The Effects of Population Size, Heuristic Crossover and Local Improvement on a Genetic Algorithm for the Traveling Salesman Problem

Prasanna Jog Jung Y. Suh Dirk Van Gucht

Computer Science Department, Indiana University
Bloomington, IN 47405-4101, USA

1 Introduction

The *Euclidean Traveling Salesman Problem* (TSP) is the problem of: given N cities, if a salesman starting from his home city is to visit each city exactly once and then return home, find the order of visits (the tour) such that the total distance traveled is minimum. The distance between two cities is just the Euclidean distance between them.

In recent years a variety of GAs for the TSP have been proposed, for example [2,5,4,10,13]. In this paper we will focus on the class of such GAs which incorporate heuristics about the problem into the recombination operators, i.e., the crossover and mutation operators [5,4,13]. We will call such algorithms *heuristic genetic algorithms* (HGA).

There are a number of issues that need to be addressed in determining whether the properties of GAs are reflected in HGAs. The reason for this results from the possibility that HGAs perform well only because of the effectiveness of the incorporated heuristics and not because of the properties typically attributed to GAs. In this paper we study the importance of the population size, the crossover operator and the type of mutation operators used. We empirically determine the following results:

1. Population size matters, i.e., tour lengths obtained at the end of the execution of a HGA decrease inversely with growing population size.
2. Crossover matters, i.e., there is a qualitative difference between a HGA that uses a crossover operator and a HGA that does not.
3. The incorporation of more sophisticated mutation operators (from now on called *local improvement operators*) improves the quality of the solutions.

2 Heuristic Genetic Algorithms

We will call a genetic algorithm a *heuristic genetic algorithm* (HGA) if problem specific heuristics are incorporated in the recombination operators. We considered the following recombination operators:

The heuristic crossover This operator is a modification of the crossover described in [5,4,13]. This operator constructs an offspring from two parent tours as follows: Pick a random city as the starting point for the offspring's tour. Compare the two edges leaving the starting city in the parents and choose the shorter edge. Continue to extend the partial tour by choosing the shorter of the two edges in the parents which extend the tour. If the shorter parental edge would introduce a cycle into the partial tour, check if the other parental edge introduces a cycle. In case the second edge does not introduce a cycle, extend the tour with this edge, otherwise, extend the tour with an edge obtained by selecting the shortest edge from a pool of random edges which do not introduce a cycle (in our case the size of the pool was 20). Continue until a complete tour is generated and then replace one of the parent tour with this offspring tour. The heuristic crossover operator has the tendency to combine "good" subpaths from both parents. Furthermore, we have implemented the heuristic crossover in such a manner that it is performed more often in the beginning than towards the end of a run of a genetic algorithm. This is determined by checking how different the tours are before they undergo crossover. We decided to allow crossover if 30% of the edges in the parents were different.

The 2-opt local improvement operator The 2-opt operator is an example of the more general r-opt operator introduced by Lin and Kernighan [11]]. An r-opt operation consists of replacing r edges from a tour by r edges not in the tour if this decreases the length of the tour. For clarity, the 2-opt strategy randomly selects two edges (e, f) and (g, h) from a tour and checks if

$$ED(e, f) + ED(g, h) > ED(e, h) + ED(f, g)$$

(ED stands for Euclidean distance). If this is the case, the new tour is obtained by removing the edges (e, f) and (g, h) and replacing them with the edges (e, h) and (g, f)

The Or-opt local improvement operator The Or-opt was introduced by Or [9,12] and is a variant of the 3-opt operator of [11]. The advantage of the Or-opt operator is the fact that it only considers a small percentage of the exchanges that would be considered by a regular 3-opt operator. To understand how the Or-opt procedure works, we refer to Figure 2. For each connected string of s cities in the current tour (s equals 3 first, then 2, then 1), we test to see if the string can be relocated between two other cities at reduced cost. If it can, we make the

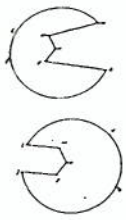

Figure 1: Illustration of a Or-opt local improvement operation.

P(t) denotes the population at time t.

```
t ← 0;
initialize P(t);
evaluate P(t);
while ( not termination condition)
{ t ← t+1;
   select P(t);
   perform x% heuristic crossover P(t);
   perform y% 2-opt local improvement P(t);
   perform z% Or-opt local improvement P(t);
   evaluate P(t); }
```

Figure 2: Layout of a heuristic genetic algorithm

appropriate changes. For $s = 3$ in Figure 1, we test to see if the string of three adjacent cities m, n, p in the current tour is considered for insertion between a pair of connected cities i and j outside of the string. The insertion is performed if the total cost of the edges to be erased, $\{a, m\}, \{p, b\}$, and $\{i, j\}$, exceeds the cost of the new edges to be added, $\{i, m\}, \{p, j\}$ and $\{a, b\}$. After considering all strings of three cities, all strings of two cities and then all strings of one city are considered.

Hence the layout of a HGA is as displayed in Figure 2. The percentages x, y and z indicate the percentage of the population that will undergo heuristic crossover, 2-opt local improvement and Or-opt local improvement respectively during a *generation* (i.e., one iteration of the while loop). We should also mention that after an offspring tour (due to a crossover) or an improved tour (due to local improvement operations) has been incorporated into the new population, the offspring tour or the improved tour has to be evaluated. We will call such an evaluation a *trial*. The number of trials will be used to measure the convergence rate of HGAs (see Section 3).

3 Experimental Setup and Results

In this section we describe the results obtained for four HGAs applied to the krolak and lattice TSPs. The empirical results were obtained by applying these HGAs to two TSP problems reported in the literature. Problem 1 is the *krolak* TSP [8], a 100 city problem of uniformly distributed cities (see Figure 3). The optimum solution for this problem is equal to 21282. Problem 2 is the 100 city lattice TSP [1] (see Figure 3). The optimum solution for this problem is equal to 100. In Section 4 we discuss our results. We will call

Figure 3: The krolak TSP

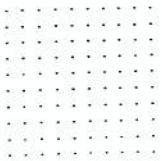

Figure 4: The lattice TSP

2-opt GA, the HGA which only uses the 2-opt operator. Hence we set the percentages x and z to 0. The percentage y was set to 50 (hence half of the population undergoes 2-opt local improvement operations). We should also mention what we mean for a tour to undergo local improvement. If a tour is selected for local improvement, it will undergo a certain number of local improvement attempts during the same generation. In all of our experiments this number of attempts was set to 10. It is also important to mention that the side effect of the selection procedure is to randomly shuffle the tours in the current population. This is to guarantee that all tours have an equal probability of being chosen to undergo local improvements during the course of the algorithm.

2-opt–cross GA, the HGA which uses the 2-opt operator in combination with the heuristic crossover. Hence we set z to 0. The parameters x and y where each set to 50 (hence half of the population undergoes heuristic crossover, the other half undergoes local improvement).

2-opt–Or-opt GA, the HGA which uses the 2-opt operator and the Or-opt operator. Hence x was set to 0. The parameters y and z where set to 25 (hence half of the population underwent local improvement operations).

2-opt–Or-opt–cross GA, the HGA which uses the 2-opt operator, the Or-opt operator and the heuristic crossover operator. The parameters x, y and z were set to 50, 25 and 25 respectively.

Population	Trials	Average	Best	Worst
10	7000	9.55	3.78	14.73
25	14000	8.14	4.48	13.00
50	19000	6.78	4.55	10.23
100	51000	7.10	2.77	11.33

Table 1: Results of the 2-opt GA on the krolak TSP.

Population	Trials	Average	Best	Worst
10	6000	7.3	4.14	8.69
25	7000	6.4	4.14	9.11
50	13000	7.1	4.14	9.52
100	28000	5.5	3.31	8.28

Table 2: Results of the 2-opt GA on the lattice TSP.

3.1 2-opt GA

In Table 1 and Table 2 we show the results of applying the 2-opt GA to the krolak and lattice problem for varying population sizes. The first column in these tables denotes the various population sizes, the second and third column indicate the average number of trials and the average percentage away from the optimum respectively (averages were taken over 10 experiments), the fourth (fifth) column gives the percentage away from optimum of the best (worst) tour found during the ten experiments. For example the first row in Table 1 (10 1400 9.55 3.78 14.73) indicates that for population size 10, after an average of 1400 trials, the average of the best tours found with the 2-opt GA was 9.55% away from the optimum, and 3.78 (14.73) is the percentage away from optimum for the best (worst) tour found with this HGA during the ten experiments. In Figure 5, we show the convergence rates for the 2-opt GA applied to the krolak problem [1] (optimum is 21282) with population sizes 10, 25, 50 and 100 respectively. It appears from Figure 5 that the tours obtained are better with growing population sizes (although in our case the 2-opt GA with population size slightly outperforms the 2-opt GA with population size 100). On the other hand, the final tours in all these 2-opt GAs are on average never better than 6.70% above the optimum. Finally we should mention (as is normal) that the convergence rate for 2-opt GAs with smaller population sizes is faster than that for 2-opt GAs with larger population sizes.

3.2 2-opt–cross GA

In Table 3 and Table 4 we show the results of applying the 2-opt-cross GA to the krolak and lattice problem for varying population sizes. When comparing these tables with Table 1 and Table 2 respectively, we notice that incorporating the heuristic crossover makes a distinct difference. For example for the krolak problem, the 2-opt–cross GA with population size 100 yields on average a tour 2.59% away from optimum, as opposed with the corresponding 2-opt GA which yields on average a tour 7.1% away from optimum. In Figure 6, we show the convergence rates for the 2-opt–cross GAs applied to the krolak problem (optimum is 21282) with population sizes 10, 25, 50 and 100 respectively. It is clear from Figure 6 that the tours obtained are better with growing population sizes. Again notice that the convergence rate for 2-opt–cross GA with smaller population sizes is faster than that for 2-opt–cross GA with larger population sizes.

Population	Trials	Average	Best	Worst
10	7000	7.53	0.07	13.47
25	14000	5.41	2.71	9.90
50	37000	3.20	0.46	5.22
100	50000	2.58	0.78	6.82

Table 3: Results of the 2-opt–cross GA on the krolak TSP.

Population	Trials	Average	Best	Worst
10	4000	4.7	2.48	6.20
25	8000	2.9	1.65	4.97
50	28000	1.7	0.0	3.72
100	33000	1.4	0.0	1.16

Table 4: Results of the 2-opt–cross GA on the lattice TSP.

3.3 2-opt-Or-opt GA

In Table 5 and Table 6 we show the results of applying the 2-opt-Or-opt GA to the krolak and lattice problem for varying population sizes. When comparing these tables with Table 1 and Table 2 respectively, we notice that incorporating the Or-opt local improvement operator makes a distinct difference. For example for the krolak problem, the 2-opt-Or-opt GA with population size 100 yields on average a tour 2.59% away from optimum, as opposed with the corresponding 2-opt GA which yields on average a tour 7.1% away from optimum. In Figure 7, we show the convergence rates for the 2-opt–cross GAs applied to the krolak problem (optimum is 21282) with population sizes 10, 25, 50 and 100 respectively. What is particularly noticable is that the best tours obtained by 2-opt-Oropt GAs with differerent population sizes are remarkably close. For example the 2-opt-Or-opt GA with population size 10 yields best tours 3.11% away from optimum, whereas the 2-opt-Or-opt GA with population size 100 yields best tours 2.59% away from optimum. So it appears that for these HGAs, population size does not play a critical role.

[1] A similar figure can be obtained for the lattice problem but is omitted from this paper.

Population	Trials	Average	Best	Worst
10	3000	3.11	0.46	5.92
25	14000	3.77	0.46	8.22
50	37000	3.25	0.11	5.97
100	50000	2.69	0.11	8.26

Table 5: Results of the 2-opt-Or-opt GA on the krolak TSP.

Population	Trials	Average	Best	Worst
10	17000	3.2	1.65	4.14
25	36000	2.6	0.82	4.14
50	31000	3.0	1.16	4.14
100	46000	2.7	1.65	3.31

Table 6: Results of the 2-opt-Or-opt GA on the lattice TSP.

3.4 2-opt-Or-opt–cross GA

In Table 7 and Table 8 we show the results of applying the 2-opt-Or-opt–cross GA to the krolak and lattice problem for varying population sizes. When comparing these tables with Table 5 and Table 6 respectively, we notice again (as in the case of the 2-opt GA and the 2-opt–cross GA) that incorporating heuristic crossover makes a distinct difference. For example for the krolak problem, the 2-opt-Or-opt–cross GA with population size 100 yields on average a tour 1.37% away from optimum, as opposed to the corresponding 2-opt–Or-opt GA which yields on average a tour 2.59% away from optimum. Furthermore when we compare Table 7 and Table 8 with Table 3 and Table 4 respectively that, we find incorporating the Or-opt operator makes a distinct difference. For example for the krolak problem, the 2-opt-Or-opt–cross GA with population size 100 yields on average a tour 1.37% away from optimum, wheras the corresponding 2-opt–cross GA yields on average a tour 2.59% away from optimum. In Figure 8, we show the convergence rates for the 2-opt-Or-opt–cross GAs applied to the krolak problem (optimum is 21282) with population sizes 10, 25, 50 and 100 respectively. It is clear from Figure 8 that the tours obtained are better with growing population sizes. Again notice that the convergence rate for 2-opt-Or-opt–cross GA with smaller population sizes is faster than that for 2-opt-Or-opt–cross GA with larger population sizes.

Population	Trials	Average	Best	Worst
10	18000	3.06	0.34	8.28
25	26000	2.31	0.80	4.95
50	38000	1.70	0.56	3.03
100	59000	1.37	0.01	3.62

Table 7: Results of the 2-opt-Or-opt–cross GA on the krolak TSP.

Population	Trials	Average	Best	Worst
10	43000	2.1	0.82	2.48
25	35000	2.0	0.82	3.31
50	32000	1.7	0.82	3.31
100	42000	0.4	0.0	0.82

Table 8: Results of the 2-opt-Or-opt–cross GA on the lattice TSP

4 Discussion

The previous section enables us to make the following observations:

1. Population size matters: for all HGAs, except perhaps the 2-opt-Or-opt–cross GA, increasing the population size improves the tour length of the final tour obtained by the various HGAs. Interestingly, it appears that the popu;lation sizew seems to play a more vital role for HGAs with crossover.
2. The heuristic crossover substantially improves the performance of HGAs.
3. Incorporating more sophisticated heuristics (such as the Or-opt operator) improves the overall performance of HGAs.
4. The convergence rate of HGAs for smaller population sizes is faster than that for HGAs with larger population sizes.
5. The is a qualitative difference between HGAs with and without crossover, i.e., the convergence rate for HGAs with crossover is faster than that for HGAs without crossover. This faster convergence rate haowever, does not lead to premature convergence.

For comparison reasons, we show in Figure 9 (Figure 10) the performance of the four HGAs (with population size 100) applied to the krolak (lattice) problem . It appears from these figures that in order of weakness, the 2-opt GA comes first, the 2-opt-Or-opt GA comes second, the 2-opt-cross GA comes third and the 2-opt-Or-opt–cross GA comes fourth.

To put our results in perspective, we refer to [3]. They reported on the performance of various heuristic algorithms on the Krolak problem. Our best results compare favorably with the best results reported there. For the lattice problem, we refer to [1]. There the best solution was 3% above optimum. Our algorithms yield better results.

We would like to conclude this discussion by giving some insights about the role of heuristic crossover in HGAs. As was already discussed in [5,4], the heuristic crossover has the tendency to quickly "glue" together

good subpaths from parent tours. This glueing process takes place mostly in the beginning of the algorithm (this is why the convergence rate of HGAs with crossover have a faster convergence rate than HGAs without the crossover). Furthermore, the heuristic crossover has the interesting property (much like simulated annealing [7,1]) to be able to get out of local optima, by allowing crossover to temporarily introduce worse offsprings [6]. We believe that this observation is one of the central reasons for the excellent performance of HGAs with the crossover operator.

Acknowledgements

We would like to thank John Grefenstette and John Holland for interesting conversations about genetic algorithms and their application to the traveling salesman problem.

References

[1] V. Cerny. Thermodynamical approach to the traveling salesman problem. *Journal of Optimization Theory and Application*, 45(1):41–52, January 1985.

[2] Goldberg D.E. and Lingle R. Alleles, loci, and the traveling salesman problem. In Grefenstette J.J., editor, *Proceedings of an International Conference on Genetic Algorithms and their Applications*, pages 154–159, July 1985.

[3] B.L. Golden, Bodin L.D., T. Doyle, and W. Stewart. Approximate traveling salesman algorithms. *Operations Research*, 28:694–711, 1980.

[4] J.J. Grefenstette. Incorporating probelm specific knowledge into genetic algorithms. In L. Davis, editor, *Genetic Algorithms and Simulated Annealing*. 1987.

[5] J.J. Grefenstette, R. Gopal, B.J. Rosmaita, and D. Van Gucht. Genetic algorithms for the traveling salesman problem. In Grefenstette J.J., editor, *Proceedings of an International Conference on Genetic Algorithms*, pages 160–168, July 1985.

[6] P Jog and D Van Gucht. Properties of the heuristic crossover for the traveling salesman problem. Unpublished Manuscript, 1989.

[7] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, May 1983.

[8] P.D. Krolak, W. Felts, and G. Marble. A man-machine approach toward solving the traveling salesman problem. *Communications of the ACM*, 14:327–334, 1971.

[9] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys. *The Traveling Salesman Problem*. Wiley Interscience, 1985.

[10] G.E. Liepins and Hilliard. Greedy genetics. In J.J. Grefenstette, editor, *Proceedings of the Second International Conference on Genetic Algorithms and their applications*, pages 90–99, July 1987.

[11] S. Lin and B.W. Kernighan. An efficient heuristic algorithm for the traveling salesman problem. *Operations Research*, 21:498–516, 1973.

[12] I Or. *Traveling Salesman-Type Combinatorial Problems and their relation to the Logistics of Regional Blood Banking*. PhD thesis, Northwestern University, 1976.

[13] J.Y. Suh and D. Van Gucht. Incorporating heuristic information into genetic search. In Grefenstette J.J., editor, *Proceedings of the 2nd International Conference on Genetic Algorithms and their Applications*, July 1987.