# CODING TSP TOURS AS PERMUTATIONS VIA AN INSERTION HEURISTIC

**Bryant A. Julstrom**
Department of Computer Science
St. Cloud State University
St. Cloud, MN 56301
julstrom@eeyore.stcloudstate.edu

**Keywords:** Traveling salesman problem, permutation coding, insertion heuristics.

## ABSTRACT

An insertion heuristic for the traveling salesman problem builds a tour one city at a time by inserting each city into the tour at a position that increases the tour's length the least. Permutations can encode TSP tours for genetic search via this heuristic. Decoding such a chromosome consists of inserting the cities into a tour in the order the chromosome lists them. The chromosome specifies when each city is inserted into the tour rather than where; this scheme enlists the heuristic's power in the genetic search.

This paper describes a genetic algorithm for TSP that encodes candidate tours as permutations via the insertion heuristic, as well as two crossover operators appropriate to the coding. Tests on thirteen TSP instances indicate that the algorithm is effective on instances of moderate size, often identifying optimal tours, and that it continues to find tours that are short, though not optimal, as the number of cities increases and the search space grows.

## 1. INTRODUCTION

In the familiar traveling salesman problem (TSP), we are given a collection of cities and the distances between them, and we seek a closed tour that visits each city exactly once in the shortest total distance. Formally, we seek a Hamiltonian tour of minimum length in the complete undirected weighted graph whose vertices are the cities and whose weights are the inter-city distances. TSP has been thoroughly studied and has long been known to be NP-hard [2]. It remains an archetypal combinatorial problem and the standard exemplar of a large and important class of sequencing and scheduling problems.


SAC '99, San Antonio, Texas
©1998 ACM 1-58113-086-4/99/0001 $5.00

Many researchers have described heuristic algorithms that seek good, though not necessarily optimal, solutions to TSP instances [6, 9]. One class of such heuristics applies the idea of *insertion*: into a (closed) tour on some of the cities, repeatedly insert one more city, always at the position that increases the tour's length the least.

TSP has also become a standard exercise for practitioners of genetic algorithms, who have described GAs for TSP that encode candidate tours in a variety of ways [8]. These have included ordinal representations, adjacency representations, sequences of pairs of symbols, matrices, strings of integer-valued weights, and others. Most commonly, GAs for TSP represent candidate tours directly, as permutations that list the cities in the order in which the represented tour visits them.

This paper presents a genetic algorithm for TSP in which each chromosome is again a permutation of symbols that represent the target instance's cities, but the tour a chromosome represents is identified by applying the insertion heuristic to the cities in their listed order. We can call this the *insertion-mediated permutation coding.*

This coding is not original here. Davis [1] described a GA for the graph coloring problem that encoded colorings as permutations of the graph's vertices. A greedy heuristic identified the coloring such a chromosome represented by coloring the vertices in their listed order.

In such a coding, the position of each symbol is as important as the order in which the symbols occur. In particular, when permutations represent TSP tours via the insertion heuristic, effective genetic operators will not be those used when permutations represent tours directly. The paper describes two crossover operators that tend to preserve the positions of cities from parent chromosomes into their offspring.

The coding and operators are implemented in an otherwise standard steady-state GA, and the GA is tested on thirteen TSP instances whose shortest tour lengths are known. The two crossovers yield approximately equal performance; the resulting GA performs well on TSP instances of moderate size, though its performance deteriorates as the number of cities grows; and the insertion heuristic contributes strongly to the algorithm's success.

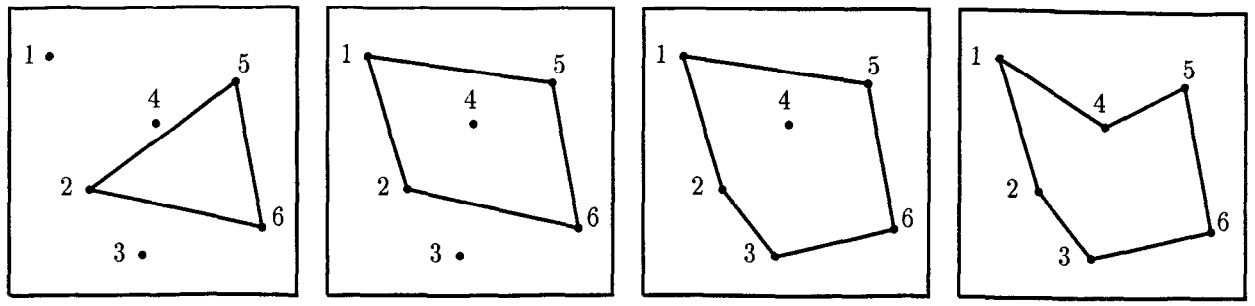


Figure 1: The development of the tour on six cities represented by the chromosome $< 6, 2, 5, 1, 3, 4 >$. The first three cities form the initial tour (a). Each subsequent city is inserted where it increases the tour's length the least (b and c), resulting in a tour on all the cities (d).

The following sections of the paper describe the insertion-mediated permutation coding, the two crossover operators, a GA that uses the coding and either of the crossovers, tests of the two versions of the GA on a range of test TSP instances, and comparisons of the GA's performance with that of two non-genetic insertion-based heuristics for the problem: a random search that applies the insertion decoding to random permutations and a hill climber that begins with a random permutation and repeatedly keeps the better of its current permutation and a variation of it.

## THE CODING

In the insertion-mediated permutation coding of TSP tours, a chromosome is a permutation of integers that represent the target TSP instance's cities. Many GAs for TSP have used permutations to represent candidate tours directly; the tour a chromosome represents visits the cities in their listed order. Here, however, the insertion heuristic determines the tour a permutation represents by *inserting* the cities in the order listed; the permutation determines not *where* each city joins the tour but *when*.

More precisely, the first three cities listed in a chromosome form an initial tour. (Their order is irrelevant.) The remaining cities are then added to the tour in their order in the chromosome and according to the insertion heuristic: each joins the tour where it increases the tour's length the least.

Consider, for example, the tour on the six cities of Figure 1 represented by the chromosome $< 6, 2, 5, 1, 3, 4 >$. The initial tour, shown in Figure 1(a), joins cities 6, 2, and 5. The chromosome lists city 1 next, and inserting it between cities 2 and 5 increases the tour's length as little as possible (Figure 1(b)). Similarly and in order, cities 3 and then 4 are inserted, resulting in the tour in Figure 1(d).

Several permutations may represent the same tour. In the example, cities 1 and 4 can be inserted in either order and city 3 can be inserted at any time without changing the resulting tour.

## 3. OPERATORS

When permutations represent tours directly, they encode adjacency information—what cities precede and follow each city—and we apply operators like Edge Recombination [10] that tend to preserve parental adjacencies in offspring permutations. Here, however, a city's position in a chromosome indicates when the insertion heuristic inserts it into the tour the chromosome represents, so we seek operators that tend to preserve the *positions* of symbols from parents to offspring.

One of these is Partially Mapped Crossover (PMX) [3], which generates one offspring permutation from two parent permutations. PMX randomly chooses a segment within one parent, then swaps symbols within that permutation until the segment is identical to the corresponding segment in the second parent.

Another plausible crossover operator applies directly the requirement that symbols be preserved in their parental positions. This operator builds a single offspring permutation in three passes over the parent permutations and the offspring, one position at a time. The first pass copies symbols that match in the parents into the offspring. The second pass copies into each unfilled position in the offspring a symbol from the corresponding position in either parent, if at least one of the parental symbols has not already been written into the offspring. The last pass completes the offspring permutation by writing into any still unfilled positions symbols chosen at random from those that the offspring does not already include. Call this operator Permutation Position Crossover (PPX); like PMX it always generates a permutation as the offspring of two parent permutations. Code that implements PPX can combine its first two passes.

An obvious mutation operator simply selects two positions in a chromosome at random and swaps the symbols in those positions.

## 4. A GA FOR TSP

The operators just described were implemented in an otherwise conventional genetic algorithm for TSP that

uses the insertion-mediated permutation coding that Section 2 described. The GA is steady-state rather than generational. To encourage variety in its population and to discourage premature convergence, it does not allow duplicate chromosomes. It initializes the population with random permutations of integers that represent the target TSP instance's cities. It selects chromosomes both to be parents and to be deleted according to linearly normalized (rank-based) probabilities; for deletion the chromosomes that represent longer tours have higher probabilities. It applies either Partially Mapped Crossover or Permutation Position Crossover and the swapping mutation, and each offspring is produced by crossover or mutation, never both. It is 1-elitist; the current best chromosome is immune from deletion until a better chromosome, representing a shorter tour, demotes it. The algorithm runs through the generation of a fixed number of new chromosomes.

## 5. TESTS

The GA was tested on thirteen instances of the traveling salesman problem, ranging in size from 70 to 318 cities. All are found in Reinelt's data base of such problems [9], and the lengths of shortest tours on them are known. Initial tests indicated that a population size equal to the number of cities in the target TSP instance provided good performance, as did probabilities of crossover and mutation equal to 30% and 70%, respectively, and the GA is robust with respect to these parameters. These values were used in all subsequent tests, and the base values of the linear normalizations for both selection and deletion were set to 1. The GA ran through the generation of $500n$ new chromosomes, where $n$ is the number of cities in the target TSP instance. The GA was run 25 times on each test instance using first PMX and then PPX as its crossover operator. The two left columns of Table 1 summarize the results of these trials.

In general, both versions of the GA performed well, particularly on the smaller instances. On **kroA100** and **pr107**, for example, at least one version of the algorithm identified an optimal tour on every trial. On almost all the instances, the average length of the shortest tours found was within 1% of the length of an optimal tour.

With both PMX and PPX, the algorithm's performance deteriorated as the number of cities increased, at least in terms of the number of trials that identified optimal tours. With neither crossover did the GA find any optimal tours in the trials on instances of 200 or more cities. However, the average lengths of the shortest tours found remained close to the optimal tour lengths. These results compare favorably with those of other non-parallel GAs that do not apply local search operators.

The tests revealed no consistent difference in performance between the two crossover operators. On a few instances, the GA showed visibly better results with one or the other—PPX on **pr107** and **pr136**, PMX on **rat99** and **kroA150**—but in general it is impossible to claim that either is more effective in this algorithm, and the low probability with which the GA applies crossover will tend to mute such differences in any case.

## 6. COMPARISONS

An insertion heuristic alone can often identify a good TSP tour. This suggests that the heuristic that decodes chromosomes is doing most of the heavy lifting and that the GA's contribution to the search for short tours is minimal. The first claim is true, but the GA does contribute to identifying short tours, as we can see by comparing the GA with two non-genetic searches: (1) an algorithm that generates random chromosomes and decodes them with the insertion heuristic, and (2) a hill-climbing algorithm that begins with a random chromosome, then always keeps the better of the chromosome and its mutated offspring. Trials of these searches generated, decoded, and evaluated the same number of chromosomes as did the GA; recall that on a TSP instance of $n$ cities this number was $500n$.

The two right columns of Table 1 show the average results of 25 trials of each non-genetic search on the test TSP instances. Both searches are surprisingly effective; the average lengths of the shortest tours they found are only slightly longer than those the GA identified. The hill-climber was consistently better than the random search, but both always found tours whose lengths were within a few percent of optimal.

Nonetheless, the GA's performance was consistently better than that of the random search and the hill-climber, and the GA was far more likely to find optimal tours on the smaller TSP instances. The insertion heuristic contributes strongly to the search for short tours and thus to the GA's success, but the GA itself focuses and accelerates the search.

## 7. DISCUSSION

Insertion heuristics have also been used in GAs for TSP that encode tours as strings of integer-valued weights associated with the cities [4, 5]. To identify the tour such a chromosome represents, each city's weight is added to all the distances in which it participates, and an insertion heuristic builds a tour using the modified distances. The length of this tour under the original distances is the chromosome's fitness. The most effective version of this algorithm used the largest sum insertion heuristic, which always inserts next the unincluded city with the largest sum of distances to cities already in the tour.

The algorithm described here, with the insertion-mediated permutation coding of tours, is nearly as effective as the best version of the weight-coded GA, and it offers two advantages. First, it does not require the specification of a maximum weight parameter. Second, it does not require, in every chromosome's evaluation, modifying the distance array and it simplifies the decoding algorithm.

Table 1: The results of sets of 25 trials of the GA, with each crossover operator, of random search, and of hill climbing on thirteen TSP instances. The number in the name of each instance is its number of cities. Under the name of each instance is the length of an optimal tour on it. The entries in the cells of the table's body list, on the left, the number of trials, out of 25, that identified optimal tours and the average number of new chromosomes generated in the trials that found optimal tours, and, on the right, the average length of the shortest tours found and that average as a percentage over the optimal tour length.

| TSP instance | GA with PMX | | GA with PPX | | Random search | | Hill climbing | |
|---|---|---|---|---|---|---|---|---|
| **st70** | 23 | 675.32 | 20 | 675.80 | 8 | 676.08 | 4 | 679.04 |
| 675 | 8268 | 0.05% | 12652 | 0.12% | 12099 | 0.16% | 3156 | 0.60% |
| **eil76** | 4 | 543.60 | 6 | 542.12 | 0 | 549.20 | 0 | 549.92 |
| 538 | 13465 | 1.04% | 12772 | 0.77% | — | 2.08% | — | 2.22% |
| **rat99** | 15 | 1212.68 | 10 | 1214.12 | 0 | 1231.84 | 0 | 1226.16 |
| 1211 | 17399 | 0.14% | 18336 | 0.26% | — | 1.72% | — | 1.25% |
| **kroA100** | 25 | 21282.00 | 25 | 21282.00 | 1 | 21312.96 | 9 | 21294.40 |
| 21282 | 5981 | 0.00% | 5113 | 0.00% | 19232 | 0.15% | 1918 | 0.06% |
| **eil101** | 3 | 632.24 | 4 | 632.72 | 0 | 643.88 | 0 | 640.84 |
| 629 | 33450 | 0.54% | 27936 | 0.59% | — | 2.37% | — | 1.88% |
| **lin105** | 23 | 14380.76 | 24 | 14380.48 | 0 | 14484.64 | 9 | 14569.24 |
| 14379 | 6779 | 0.01% | 6772 | 0.01% | — | 0.73% | 3804 | 1.32% |
| **pr107** | 21 | 44320.52 | 25 | 44303.00 | 5 | 44336.04 | 8 | 44398.76 |
| 44303 | 6332 | 0.04% | 7555 | 0.00% | 31131 | 0.07% | 5435 | 0.22% |
| **pr136** | 11 | 96824.92 | 16 | 96806.64 | 0 | 97960.68 | 0 | 97058.12 |
| 96772 | 32313 | 0.05% | 34683 | 0.04% | — | 1.23% | — | 0.30% |
| **kroA150** | 8 | 26572.60 | 1 | 26585.76 | 0 | 27014.04 | 0 | 26759.24 |
| 26524 | 19447 | 0.18% | 30789 | 0.23% | — | 1.85% | — | 0.89% |
| **pr152** | 4 | 73796.24 | 6 | 73785.36 | 0 | 73916.72 | 2 | 73970.40 |
| 73682 | 22644 | 0.16% | 10729 | 0.14% | — | 0.32% | 388 | 0.39% |
| **kroA200** | 0 | 29492.04 | 0 | 29503.16 | 0 | 30075.68 | 0 | 29694.24 |
| 29368 | — | 0.42% | — | 0.46% | — | 2.41% | — | 1.11% |
| **gil262** | 0 | 2399.80 | 0 | 2395.88 | 0 | 2464.32 | 0 | 2416.28 |
| 2378 | — | 0.92% | — | 0.75% | — | 3.63% | — | 1.61% |
| **lin318** | 0 | 42563.84 | 0 | 42604.96 | 0 | 43935.44 | 0 | 42875.00 |
| 42029 | — | 1.27% | — | 1.37% | — | 4.54% | — | 2.01% |

There are, of course, several ways in which this GA might be improved. There may be better alternatives for design choices such as the selection and deletion algorithms and parameter values. It is straightforward to seed the algorithm's population with a promising chromosome obtained by recording the order of cities inserted by a heuristic like largest sum insertion. The cities on the boundary of the convex hull of all the cities must appear in order in any optimal tour; the GA could use that boundary as the initial tour and search only the permutations of the remaining (interior) cities. The coding can be used in an island-model parallel GA, where immigration from other islands might free subpopulations from local minima.

The most effective recent GAs for TSP have employed local search, as in [7], but it is difficult to apply that technique here. Modifying chromosomes before they are decoded is best left to the genetic operators, and modifying decoded tours might identify shorter solutions, but those solutions cannot easily be re-encoded as chromosomes. Another approach would be Baldwinian: improve fitnesses through local search but do not modify the chromosomes.

## 8. CONCLUSION

This paper has presented a novel coding for TSP tours that is mediated by an insertion heuristic, two crossover operators appropriate to this coding, and a genetic algorithm for TSP that uses the coding and either of the crossovers. In tests on a range of TSP instances whose shortest tour lengths are known, the GA regularly identified short, often optimal, tours on instances containing a moderate number of cities, and it identified short, though not optimal, tours on larger instances. The two crossovers offer approximately equal performance.

The coding obtains much of its utility from the insertion heuristic that decodes it. Comparisons with two non-

genetic heuristics that also applied insertion decoding—a random search and a hill climber—indicated that while the insertion heuristic makes a large contribution to the search, the GA contributes to the search's success by focusing the search in promising regions of the search space. Various avenues for improving the algorithm remain to be explored.

## REFERENCES


[1] Lawrence Davis. *Handbook of Genetic Algorithms.* Van Nostrand Reinhold, New York, 1991.

[2] M. R. Garey, D. S. Johnson, and R. E. Tarjan. The planar Hamiltonian circuit problem is NP-complete. *SIAM Journal of Computing*, 5:704–714, 1976.

[3] David E. Goldberg and Robert Lingle, Jr. Alleles, loci, and the traveling salesman problem. In John J. Greffenstette, editor, *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, pages 154–150. Lawrence Erlbaum Associates, 1985.

[4] Bryant A. Julstrom. Strings of weights as chromosomes in genetic algorithms for combinatorial problems. In Jarmo T. Alander, editor, *Proceedings of the Third Nordic Workshop on Genetic Algorithms and their Applications (3NWGA)*, pages 33–48, August 1997.

[5] Bryant A. Julstrom. Comparing decoding algorithms in a weight-coded GA for TSP. In Janice Carroll, Gary B. Lamont, Dave Oppenheim, K. M. George, and Barrett Bryant, editors, *Applied Computing 1998: Proceedings of the 1998 ACM Symposium on Applied Computing*, pages 313–317, New York, 1998. ACM Press.

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

[7] Peter Merz and Bernd Freisleben. Genetic local search for the TSP: New results. In *Proceedings of 1997 IEEE International Conference on Evolutionary Computation*, pages 159–163. IEEE, IEEE Neural Network Council, Evolutionary Programming Society, IEEE, April 1997.

[8] Zbigniew Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer-Verlag, Berlin, third edition, 1996.

[9] Gerhard Reinelt. *The Traveling Salesman: Computational Solutions for TSP Applications.* Springer-Verlag, Berlin, 1994. Pages 82–87, 211–213.

[10] Darrel Whitley, Timothy Starkweather, and D'Ann Fuquay. Scheduling problems and traveling salesmen: The genetic edge recombination operator. In J. David Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 133–140, San Mateo, CA, 1989. Morgan Kaufmann Publishers.