optimum has been found.

Homaifar et al. [10] recently developed another crossover for the TSP, called *matrix crossover*. Using 2-opt as the improvement operator, their GA is able to find tours up to 1.96% above the optimum in case of a 318-cities problem.

The probably most effective approach to the TSP is the one proposed by Johnson [12, 13]. His *Iterated Lin-Kernighan* heuristic (ILK) works as follows: a locally optimal solution produced by LK is slightly mutated by a random non-sequential 4-change [17] and LK is applied again to the modified solution. This is repeated until the results are satisfactory. Although ILK has recently been regarded by Johnson as a "genetic" approach [13], we do not refer to ILK as a GA since the key concepts of GAs, a population of individuals and recombination of (parts of) several individuals, are missing. In ILK, a single individual rather than a population of individuals is considered at each iteration step. Nevertheless, the results obtained by ILK for many TSP instances are truly impressive.

3 A New Approach

The above results indicate that the use of improvement operators within a GA seems to be the only way to obtain results comparable to conventional local search techniques. In [20], Mühlenbein justifies his genetic local search approach by a configuration space analysis of the TSP. From his observations he concludes that edges of best 2-opt tours are most likely also contained in other 2-opt tours and that by combining two good tours the probability of obtaining a still better tour is higher than by combining two arbitrary 2-opt tours. We believe that this also holds for tours produced by other improvement heuristics, such as 3-opt or LK. In support of this claim, Boese [1, 2] argues that the cost surfaces of TSPs exhibit a "big valley" structure, i.e. the optimum lies near the center of a single valley of near-optimal solutions.

A GA which is specifically designed to search within the space of local optima instead of the set of feasible solutions differs in some respects from a standard GA. In a standard GA, the crossover is aimed at preserving most of the information contained in the parents. Since in a local search GA the offspring produced by the crossover operator is subsequently modified by an improvement heuristic, this aim may be disadvantageous. If the two parents are too similar, the GA may fail to produce a new offspring, since after local improvement the offspring may become identical to one of the parents. The probability of getting an identical offspring increases for more effective local search methods like 3-opt or LK.

To estimate the similarity of two individuals, we need a metric for defining distances between tours. Most operations on tours are performed on the edges of the tour, so it is straightforward to define the distance between two tours as the number of edges in which the two tours differ. If this distance between two individuals is zero, then these two individuals are said to be identical, because all edges of the first can be found in the second. This distance is also used in [1, 2, 20]. It can be seen as a counterpart to the Hamming distance for bitstrings.

It is easy to see that the smaller the distance between two parents, the higher the probability of falling back in one of the two local minima. Therefore, the new crossover operator presented in the following produces an offspring by preserving the distance between tours rather keeping the number of newly introduced edges to a minimum. The proposed operator, called *DPX* (*distance preserving crossover*), tries to generate an offspring that has equal distance to both of its parents, i.e. 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 non-shared 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.

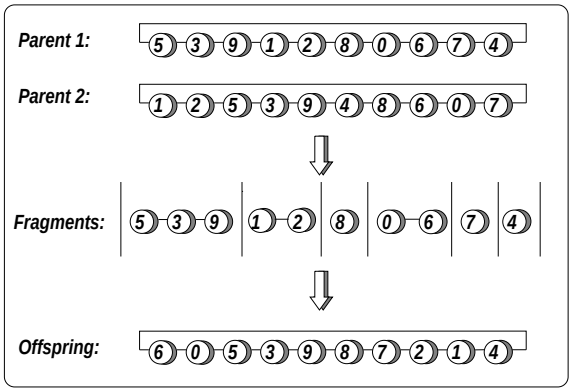

Fig. 2. The DPX crossover

Suppose that the two parents shown in Fig. 2 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. 2 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 not yet visited tour fragments. 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 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. The entire GA using the DPX crossover is shown in Fig. 3.

```
procedure TSP-GA;
    begin
        initialize population with nearest-neighbor heuristic;
        foreach individual do Lin-Kernighan-Opt(individual);
        repeat
            for  i:=0 to # of crossovers do
                select two individuals randomly;
                DPXcrossover(mom, dad);
                Lin-Kernighan-Opt(child);
                with predefined probability do mutation(child);
                replace an individual in population by mutated child;
            end;
        until converged;
    end;
```

Fig. 3. The GA

The initial population of tours is generated by a simple nearest-neighbor tour construction heuristic, applied to P different (randomly chosen) starting points. Each individual of the initial population is then locally optimized by the LK tour improvement heuristic. The recombination process starts with randomly selecting two individuals, performing the DPX crossover as described above, and applying the LK heuristic to the produced offspring to again obtain a locally optimal tour. With a predefined probability, this tour is then mutated by a random non-sequential 4-change [17], followed by a run of the LK heuristic to transform the mutated tour into a local minimum. The resulting tour finally replaces an individual of the current population before the next iteration of the algorithm is executed.

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 and it does not go too far away from a good solution; LK performs only sequential changes, i.e. the endpoint of a removed edge is the startpoint of a newly inserted edge, and the chance to eliminate the effect of the 4-change by two or more LK runs is low.

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 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, that individual is replaced by the new offspring, unless that individual is the currently best individual in which case it is only replaced when the new offspring has a higher fitness; otherwise the individual with the worst fitness in the current population will be replaced.

The motivation for the individual actions performed by our approach shown in Fig. 3 can be explained as follows. The GA investigates the search space of local optima by "jumping" from one local optimum to another, with the hope to find still better local minima until the global minimum is reached. The difficulty is to jump far enough away to leave the basin of attraction of the current local minimum, but not too far to possibly miss the optimum which is expected to be somewhere near the current local minimum (see Fig. 4).

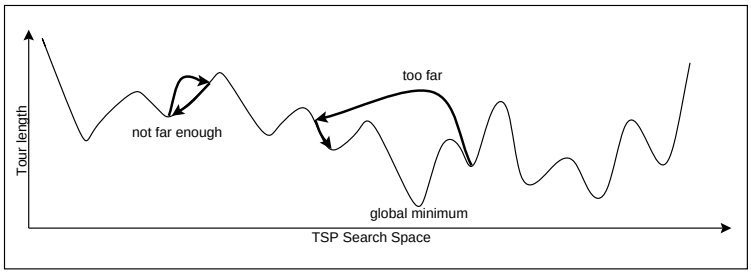

Fig. 4. Illustration of jumps within the search space of local minima

In our approach, the two entities responsible for providing the GA with the "jumping" capability are the DPX crossover and the mutation operator. Since the direction of a jump is determined by the particular edges removed/reinserted, our mutation operator performs a jump with constant distance and random direction, while the DPX performs a jump with variable distance and guided direction. The direction of a jump performed by the DPX is guided in the sense that it removes edges that most probably are not part of the optimal tour, and it selects edges for reinsertion that are not contained in the two near-optimum individuals acting as the parents.

4 Implementation and Results

The proposed approach has been implemented in C++ on a DEC-Alphastation 3000/300X (175 MHz) under OSF/1. Our implementation of the GA operators is based on Wall's GALIB library (Version 2.3.2) [28]. In Table 1 the results obtained in our experiments are shown. The TSP instances investigated were taken from the TSPLIB [23] (the numbers in the abbreviations indicate the number of cities). The shortest tours are known for all of them, thus the third and the fourth columns indicate the deviation from the optimum in percent. For each of the problem instances, 10 independent runs were performed. In the

experiments, the following parameters were used: (a) population size: 20, (b) crossover rate: 0.5 (i.e. 10 new offsprings per generation), and (c) mutation rate: 0.3 (i.e. 30% of the population per generation are mutated). The experiments were stopped after 200 (lin318: 100) generations.

Results				
instance	optimum	best(quality)	average(quality)	t(hh:mm:ss)
lin318	42029	42029 (0.00%)	42029.0 (0.00%)	00:34:14
att532	27686	27686 (0.00%)	27693.7 (0.03%)	03:16:20
rat783	8806	8806 (0.00%)	8807.3 (0.01%)	05:53:30
pcb1173	56892	56893 (0.00%)	56894.7 (0.01%)	20:05:20
fl1400	20127	20127 (0.00%)	20129.5 (0.01%)	57:53:39

Table 1. The results of the proposed GA

The results shown in Table 1 demonstrate that the proposed approach is capable of producing high quality solutions for each of the TSP instances investigated. Our results are qualitatively superior to any of the results obtained by all other GA approaches known to us and comparable to the best non-GA approaches such as ILK; for example, the average quality produced by ILK for att532 [13] is slighly worse, but its runtimes are significantly better than ours. Actually, ILK may be regarded as a special case of our approach, since it is obtained by setting the population size and the mutation rate to 1 and the crossover rate to 0.

For all the instances the optimal solutions have been found with the proposed approach, except for the instance pcb1173. In this case, the best obtained tour was only slightly longer than the optimum (56893 vs. 56892); the percentage value given in the third column of Table 1 has been rounded. The average quality shows for all instances that the best solutions were found in the majority of runs.

In addition to the information shown in Table 1, it is interesting to consider the increase of quality when executing the individual steps of the proposed GA. The length of the best tour generated by the nearest neighbor tour construction heuristic is usually around 20% above the optimum and takes up to 12 seconds (for fl1400) to compute. The subsequent use of the LK improvement heuristic yields solutions which are at best between 2% and 3% above the optimum, with computation times ranging between 1 second (for lin318) and 68 seconds (for fl1400). Thus, the GA is responsible for the smallest but hardest part of the optimization process, namely contributing the remaining quality improvements to reach the values shown in Table 1.

5 Conclusions

In this paper, an approach has been presented to incorporate domain knowledge into a genetic algorithm which is supposed to compute near-optimum solutions to the traveling salesman problem (TSP). The approach is based on the use of

a tour construction heuristic for generating the initial population of the GA, a tour improvement heuristic for finding local optima in the TSP search space, and new genetic operators for efficiently searching the space of local optima in order to find the global optimum. The feasibility of the proposed approach has been demonstrated by presenting performance results for a number of TSP instances containing between 318 and 1400 cities. The results are superior to those produced by other GA approaches described in the literature and comparable to the best non-GA approaches.

There are several issues for future research. For example, the efficiency of the implementation must be increased to reduce the computation times; in particular our implementation of the Lin-Kernighan heuristic is a target for improvement, and a parallel version of the whole approach would be desirable. The results of Johnson [13] indicate that additional implementation optimizations can reduce the runtimes drastically. Furthermore, additional tests of the proposed algorithm on other possibly more complex or asymmetric TSP instances [4] and comprehensive comparisons with other (possibly non-GA) approaches, in particular the ILK heuristic, are required to provide a detailed assessment of the merits of the proposed approach.

References

1. K. Boese, "Cost versus Distance in the Traveling Salesman Problem," Tech. Rep. TR-950018, UCLA CS Department, 1995.
2. K.D. Boese and S. Muddu, "A New Adaptive Multi-Start Technique for Combinatorial Global Optimizations," *Operations Research Letters 16*, pp. 101–113, 1994.
3. B. Freisleben and M. Schulte, "Combinatorial Optimization with Parallel Adaptive Threshold Accepting," in *Proceedings of the 1992 European Workshop on Parallel Computing*, Barcelona, pp. 176–179. IOS Press, 1992.
4. B. Freisleben and P. Merz, "A Genetic Local Search Algorithm for Solving Symmetric and Asymmetric Traveling Salesman Problems," in *Proc. 1996 IEEE Int. Conf. on Evolutionary Computation*, Nagoya, Japan, pp. 616–621, 1996.
5. D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1989.
6. M. Gorges-Schleuter, "ASPARAGOS: An Asynchronous Parallel Genetic Optimization Strategy," in *Proc. of the 3rd Int. Conf. on Genetic Algorithms*, Morgan Kaufmann, 1989.
7. M. Gorges-Schleuter, "Explicit Parallelism of Genetic Algorithms through Population Structures," in *Parallel Problem Solving from Nature*, (H. Schwefel and R. Männer, Eds.), pp. 150–159, Springer-Verlag, 1991.
8. J. J. Grefenstette, "Incorporating Problem Specific Knowledge into Genetic Algorithms," in *Genetic Algorithms and Simulated Annealing*, (L. Davis, ed.), pp. 42–60, Morgan Kaufmann, 1987.
9. J. Grefenstette, R. Gopal, B. Rosimaita, and D. van Gucht, " Genetic Algorithms for the Traveling Salesman Problem," in *Proc. of an Int. Conf. on Genetic Algorithms and their Applications*, pp. 160–168, 1985.
10. L. Homaifar, C. Guan, and G. Liepins, "A New Approach to the Traveling Salesman Problem by Genetic Algorithms," in *Proc. of the 5th Int. Conf. on Genetic Algorithms*, pp. 460–466, Morgan Kaufmann, 1993.

11. 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. of the 3rd Int. Conf. on Genetic Algorithms*, pp. 110–115, Morgan Kaufmann, 1989.
12. D. S. Johnson, "Local Optimization and the Traveling Salesman Problem," in *Annual Int. Colloquium on Automata, Languages and Programming*, 1990.
13. D. S. Johnson and L. A. McGeoch , "The Traveling Salesman Problem: A Case Study in Local Optimization," in (E. H. L. Aarts and J. K. Lenstra, eds.) *Local Search in Combinatorial Optimization*, Wiley & Sons, New York, to appear.
14. P. van Laarhoven and E. H. L. Aarts, *Simulated Annealing: Theory and Applications*. Kluwer Academic Publ., 1987.
15. E. L. Lawler, J. K. Lenstra and D. B. Shmoys, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. Wiley and Sons, New York, 1985.
16. S. Lin, "Computer Solutions of the Travelling Salesman Problem," *Bell System Tech. Journal*, Vol. 44, pp. 2245–2269, 1965.
17. S. Lin and B. Kernighan, "An Effective Heuristic Algorithm for the Traveling Salesman Problem," *Operations Research*, Vol. 21, pp. 498–516, 1973.
18. K. Mathias and D. Whitley, "Genetic Operators, the Fitness Landscape and the Traveling Salesman Problem," in *Parallel Problem Solving from Nature*, (R. Männer and B. Manderick, Eds.), pp. 219–228, Elsevier, 1992.
19. H. Mühlenbein, "Parallel Genetic Algorithms, Population Genetics and Combinatorial Optimization," in *Proc. of the 3rd Int. Conf. on Genetic Algorithms*, (J. D. Schaffer, ed.), pp. 416–421, Morgan Kaufmann, 1989.
20. H. Mühlenbein, "Evolution in Time and Space – The Parallel Genetic Algorithm," in *Foundations of Genetic Algorithms*, (G. J. E. Rawlins, ed.), M. Kaufmann, 1991.
21. H. Mühlenbein, M. Gorges-Schleuter, and O. Krämer, "Evolution Algorithms in Combinatorial Optimization," *Parallel Computing*, Vol. 7, pp. 65–88, 1988.
22. M. Padberg and G. Rinaldi, "Optimization of a 532-city Symmetric Traveling Salesman Problem by Branch & Cut," *Operations Research Lett. 6*, pp. 1–7, 1987.
23. G. Reinelt, "TSPLIB — A Traveling Salesman Problem Library," *ORSA Journal on Computing*, Vol. 3, No. 4, pp. 376–384, 1991.
24. G. Reinelt, *The Traveling Salesman: Computational Solutions for TSP Applications*. Vol. 840 of *Lecture Notes in Computer Science*, Springer-Verlag, 1994.
25. J. Y. Suh and D. van Gucht, "Incorporating Heuristic Information into Genetic Search," in *Proc. of the 2nd Int. Conf. on Genetic Algorithms*, pp. 100–107, Lawrence Erlbaum, 1987.
26. A. Y. C. Tang and K. S. Leung, "A Modified Edge Recombination Operator for the Travelling Salesman Problem," in *Parallel Problem Solving from Nature*, (H.-P. Schwefel and R. Männer, Eds.), pp. 180–188, Springer-Verlag, 1994.
27. 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*, (H. P. Schwefel and R. Männer, Eds.), pp. 109–116, Springer-Verlag, 1991.
28. M. Wall, "GALIB 2.3.2," http://lancet.mit.edu/ga, 1995.
29. D. Whitley, T. Starkweather, and D. Fuquay, "Scheduling Problems and Traveling Salesman: The Genetic Edge Recombination Operator," in *Proc. of the 3rd Int. Conf. on Genetic Algorithms*, pp. 133–140, Morgan Kaufmann, 1989.

This article was processed using the LaTeX macro package with LLNCS style