Pitfalls with Adaptive Methods for Combinatorial Optimization: The Traveling Salesman Problem - A Case Study

Wolfgang W. Bein * Bradley J. Hendricks †

Abstract

We study adaptive approaches for the symmetric Euclidean Traveling Salesman problem. Specifically, we consider 2-opt, simulated annealing, and genetic algorithms with cycle crossover, ordered crossover, and edge recombination. Our algorithms are implemented using GALib. We note that difficulties associated with the TSP problem are representative of complexities in solving combinatorial optimization problems with adaptive methods in general.

1 Introduction

We focus on the the symmetric Euclidean Traveling Salesman Problem, a variant of the Traveling Salesman Problem (TSP) where the underlying metric space is the Euclidean plane. In this problem, coordinates $(x_1, y_1), \ldots, (x_n, y_n)$ of n points (cities) $1, \ldots, n$ are given and the distance between two points is computed as

$$\mathrm{dist}(i,j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}.$$

We are interested in finding a permutation t ("the tour"), such that

$$f(t) = \mathrm{dist}(t(n), t(1)) + \sum_{i=1}^{n-1} \mathrm{dist}(t(i), t(i+1))$$

is minimized.

TSP is without doubt the combinatorial optimization which has the most notoriety; see the now classic book by Lalwer *et al.* for an excellent survey [9]. TSP (more precisely, its decision problem version) is $\mathcal{NP}$-complete even under the Euclidean restriction. More importantly, TSP is PLS-complete (PLS stands for "polynomial-time local search"), suggesting that TSP is hard under local search, see [1]. Furthermore, solutions of TSP involve permutations rather than sets, a fact, which complicates genetic encoding. Recently, Boese [3] studied the fitness landscape of the symmetric TSP, and suggested that "the set of local minima 'viewed from afar' may appear to have a single minimum point, but from 'up close' has many local minima."

In this paper we consider 2-opt, simulated annealing (see [1]), and genetic algorithms (see [6, 10]) with cycle crossover, ordered crossover, and edge recombination. The results are based on software, which we have written under GALib, an object-oriented library for genetic algorithms, which was recently developed by Matthew Wall at MIT

*Department of Computer Science, University of Nevada, Las Vegas, NV 89154. Email: bein@cs.unlv.edu. Research supported by a UNLV SITE grant.

†Department of Computer Science, University of Nevada, Las Vegas, NV 89154. Email: hendricks@lvcm.com.

Parent 1	184 637 |25	random slice 637
Parent 2	352718|64	add underlined
Child	218637|45	child is a valid tour

Figure 1: Example of Ordered Crossover

[16]. Our package "TSPView" is available under `www.cs.unlv.edu/~bein/TSPView.html`.

2 The Difficulty in Constructing Effective Crossovers

One "pitfall" with sequencing problems is that standard simple string crossovers do not result in valid tours. It is therefore not at all obvious, what a crossover should look like. However, an effective crossover must not only produce valid children. Children should resemble their parents in certain ways, and should enable the inheritance of highly fit schemata. Below we list three attempts to find such crossovers.

The Cycle Crossover. The cycle crossover, as described by Oliver *et al.* [12], is perhaps the simplest crossover that merges the genetic material from two parents and generates two valid tours. This crossover identifies a subset of positions in the tour which visits the same set of cities and then swaps. For example, consider the two tours of five cities 12534 and 31425. The first, second, and fourth location contain exactly the same set of cities $\{1, 2, 3\}$. Now swapping at these positions results in 12435 and 31524. Because the same subset of cities is replaced in both parents, the children are always permutations.

The Ordered Crossover and the One-Point Crossover. In the cycle crossover, because non-adjacent cities in the parent tours are used in the crossover, the edges in the parent's tour rarely survive the crossover. Thus the child consists of many random edges and often does not resemble its parents sufficiently. The ordered crossover, first used by by Prins (see [14]), preserves more of the general shape of the parental tours.

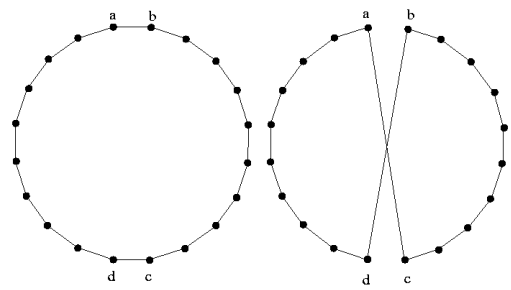

Figure 2: 2-Exchange Move

We take a random subsequence of the first parent's tour and insert it directly into the child. As described in Figure 1, the child is then completed by taking material from the second second parent's tour, where cities are inserted into the child in the order they occur in that parent, starting after the second cut location, and ignoring cities already inserted from the first parent.

The one-point crossover from Prins ([14]) is similar to the ordered crossover. The difference is that the random slice taken from the first parent always begins at the left of the array used to store the tour.

The Edge Recombination Crossover. The edge recombination crossover (see Matthew Wall [16]) is another simple crossover which uses a graph to keep similarities preserved during crossover. It first combines the edges which form

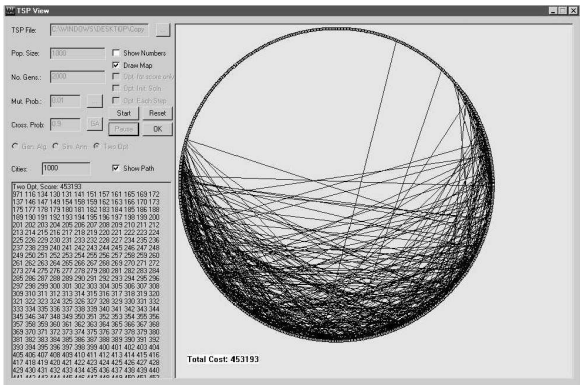

Figure 3: 2-Opt on a 1000 Point Circle

the tours of the two parents into a single graph G. This graph is then used to create a child, as follows: The first city in the child tour is selected at random. For the general step, assuming we have reached city v, the next city is chosen to be any city that can be reached by an edge in G; if there is more than one choice, a city with smallest degree is chosen. If no city is adjacent v, a random location not yet inserted into the child is chosen. After the successor city of v is determined in this manner and after it is inserted into the child tour, all edges adjacent to v are deleted from the graph.

3 Local Search and Simulated Annealing

There are straightforward local improvement algorithms for the TSP, and such schemes have been studied extensively, see Johnson and McGeoch [7] for a survey of local optimization. The idea is to define for any tour t, a neighborhood $N(t)$. Certainly the simplest neighborhood for TSP is the 2-exchange neighborhood. A neighbor is obtained by deleting two edges from the tour, and reconnecting in the obvious other way, see Figure 2. The 2-opt algorithm repeatedly transitions to a better neighbor, until no better neighbor can be found and a local minimum is reached.

1) Generate a random starting solution S;
   Set initial temperature c;
2) Set $S' =$
   a random two-exchange neighbor of S;
3) $\Delta = Length(S') - Length(S)$;
4) If $\Delta \leq 0$, set $S = S'$;
   Else with probability $e^{-\Delta/c}$, set $S = S'$;
5) If equilibrium reached, $n(n-1)$ iterations completed, reduce c,
   Else goto 2;
6) If $c > 0$ goto 2;
7) Return S as final solution;

Figure 4: The Simulated Annealing Algorithm

Figure 3 visualizes such a search. We used TSPView on a 1000 city instance, in which the cities are arranged equidistantly along a circle. In this case, of course, there is only one minimum. The figure shows a situation midway through the run. It is interesting to note that Boese's work [3] suggests that multi-start approaches involving simple neighborhoods give surprisingly good results in general.

Simulated annealing, as proposed by Kirkpatrick *et al.* [8] and Černý [5], uses an optimization method which will occasionally accept worse neighbors. The details of the algorithm are

shown in Figure 4. Boese's work [3] shows that the fitness landscape of the symmetric TSP exhibit a "globally convex" structure, which makes favorable behavior of simulated annealing plausible. This is born out by our results, see Section 5.

4 TSPView

We have developed "TSPView", which implements 2-opt, simulated annealing, and genetic algorithms with cycle crossover, ordered crossover, and edge recombination. We have written TSPView under GALib, an object-oriented library for genetic algorithms, which was recently developed by Matthew Wall at MIT [16]. The implementations are under Microsoft Visual C++. TSPView reads files in the format found under the now standard library of problems TSPLIB (see [15]). A user interface allows the users to modify many parameters in the algorithms including the choice of crossover. Figures 3 and 6 show part of this interface.

For the genetic algorithm, a 2-opt optimizer can be switched on at the beginning for the initial population set, or throughout the entire evolution. Mutations include rotations, inversions, and swaps. The rotations and inversions do not change the content of the tour; they only change the order in which they are stored in the genome. Rotation "rotates" the array by shifting the contents a random position to the right. Elements pushed off the array on the right are moved to the front of the array. An inversion inverts the order in which elements are stored in the array. The swap mutator simply swaps two elements in the array.

There is also a choice for the type of genetic algorithm: The `GASimpleGA` is the simple algorithm as described by Goldberg (see [6]). Every generation the current population is completely replaced by the children generated by the crossover and mutation operators. The `GASteadyStateGA` uses an overlapping population model. It merges a new population of individuals with the previous population and removes the worst individuals to return to the original number of individuals.

5 Results

We know from results of previous work by Brady [4] and Mühlenbein *et al.* [11] that the genetic algorithm is not the best method for solving TSP. Our experiments using the crossovers mentioned in Section 2 were indeed not very promising. We were only able to obtain near optimal tours for instances of up to roughly 200 nodes and only for certain instances of TSPLIB. Results improved when we switched on the 2-opt optimizer. Johnsen and McGeoch [7] state that using such an optimizer "could be viewed as an almost heretical addition: In the context of the biological motivation for the genetic approach, it embodies the discredited Lamarckian principle that learned traits can be inherited"[1]. What we observed was that genetic algorithms might "hypothetically" have converged to a global optimum [10], but given the sheer number and position of local optima this really was a hypothetical statement.

Figure 5 shows a result for `PR152` from TSPLIB with a value of 74846. The optimal value is 73682. This result was obtained after 7500 iterations with a population size of 2000. The `GASteadyStateGA` algorithm with ordered crossover was used, with mutations described as

[1] direct quote

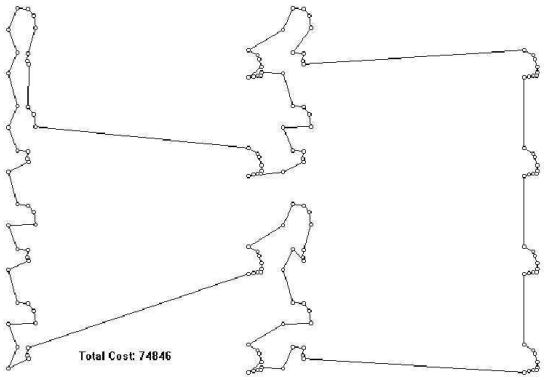

Figure 5: A Close to Optimal Solution for `PR152` Using Genetic Algorithms

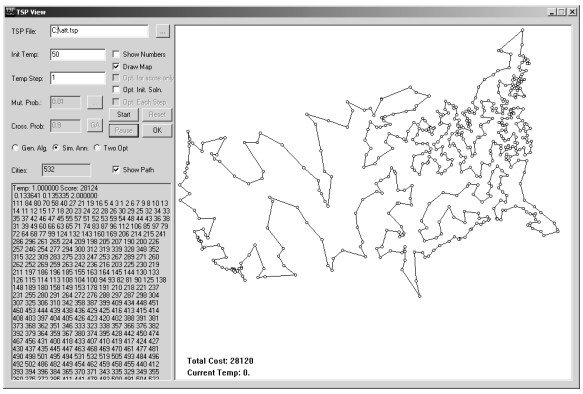

Figure 6: A Close to Optimal Solution for `ATT532` Using Simulated Annealing

above.

As mentioned in Section 3 simulated annealing seems to be better suited to TSP. The globally convex fitness landscape observed by Boese [3] lends itself quite well to a method that slowly wanders from the vicinity of one local minimum to another. Our results bear this out. It was quite easy to obtain near optimal solutions for the Padberg - Rinaldi tour `ATT532` [13]. Figure 6 shows such a tour of 532 United States cities. (The optimal value is 27686 and our tour has value 28120.)

We mention that there are many problems, which are less combinatorial in nature, where the genetic approach is effective. One such area which have we studied earlier [2] is ANN training and construction, where adaptive method seem to be a natural fit.

References

[1] Emile Aarts and Jan Karel Lenstra. *Local Search in Combinatorial Optimization.* John Wiley & Sons Ltd, West Sussex, England, 1997.

[2] W. W. Bein and B. J. Hendricks. Artificial neural network architectures via genetic algorithms using nested graph grammars. In *Proceedings of the International Conference on Mathematics and Engineering Techniques in Medicine and Biological Sciences 2000*, pages 575–578. CSREA Press, 2000.

[3] K. D. Boese. *Models for Iterative Global Optimization.* PhD thesis, UCLA, Los Angeles, California, 1996.

[4] R. M. Brady. Optimization strategies gleaned from biological evolution. *Nature*, 317:804–806, 1985.

[5] V. Černý. Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45:45–51, 1985.

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

[7] David S. Johnson and Lyle A. McGeoch. The traveling salesman problem: A case study. In Emile Aarts and Jan Karel Lenstra, editors, *Local Search in Combinatorial Optimization.* John Wiley & Sons Ltd, West Sussex, England, 1996.

[8] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[9] E. L. Lawler, Lenstra J. K., Rinnooy Kan A. H. G, and D. B. Shmoys. *The Traveling Salesman Problem: A guided Tour of Combinatorial Optimization.* John Wiley & Sons Ltd, 1985.

[10] Melanie Mitchell. *An Introduction to Genetic Algorithms.* The MIT Press, Cambridge, Massachusetts, 1996.

[11] H. Mühlenbein, M. Gorges-Schleuter, and O. Krämer. Evolution algorithms in combinatorial optimization. *Parallel Computing*, 7:65–85, 1988.

[12] I. M. Oliver, D. J. Smith, and J. R. C. Holland. A study of permutation crossover operators on the traveling salesman problem. In J. J. Grefenstette, editor, *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, Hillsdale, New Jersey, 1987. Lawrence Erlbaum.

[13] W. Padberg and G. Rinaldi. Optimization of a 532 city symmetric TSP. *Optimization Research Letters*, 6(1):1–7, 1987.

[14] Christian Prins. Competitive genetic algorithms for the open-shop scheduling problem. Technical report, Ecole des Mines de Nantes, 1999.

[15] Gerhard Reinelt. TSPLIB95. Technical report, Universität Heidelberg Institut für Angewandte Mathematik, 1995.

[16] Matthew Wall. *GAlib: A C++ Library of Genetic Algorithm Components.* Cambridge, Massachusetts, version 2.4 edition, 1996.