The Generalized Traveling Salesman Problem: A New Genetic Algorithm Approach

John Silberholz[1] and Bruce L. Golden[2]

[1] R.H. Smith School of Business
University of Maryland
College Park, MD 20742
josilber@mail.umd.edu

[2] R.H. Smith School of Business
University of Maryland
College Park, MD 20742
bgolden@rhsmith.umd.edu

Summary. The Generalized Traveling Salesman Problem (GTSP) is a modification of the Traveling Salesman Problem in which nodes are partitioned into clusters and exactly one node from each cluster is visited in a cycle. It has numerous applications, including airplane routing, computer file sequencing, and postal delivery. To produce solutions to this problem, a genetic algorithm (GA) heuristic mimicking natural selection was coded with several new features including isolated initial populations and a new reproduction mechanism. During modeling runs, the proposed GA outperformed other published heuristics in terms of solution quality while maintaining comparable runtimes.

Key words: Generalized traveling salesman problem; genetic algorithm.

1 Introduction

The Generalized Traveling Salesman Problem (GTSP) is a variant of the well-known Traveling Salesman Problem (TSP). As in the TSP, the graph considered consists of *n* nodes, and the cost between any two nodes is known. The GTSP differs from the TSP in that the node set is partitioned into *m* clusters. An optimal GTSP solution is a cycle of minimal cost that visits exactly one node from each cluster.

The GTSP has numerous real-world applications including airplane routing [9], mail delivery [6], warehouse order picking [9], welfare agency routing [13], material flow system design [6], vehicle routing [6], and computer file sequencing [5]. Finding efficient solutions to complex GTSP problems is vital to many disciplines, especially as agencies struggle to cope with today's increased transportation costs due to higher fuel prices.

Several GTSP variations have emerged based upon the specifics of the set of nodes considered. This paper assumes symmetric costs or distances, that is, $c_{ij} = c_{ji}$, where c_{ij} is the cost or distance between nodes *i* and *j*. This means that the direction of travel between two nodes doesn't affect the cost. Additionally, some versions of the GTSP require that at least one node from each cluster be visited, instead of exactly one. While these two variations are equivalent as long as the triangle inequality holds, it may cost less to visit extra nodes if the triangle inequality does not hold. This paper assumes that exactly one node from each cluster is visited, an approach that is sometimes called the Equality GTSP (E-GTSP) [2].

Ideally, an exact algorithm, or one that always produces optimal solutions, would be most desirable. However, use of such procedures, like the one presented in [3], is not always feasible, because they tend to have prohibitively long runtimes for problems defined on a large number of nodes or clusters. For instance, the authors of [3] did not attempt to run their exact algorithm on problems larger than 442 nodes or 89 clusters because runtime of their algorithm was rapidly approaching one day. This shortcoming introduces the need for quicker heuristic methods, or approximate algorithms, which provide reasonable solutions to a problem in shorter runtimes. Examples of some heuristics for the GTSP include Snyder and Daskin's Genetic Algorithm (S+D GA) solution [14], Renauld and Boctor's GI^3 heuristic [11], Noon's generalized nearest neighbor heuristic with GI^3 improvement (NN) [11], and Fischetti et al.'s Lagrangian and root-node heuristics [3].

A genetic algorithm (GA) is a heuristic that mimics the process of natural selection. In such an algorithm, a population slowly converges to a final individual with an associated objective value after a number of iterations each of which corresponds to a new generation of that population. To facilitate this, the most desirable solutions within the population are assigned the highest survival rate from one generation to the next.

GAs store a population of chromosomes, each of which is a candidate solution for its corresponding problem (in this case, the GTSP). In each generation (iteration) of the heuristic, several operations are performed on the chromosomes to improve the overall fitness (i.e., cost) of the population. First, replication can occur, in which chromosomes are directly passed along to the next generation. These chromosomes are selected with a weighting system favoring better (lower) total cycle costs. Then, crossover, or reproduction, can occur — the GA equivalent of two parents mating and producing two children, both of whom bear a resemblance to each parent. Crossover operators that facilitate this reproduction include the partially mapped crossover (PMX) found in [4], the maximal preservative crossover (MPX) found in [8], and the ordered crossover (OX) found in [2]. A comparison of different crossovers used for the TSP can be found in [15]. Finally, mutation, a process that alters randomly selected portions of the chromosome, is also possible. For the GTSP, a common method of mutation is inversion, following [7], which is considered later in this paper.

Using the basic structure of a GA as defined in [7], this paper explores effective alternative genetic structures and crossover operators. This paper supplements the current literature by testing an effective algorithm that uses these GA improvements. The proposed GA generates high quality solutions to instances of the GTSP in reasonable runtimes.

2 The Genetic Algorithm

Data were collected on a Dell Dimension 8400 with 1.0 GB RAM and a 3.0 GHz Intel Pentium 4 processor, using programs coded in Java 1.4 and run on the Eclipse platform. This paper's GA was developed based upon a general discussion of heuristics developed for the TSP in [7]. Due to the simplicity and effectiveness of using a path representation of a TSP, as described in [7], a path representation was used for the storage of GTSP candidate solutions in chromosomes.

2.1 Path Representation

In the path representation, the most natural and simplistic way to view GTSP pathways, each consecutive node in the representation is listed in order. For instance, the chromosome (1 5 2) represents the cycle visiting node 1, then node 5, then node 2, and finally returning to node 1. Advantages of this representation include simplicity in fitness evaluation, as the total cost of a cycle can easily be calculated by summing the costs of each pair of adjacent nodes, and the usefulness of the final representation, as it directly lists all of the nodes and the order in which they are visited. However, a shortcoming of this representation is that it carries no guarantee that a randomly selected representation will be valid for the GTSP, because there is no guarantee that each cluster is represented exactly once in the pathway without specialized procedures or repair algorithms.

2.2 Population Initialization

At the beginning of the GA, each new chromosome was generated by continuously selecting random nodes and adding them to the new chromosome one by one provided that another node from the same cluster had not already been incorporated. An initial population consists of 50 of these chromosomes, a size which was deemed reasonable considering examples provided in [7].

2.3 Crossover

A novel reproductive method based upon the TSP ordered crossover (OX) operator proposed by Davis in [2] was used. The TSP's OX crossover randomly selects two cut points on one of two parent chromosomes. The nodes between these two points on the first parent are maintained in their same locations, and the remaining non-duplicate nodes from the second parent are placed, in order, at the remaining locations of the offspring, yielding a child containing ordered genetic material from both parents. For instance, from two parents p_1 = (1 | 5 4 | 3 2) and p_2 = (2 | 3 5 | 1 4), with cut points denoted by vertical bars, the material (genes) between the cut points in p_1, nodes 5 and 4, are maintained and the non-duplicate nodes from p_2, copied in order from after the second breakpoint, are nodes 1, 2, and 3. Insertion of these nodes into the offspring would yield a final chromosome c_1 = (3 5 4 1 2). Note that the inserted material, which was added after the second

Table 1. Example's explicit, symmetric distance matrix

Node (Cluster)	1	2	3	4	5	6	7	8	9	10	11	12
1 (1)	0	41	31	86	25	57	7	13	21	19	41	47
2 (1)	41	0	38	74	43	98	35	31	11	48	24	69
3 (2)	31	38	0	50	89	7	30	74	69	16	20	58
4 (2)	86	74	50	0	89	92	34	9	69	13	44	79
5 (3)	25	43	89	89	0	56	28	35	68	86	82	83
6 (3)	57	98	7	92	56	0	85	52	32	77	31	46
7 (4)	7	35	30	34	28	85	0	59	47	36	42	18
8 (4)	13	31	74	9	35	52	59	0	43	86	81	74
9 (5)	21	11	69	69	68	32	47	43	0	50	16	95
10 (5)	19	48	16	13	86	77	36	86	50	0	29	8
11 (6)	41	24	20	44	82	31	42	81	16	29	0	19
12 (6)	47	69	58	79	83	46	18	74	95	8	19	0

cut point, wraps around to the beginning of the chromosome when it reaches the end, providing for a complete offspring. Maintaining the same cut points, the other offspring would be (4 3 5 2 1). An illustration of the OX operation is provided in [7] on pp. 217-218.

A simple modification to convert this crossover to the GTSP involves insertion of nodes from the second parent whose clusters do not coincide with those of the selected nodes from the first parent.

The initial crossover mechanism was further modified by adding a rotational component. Nodes selected for insertion from the second chromosome were rotated, and numerous orientations of the nodes to be inserted were considered. For instance, instead of simply inserting the nodes from the second parent in the previous example in the order 1-2-3, the orderings 2-3-1 and 3-1-2 were also considered, and the ordering which created an offspring with the least cost was added. Though a large number of orderings are considered for larger subtours from the second parent, little computation time is expended, as only two cost evaluations are needed to determine the effectiveness of a rotation, each directly at a cut point. An additional component of this rotational crossover, which allows reversals of strings to be inserted, was also implemented. This would have yielded the additional consideration of orderings 3-2-1, 2-1-3, and 1-3-2. This reversed insertion is applicable only to a symmetric GTSP, because each reversed string would have to be completely reevaluated for an asymmetric GTSP dataset. This modified crossover, including both the rotational and reverse rotational components, will be referred to as the rotational ordered crossover, or rOX.

The rOX was further modified with an additional rotational component at the cut points. This operator rotates both of the bordering nodes from the second parent through each of the possible nodes within its cluster, selecting the one that would minimize the final cost of the tour. While this modification required significantly more runtime, it produced better solutions that tended to increase population diversity. It did so by increasing the one-generation survival probability of a promising new orientation of solutions that has not yet been locally optimized but may eventually produce better results than the current best result. This further improvement on the rOX will, hereafter, be referred to as the modified rotational ordered crossover, or mrOX. As this crossover is a defining characteristic of this paper's heuristic, the algorithm presented in this paper will be referred to as the mrOX GA. An example is provided in Table 1. Consider two parents, p_1 and p_2. They are defined based on the distance matrix provided in Table 1, and cut points were selected around the middle two nodes.

p_1 = (12 1 | 3 10 | 6 8) with cost = 297
p_2 = (2 4 | 6 8 | 10 12) with cost = 381

Nodes 3 and 10, which are between the cut points in p_1, are in clusters 2 and 5. After the right cut point (with wrap-around), p_2 visits clusters 5, 6, 1, 2, 3, and 4. Removing clusters 2 and 5 (to ensure that chromosome produced contains exactly one constituent of each cluster, making it legal) leaves, in order, clusters 6, 1, 3, and 4. Forward rotation yields the following orderings of clusters —

6, 1, 3, 4
1, 3, 4, 6
3, 4, 6, 1
4, 6, 1, 3.

Reverse rotation yields the following orderings of clusters —

4, 3, 1, 6
3, 1, 6, 4
1, 6, 4, 3
6, 4, 3, 1.

If the rOX were being performed, the nodes from p_2 in the clusters listed above would be inserted in order to the right of the nodes retained from p_1, wrapping around the chromosome if necessary. However, as an mrOX is being performed, full rotation is completed on the two clusters that border the retained nodes (the first and last clusters listed above). Thus, for the first list of clusters (6, 1, 3, 4), it is clear that the nodes to be inserted from p_2 are 12, 2, 6, and 8. However, in the mrOX, rotating through the bordering clusters also yields orderings 11, 2, 6, 8; 12, 2, 6, 7; and 11, 2, 6, 7. These four possible insertion orders are the top four orderings considered in Table 2. Table 2 contains all of the 32 possible orderings considered by the mrOX crossover, along with the associated cost of each considered pathway. It should be noted that, given a node set with *n* nodes, *m* clusters, and a distance between cut points of *d*, and an ordering of clusters from p_2 named *w*, the number of chromosomes considered is $R * \sum_{i=1}^{m} \beta_i r_i (r_f + r_g)$. *R* is the reversal constant which equals 2 if reversals are considered (the space is symmetric) and 1 if not, and β_i is the cluster exclusion constant which equals 1 if a cluster is outside of the cut points retained from p_1 and 0 if it is contained between

Table 2. Chromosomes considered in example mrOX crossover

Rotational Crossover							Reverse Rotational Crossover						
pos1	pos2	pos3	pos4	pos5	pos6	cost	pos1	pos2	pos3	pos4	pos5	pos6	cost
6	8	3	10	12	2	317	2	12	3	10	8	6	379
6	8	3	10	11	2	293	2	12	3	10	7	6	362
6	7	3	10	12	2	306	2	11	3	10	8	6	296
6	7	3	10	11	2	282	2	11	3	10	7	6	279
8	12	3	10	2	6	346	12	8	3	10	6	2	408
8	12	3	10	1	6	276	12	8	3	10	5	2	362
8	11	3	10	2	6	315	12	7	3	10	6	2	308
8	11	3	10	1	6	245	12	7	3	10	5	2	262
12	2	3	10	6	8	326	8	6	3	10	2	12	266
12	2	3	10	5	8	318	**8**	**6**	**3**	**10**	**1**	**12**	**215**
12	1	3	10	6	8	297	8	5	3	10	2	12	331
12	1	3	10	5	8	289	8	5	3	10	1	12	280
2	6	3	10	8	12	350	6	2	3	10	12	8	286
2	6	3	10	7	12	244	6	2	3	10	11	8	314
2	5	3	10	8	12	377	6	1	3	10	12	8	238
2	5	3	10	7	12	271	6	1	3	10	11	8	266

the cut points. $f = o_{(x_i+1)\,\mathrm{mod}(m-d)}$ and $g = o_{(x_i-1)\,\mathrm{mod}(m-d)}$. r_i is a function that returns the number of nodes in a cluster i, x_i is a function that returns the position of a cluster i in w, and o_q is the cluster at a certain position q in w. This equation returns 32 when considering the example crossover provided in Table 2.

As (8 6 3 10 1 12) is the possible offspring with the lowest cost, 215, this bolded entry in Table 2 becomes the actual offspring of p_1 and p_2. It should be noted that the standard OX crossover, when applied to this situation, returns the chromosome (6 8 3 10 12 2), with cost of 317.

To improve the speed of crossover execution, the distance between cut points on the first parent was increased, decreasing the number of necessary comparisons. The first cut point was randomly selected, and if it was on the right side of the chromosome, the other point was inserted at position $\left\lfloor rand^2 \frac{m}{2} \right\rfloor + 1$. Otherwise, the point was inserted at position $m - \left\lfloor rand^2 \frac{m}{2} \right\rfloor$. In these expressions, *rand* is a random real number on [0, 1) and m is the number of clusters in the dataset.

2.4 Population Structure

Additional improvements were made to the fundamental structure of a GA. First, to maintain diversity, no duplicate chromosomes (including rotations or reversals of the same chromosome) were allowed to coexist in a population. This is easily facilitated by maintaining the position of the cluster 1 gene in each of the chromosomes in the population for easier comparison to determine similarity.

Instead of a standard GA structure, which involves the evolution of one population of chromosomes into a final solution, the new structure involves isolating several groups of chromosomes for a relatively short time at the beginning of the solution procedure and using less computationally intensive genetic procedures and local improvement to rapidly generate reasonable solutions. Then, the best chromosomes from each of the smaller populations are merged into a final population, which is improved with a standard genetic algorithm structure. For the algorithm presented in this paper, seven isolated populations were maintained, each containing 50 chromosomes. After none of the populations produced a new best solution in 10 generations, the best 50 solutions from the combined pool of 350 became the final population to be improved. To ensure the speed of convergence of the initial populations, each used the rOX crossover and quicker local improvement heuristics (see Section 2.6).

In each generation, 20 of the 50 chromosomes in the population remained unaltered through replication from the previous generation. Instead of directly selecting these individuals, the thirty non-replicated chromosomes were selected through a spinner procedure, in which each chromosome was given a probability of death (with all probabilities adding to 1), and a spinner was spun to determine which chromosomes died. The affinity for death, ad_i, was calculated as $ad_i = (c_i - c_{best})^{deathPow}$ for each chromosome of index *i*, where c_i is the cost of that solution, c_{best} is the cost of the best (least cost) solution in the population, and *deathPow* is a constant that controls algorithmic convergence. The *deathPow* was set at 0.375, which was determined by experimentation to provide reasonable population diversities and convergence speeds. The probability of death of each individual chromosome was calculated by dividing each ad_i by $\sum_{i=1}^{50} ad_i$.

2.5 Reproduction

In each generation, the last 30 chromosomes added were individuals produced through reproduction. Parents were determined through a spinner selection similar to that used to determine death. The affinity for reproduction, ar_i, was calculated as $ar_i = (c_{worst} - c_i)^{reprodPow}$ for each chromosome of index *i*, where c_i is the cost of that solution, c_{worst} is the cost of the worst (most costly) solution in the population, and *reprodPow* is a constant that controls algorithmic convergence. The *reprodPow* was set at 0.375, which was determined by experimentation to provide reasonable population diversities and convergence speeds. The probability of reproduction of each individual chromosome was calculated by dividing each ar_i by $\sum_{i=1}^{50} ar_i$. Individual chromosomes can be selected more than once for reproduction.

Once a list of 30 parents was generated, each pair produced two children. Before isolated populations merged, each child was generated with the rOX crossover, but subsequent generations of offspring were created using the mrOX crossover.

2.6 Local Improvement Heuristics

Local improvement heuristics, which apply a set of transformations to a single solution, significantly improve the performance of GAs [14]. Thus, the popular *2-*

opt local improvement heuristic was implemented. In the context of a Euclidean GTSP, in which all nodes are points on a plane, this is equivalent to uncrossing two crossed pathways.

Additionally, the swap operator described in [14] was used to further strengthen local optimization in the solution. The swap operator removes each node from the tour and replaces it in every other possible position, selecting the first position that improves overall solution quality. In replacement, the node can be rotated through its cluster. Consider the example below, which uses the distance matrix from Table 1.

The chromosome considered is (2 12 3 5 7 9), with a cost of 302. The first node to be considered is 2. Insertion into each other possible position in the chromosome yields possible solutions (12 2 3 5 7 9), (12 3 2 5 7 9), (12 3 5 2 7 9), (12 3 5 7 2 9), and (12 3 5 7 9 2). The costs of these solutions are, respectively, 366, 309, 367, 316, and 302. Since none of these new positionings produced an improvement in solution cost, the other node in 2's cluster, 1, is considered in each position as a replacement for 2. The first chromosome considered, (1 12 3 5 7 9), has a cost of 290, which is lower than the cost of the initial chromosome considered, and thus becomes the final solution produced by the swap operation.

For the initial isolated populations, a lower level of local optimization was used to shorten runtime, in which the best chromosome found in the previous generation replaces the first chromosome in the current population if it is not already present, and the best chromosome in the current generation receives exactly one two-opt (or one swap if all available 2-opts are exhausted).

After the isolated populations are merged, each child produced with a better fitness (lower cost) than its parents receives full local improvement, which involves carrying out 2-opts until none are available and then swaps until none are available. Since a swap could cause a two-opt to become available, and vice versa, the cycle is repeated until no more local improvements are available. Full local optimization is also used on a randomly selected 5% of the new chromosomes produced through reproduction to improve diversity and solution quality at the cost of increased runtime.

2.7 Mutation

To facilitate mutation and thus improve population diversity, each chromosome in the population had a 5% probability of being selected for mutation, a rate similar to those used in example algorithms in [7]. If selected, two cut points were randomly selected from each chromosome's interior, and the nodes between these two points were reversed. If p_1 = (1 | 5 4 | 3 2), with the selected cut points denoted by the vertical bars, then the inverted chromosome p_1' = (1 4 5 3 2).

2.8 Termination Conditions

The algorithm terminated after the merged population did not produce a better solution for 150 generations. This termination generation count is larger than that of most genetic algorithms because the heuristic proposed has less local optimization than most other approaches.

3 Computational Experiments

The Snyder and Daskin GA (S+D GA) was selected for machine-independent comparison with this paper's mrOX GA both because it is also a genetic algorithm, and thus comparable, and because it produced some of the best heuristic results for the GTSP to date, as detailed in [14]. We implemented the S+D GA, whose attributes are detailed in [14]. In particular, we coded it in Java to produce comparable runtimes and to allow comparisons with our GA for larger datasets than those tested in [14]. The Java implementation had nearly identical performance to the Snyder and Daskin program over the datasets cited in [14], which ranged in size from 48 to 442 nodes. A two-sided paired t-test comparing results of five trials for each dataset considered in [14] with a null hypothesis that the algorithms were identical yielded a p-value of 0.9965, suggesting near-identical results. Because all heuristics rely heavily on random numbers, it is expected that the results are slightly different from the published values.

The datasets tested, as with all testing sets considered in this paper, were acquired from Reinelt's TSPLib [10]. This data source was selected because of easy Internet accessibility at softlib.rice.net, and because most papers concerning GTSP heuristics have used these datasets.

Each dataset was clustered using the procedure "CLUSTERING" described in Section 6 of [3] and implemented in, for example, [11] and [14]. This method clusters nodes based on proximity to each other, iteratively selecting $m = \lceil n/5 \rceil$ centers of clusters such that each center maximizes its distance from the closest already-selected center. Then, all n nodes are added to the cluster whose center is closest.

Computational tests were run on the data. Since Fischetti et al.'s branch and cut (B&C) algorithm provided exact values for TSPLib datasets with size $48 \leq n \leq 442$ in [3], direct comparisons with the optimal values were possible on these datasets.

Table 3 follows the format in [14] and provides for each dataset a comparison of percentage above optimal and runtime for the heuristics considered, with bolded entries denoting the best average heuristic solution quality on a dataset. The entries that are not bolded even though they have the value 0.00 indicate that modeling runs were not perfectly optimal, but that the average percentage above optimal rounded down to 0.00. The "Dataset Name" category identifies the name of the dataset considered, with the number of clusters preceding the name and the number of nodes following. For each grouping of columns, "Pct" denotes the average percentage above optimal of the run or runs and "Time" denotes the average runtime of the run or runs, in seconds. The "# Trials" row details the number of trials run per dataset for each algorithm, and the "Platform" row contains the computing platform on which testing was performed. The "GI^3" column refers to Renaud and Boctor's GI^3 heuristic found in [11], the "NN" column refers to Noon's generalized nearest neighbor heuristic followed by GI^3 improvement found

Table 3. Comparison of heuristic solution qualities and runtimes

	mrOX GA		S+D GA		GI^3		NN		FST-Lagr		FST-Root		B&C
Dataset Name	Pct	Time	Pct	Time	Pct	Time	Pct	Time	Pct	Time	Pct	Time	Time
10ATT48	**0.00**	0.36	**0.00**	0.18	*	*	*	*	**0.00**	0.90	**0.00**	2.10	2.10
10GR48	**0.00**	0.32	**0.00**	0.08	*	*	*	*	**0.00**	0.50	**0.00**	1.90	1.90
10HK48	**0.00**	0.31	**0.00**	0.08	*	*	*	*	**0.00**	1.10	**0.00**	3.80	3.80
11EIL51	**0.00**	0.26	**0.00**	0.08	**0.00**	0.30	**0.00**	0.40	**0.00**	0.40	**0.00**	2.90	2.90
12BRAZIL58	**0.00**	0.78	**0.00**	0.10	*	*	*	*	**0.00**	1.40	**0.00**	3.00	3.00
14ST70	**0.00**	0.35	**0.00**	0.07	**0.00**	1.70	**0.00**	0.80	**0.00**	1.20	**0.00**	7.30	7.30
16EIL76	**0.00**	0.37	**0.00**	0.11	**0.00**	2.20	**0.00**	1.10	**0.00**	1.40	**0.00**	9.40	9.40
16PR76	**0.00**	0.45	**0.00**	0.16	**0.00**	2.50	**0.00**	1.90	**0.00**	0.60	**0.00**	12.90	12.90
20RAT99	**0.00**	0.50	**0.00**	0.24	**0.00**	5.00	**0.00**	7.30	**0.00**	3.10	**0.00**	51.40	51.50
20KROA100	**0.00**	0.63	**0.00**	0.25	**0.00**	6.80	**0.00**	3.80	**0.00**	2.40	**0.00**	18.30	18.40
20KROB100	**0.00**	0.60	**0.00**	0.22	**0.00**	6.40	**0.00**	2.40	**0.00**	3.10	**0.00**	22.10	22.20
20KROC100	**0.00**	0.62	**0.00**	0.23	**0.00**	6.50	**0.00**	6.30	**0.00**	2.20	**0.00**	14.30	14.40
20KROD100	**0.00**	0.67	**0.00**	0.43	**0.00**	8.60	**0.00**	5.60	**0.00**	2.50	**0.00**	14.20	14.30
20KROE100	**0.00**	0.58	**0.00**	0.15	**0.00**	6.70	**0.00**	2.80	**0.00**	0.90	**0.00**	12.90	13.00
20RD100	**0.00**	0.51	**0.00**	0.29	0.08	7.30	0.08	8.30	0.08	2.60	**0.00**	16.50	16.60
21EIL101	**0.00**	0.48	**0.00**	0.18	0.40	5.20	0.40	3.00	**0.00**	1.70	**0.00**	25.50	25.60
21LIN105	**0.00**	0.60	**0.00**	0.33	**0.00**	14.40	**0.00**	3.70	**0.00**	2.00	**0.00**	16.20	16.40
22PR107	0.00	0.53	0.00	0.20	**0.00**	8.70	**0.00**	5.20	**0.00**	2.10	**0.00**	7.30	7.40
24GR120	**0.00**	0.66	**0.00**	0.32	*	*	*	*	1.99	4.90	**0.00**	41.80	41.90
25PR124	**0.00**	0.68	**0.00**	0.26	0.43	12.20	**0.00**	12.00	**0.00**	3.70	**0.00**	25.70	25.90
26BIER127	**0.00**	0.78	**0.00**	0.28	5.55	36.10	9.68	7.80	**0.00**	11.20	**0.00**	23.30	23.60
28PR136	**0.00**	0.79	0.16	0.36	1.28	12.50	5.54	9.60	0.82	7.20	**0.00**	42.80	43.00
29PR144	**0.00**	1.00	**0.00**	0.44	**0.00**	16.30	**0.00**	11.80	**0.00**	2.30	**0.00**	8.00	8.20
30KROA150	**0.00**	0.98	**0.00**	0.32	**0.00**	17.80	**0.00**	22.90	**0.00**	7.60	**0.00**	100.00	100.30
30KROB150	**0.00**	0.98	**0.00**	0.71	**0.00**	14.20	**0.00**	20.10	**0.00**	9.90	**0.00**	60.30	60.60
31PR152	**0.00**	0.97	**0.00**	0.38	0.47	17.60	1.80	10.30	**0.00**	9.60	**0.00**	51.40	94.80
32U159	**0.00**	0.98	**0.00**	0.55	2.60	18.50	2.79	26.50	**0.00**	10.90	**0.00**	139.60	146.40
39RAT195	**0.00**	1.37	**0.00**	1.33	**0.00**	37.20	1.29	86.00	1.87	8.20	**0.00**	245.50	245.90
40D198	**0.00**	1.63	0.07	1.47	0.60	60.40	0.60	118.80	0.48	12.00	**0.00**	762.50	763.10
40KROA200	**0.00**	1.66	**0.00**	0.95	**0.00**	29.70	5.25	53.00	**0.00**	15.30	**0.00**	183.30	187.40
40KROB200	0.05	1.63	0.01	1.29	**0.00**	35.80	**0.00**	135.20	0.05	19.10	**0.00**	268.00	268.50
45TS225	0.14	1.71	0.28	1.09	0.61	89.00	**0.00**	117.80	0.09	19.40	0.09	1298.40	37875.90
46PR226	**0.00**	1.54	**0.00**	1.09	**0.00**	25.50	2.17	67.60	**0.00**	14.60	**0.00**	106.20	106.90
53GIL262	**0.45**	3.64	0.55	3.05	5.03	115.40	1.88	122.70	3.75	15.80	0.89	1443.50	6624.10
53PR264	**0.00**	2.36	0.09	2.72	0.36	64.40	5.73	147.20	0.33	24.30	**0.00**	336.00	337.00
60PR299	0.05	4.59	0.16	4.08	2.23	90.30	2.01	281.80	**0.00**	33.20	**0.00**	811.40	812.80
64LIN318	**0.00**	8.08	0.54	5.39	4.59	206.80	4.92	317.00	0.36	52.50	0.36	847.80	1671.90
80RD400	**0.58**	14.58	0.72	10.27	1.23	403.50	3.98	1137.10	3.16	59.80	2.97	5031.50	7021.40
84FL417	0.04	8.15	0.06	6.18	0.48	427.10	1.07	1341.00	0.13	77.20	**0.00**	16714.40	16719.40
88PR439	**0.00**	19.06	0.83	15.09	3.52	611.00	4.02	1238.90	1.42	146.60	**0.00**	5418.90	5422.80
89PCB442	**0.01**	23.43	1.23	11.74	5.91	567.70	0.22	838.40	4.22	78.80	0.29	5353.90	58770.50
Averages	**0.03**	2.69	0.11	1.77	0.98	83.09	1.48	171.56	0.46	16.44	0.11	964.79	3356.47
# Trials	5		5		1		1		1		1		1
Platform	Dell Dimension 8400				Sun Sparc Station LX				HP 9000 / 720				

in [11], and "FST-Lagr" and "FST-Root" respectively refer to the Lagrangian and root-node heuristics found in [3].

No percentage above optimal was provided for the B&C column, as that algorithm always produces optimal solutions.

The mrOX GA produced, on average, better solution qualities than the other heuristics. Over the datasets considered in Table 3, the mrOX GA averaged only a 0.03% error, less than a third that of the S+D GA and FST-Root heuristic, the two algorithms with the nearest solution qualities. It should be noted that FST-Root had slow runtimes, running within 5% of the exact algorithm's runtime on 35 of the 41 datasets.

The solution qualities produced by the mrOX GA were also close to the published optimal solutions to certain difficult problems being investigated, like 89PCB442, an 89-cluster, 442-node GTSP dataset found in the TSPLib [10]. The

* The NN and GI^3 heuristics were not tested in [11] on these datasets.

algorithm found an optimal solution in four of the five trials run, averaging a 0.01% error over the five trials.

While previous papers presenting heuristics have, in general, limited their scope to problems for which optimal solutions have been published so that percentages above optimal can be calculated, this paper seeks to investigate larger datasets for which the exact algorithm's solution has not been determined due to prohibitively high runtimes. These datasets are clearly the ones for which heuristics are most applicable, and thus should be of the most interest to those who design approximate algorithms.

Thus, five trials were completed comparing the S+D GA and the mrOX GA based on runtime and solution quality on TSPLib datasets of size $493 \leq n \leq 1084$, with full results presented in the appendix.

Since no optimal solutions have been published for the larger problems, the success of the mrOX GA was gauged by its performance in relation to the S+D GA on the same datasets. Nearly all mrOX GA solutions to datasets had equal or superior solution qualities compared to those of the S+D GA.

Over all datasets, the mrOX GA provided 0.31% better solutions than the S+D GA, though over the larger datasets (containing more than 442 nodes), the average advantage of the mrOX GA was 1.09%. These are significant improvements, as neither the S+D GA nor the mrOX GA averaged more than 1.09% above optimal for any dataset with 442 or fewer nodes, and the average percentage above optimal for the S+D GA was just 0.11% for the smaller problems. Over the same larger datasets, the mrOX GA produced a better average solution quality than the S+D GA on 12 of the 13 datasets.

The S+D GA, meanwhile, demonstrated on average a 42.79% faster runtime than the mrOX GA. On the larger datasets tested (containing more than 442 nodes), the S+D GA had a 28.79% advantage in runtime, significantly less than the advantage over all datasets, suggesting that the runtimes will continue to remain comparable for larger datasets.

Runtime comparisons with other heuristics were difficult because different computers with various computing powers were used to test the algorithms.

Experimentation was then completed to consider the feasibility of decreasing total runtime of the mrOX GA while maintaining similar solution qualities. Decreasing the number of static generations before termination in the mrOX GA from 150 to 50 provided this effect. Experimentation (with results available in the "50-Gen Value" and "50-Gen Time (ms)" columns of Table 4 in the appendix) demonstrated an overall decrease of 16.51% in runtime, with a decrease of 0.21% in solution quality. The effects were magnified for datasets of size $493 \leq n \leq 1084$, with an overall average decrease of 47.52% in runtime and an average decrease of 0.56% in solution quality. Thus, while solution quality, not runtime, was the focus of this paper, the mrOX GA can produce results of reasonable quality very quickly if slightly modified.

Data were collected to quantify the effects of this paper's novel improvements. The population structure involving seven isolated populations, which was used in the mrOX GA, produced 0.04% better solution qualities than the 1-population (standard GA), which was also tested. Considering the small deviation from optimal for the mrOX GA (the average error for mrOX GA solutions on datasets of size $48 \leq n \leq 442$ was 0.03%), this represents a significant improvement in solution quality. However, the 20-population model tested was not significantly different from the 7-population scheme, averaging only 0.006% better solution qualities. Thus, limited benefits can clearly be gained through using isolated populations.

Naturally, maintaining more isolated populations caused a longer runtime for the heuristic. For each dataset tested, the 1-population model averaged 43.05 seconds of runtime, the 7-population model averaged 44.44 seconds of runtime, and the 20-population model averaged 49.04 seconds of runtime. As dataset size increases, the percentage of total runtime used in early improvement significantly decreases, from 48.02% for the small 22PR107 to 5.09% for the large 212U1060.

Experimentation was also carried out to determine the advantages of the mrOX crossover over the OX crossover. The mrOX crossover demonstrated a significant advantage in solution quality over the OX crossover, averaging a 0.18% increase in solution quality. The runtimes of the algorithms using the mrOX and OX crossovers were not significantly different, with the mrOX GA running on average 2.59% quicker.

4 Conclusions

Based on the data collected, the mrOX GA detailed in this paper outperformed all of the other heuristic solutions considered in terms of solution quality, while maintaining comparable runtimes, especially on larger datasets. A trend was established demonstrating an overall improvement in mrOX GA solution qualities in comparison to other heuristics like Snyder and Daskin's GA described in [14]. Additionally, the mrOX GA consistently provided optimal solutions to historically difficult datasets like 89PCB442. It could also be easily modified to provide faster solutions of good (but slightly diminished) quality.

The heuristic thus performed well in comparison to other published algorithms for run-time characteristics, and is further useful because GAs are quite simple to implement in comparison to other heuristics like the FST-Root method. Additionally, changing evaluation functions or performing basic structural transformations into other related problems like the Median Tour Problem described in [1] or Traveling Circus Problem considered in [12] are simple tasks with a GA. However, the effectiveness of these transformations would have to be investigated experimentally.

This paper's research can be applied to other GA solutions of transportation problems through the mrOX crossover. This new crossover significantly improved solution qualities while maintaining similar runtimes in comparison to the OX crossover, characteristics that make it useful in a variety of GAs. Additionally, the initial population isolation mechanism, which was proven to provide better results than a standard GA, can be applied to a wide variety of GAs. This is a far-reaching application of this paper's findings, considering the many GAs used as heuristic solutions in computing today.

Table 4: Experimental data collected

Dataset Name	Value	Time (ms)	Merge Time (ms)	Swaps	2-opts	Cross-overs	50-Gen Value	50-Gen Time (ms)	S+D GA Value	S+D GA Time (ms)
10ATT48	5394.0	356.0	118.6	3171.2	1592.6	31565.4	5394.0	209.6	5394.0	178.2
10GR48	1834.0	321.8	90.6	2238.8	1601.2	33357.8	1834.0	190.6	1834.0	75.2
10HK48	6386.0	312.8	90.8	3837.6	1102.2	30119.2	6386.0	175.2	6386.0	81.2
11EIL51	174.0	259.2	75.0	1553.2	695.0	23491.8	174.0	134.6	174.0	78.2
11BERLIN52	4040.0	315.4	87.2	2694.8	1490.8	29458.0	4040.0	196.8	4040.0	106.2
12BRAZIL58	15332.0	775.2	228.0	1798.0	1715.4	28722.6	15332.0	190.6	15332.0	97.0
14ST70	316.0	353.0	137.4	1671.4	820.6	26347.8	316.0	225.0	316.0	65.6
16EIL76	209.0	369.0	134.4	1249.2	890.0	26419.8	209.0	228.4	209.0	106.4
16PR76	64925.0	447.0	172.0	3587.2	1293.4	31823.2	64925.0	290.8	64925.0	156.2
20RAT99	497.0	500.0	169.0	2389.8	1437.2	32818.8	497.0	356.2	497.0	243.8
20KROA100	9711.0	628.2	222.0	4876.2	2007.8	39313.8	9711.0	731.2	9711.0	249.8
20KROB100	10328.0	603.2	224.8	4627.0	1682.2	37787.2	10328.0	462.4	10328.0	215.6
20KROC100	9554.0	621.8	206.4	5018.4	2335.4	39378.4	9554.0	443.4	9554.0	225.0
20KROD100	9450.0	668.8	250.0	5078.0	2464.0	39192.8	9450.0	853.0	9450.0	434.4
20KROE100	9523.0	575.0	240.6	3663.8	1721.8	37705.8	9523.0	672.0	9523.0	147.0
20RD100	3650.0	506.2	231.4	2891.6	1292.2	32891.0	3650.0	1003.2	3650.0	290.8
21EIL101	249.0	478.2	218.8	2261.0	1378.2	30152.2	249.0	1434.4	249.0	184.6
21LIN105	8213.0	603.2	256.4	3754.4	1910.0	37844.6	8213.0	1887.2	8213.0	334.4
22PR107	27898.6	534.4	256.6	1340.8	888.8	34388.4	27898.0	1537.4	27898.6	197.0
24GR120	2769.0	659.6	284.4	3176.6	1868.4	37310.2	2769.0	1606.0	2769.0	321.8
25PR124	36605.0	678.0	322.0	3734.8	1626.6	38968.6	36605.0	1118.8	36605.0	259.0
26BIER127	72418.0	784.4	334.4	5499.6	2792.4	40084.2	72418.0	906.4	72418.0	275.2
26CH130	2828.0	790.6	328.2	4126.8	2108.8	43434.2	2828.0	750.4	2828.0	418.4
28PR136	42570.0	793.8	356.2	4200.2	2066.0	38556.8	42570.0	568.8	42639.8	362.8
29PR144	45886.0	1003.2	434.6	5946.6	2776.4	53049.8	45886.0	709.2	45887.4	437.6
30CH150	2750.0	884.4	378.0	4454.6	2743.6	41030.6	2750.0	630.8	2750.0	403.2
30KROA150	11018.0	981.2	421.8	4315.0	2471.0	46399.0	11018.0	621.8	11018.0	319.0
30KROB150	12196.0	978.4	368.8	5270.4	2252.0	45276.6	12196.0	675.2	12196.0	712.4
31PR152	51576.0	965.4	349.8	5753.6	3424.4	39005.6	51577.6	587.6	51576.0	381.2
32U159	22664.0	984.4	381.2	4529.0	2789.8	42891.8	22664.0	675.0	22664.0	553.2
35SI175	5564.0	974.8	353.2	3826.4	3886.2	36402.2	5564.2	806.4	5590.4	387.2
39RAT195	854.0	1374.8	543.8	4307.8	2485.6	50919.2	854.0	868.8	854.0	1325.0
40D198	10557.0	1628.2	572.0	7795.4	3864.4	51261.8	10563.8	996.6	10564.0	1468.6
40KROA200	13406.0	1659.4	590.6	6197.6	3389.6	59078.2	13406.0	1037.2	13406.0	950.2
40KROB200	13117.6	1631.4	618.8	5786.0	2949.6	62330.8	13115.4	1081.4	13112.2	1294.2
45TS225	68435.2	1706.2	593.6	5156.8	3472.8	52474.6	68613.6	1078.0	68530.8	1087.4
46PR226	64007.0	1540.6	712.4	2783.4	2501.2	60787.4	64007.0	968.6	64007.0	1094.0
53GIL262	1017.6	3637.4	912.4	8949.2	5856.4	73077.2	1022.2	1587.6	1018.6	3046.8
53PR264	29549.0	2359.4	1012.6	4445.4	2638.2	71733.4	29549.0	1475.0	29574.8	2718.6
56A280	1080.8	2921.8	1018.8	5591.8	3314.8	68932.6	1088.8	1806.2	1080.6	3321.8
60PR299	22627.0	4593.8	1415.8	8194.2	4062.8	92713.4	22647.2	3540.8	22650.2	4084.4
64LIN318	20765.0	8084.4	1475.2	16282.6	7666.8	91508.2	21036.6	3565.6	20877.8	5387.6
80RD400	6397.8	14578.2	2453.2	19330.2	7989.2	117979.2	6413.2	8041.0	6407.0	10265.6
84FL417	9654.6	8152.8	2312.2	6724.2	5790.8	110035.2	9668.2	4553.4	9657.0	6175.2
88PR439	60099.0	19059.6	3581.6	19792.0	8235.8	143845.4	60348.2	10996.6	60595.4	15087.6
89PCB442	21658.2	23434.4	3309.4	26512.2	12235.8	137437.0	21904.0	10927.8	21923.0	11743.8
99D493	20117.2	35718.8	3675.0	33168.8	13203.8	134546.2	20146.2	21972.0	20260.4	14887.8
107ATT532	13510.8	31703.0	4440.4	24098.0	10421.6	145720.0	13520.0	20043.6	13529.8	31875.2
107SI535	13513.2	26346.8	3518.8	16626.2	19904.8	118799.6	13533.2	14543.8	13557.6	11250.2
113PA561	1053.6	21084.2	3837.6	10258.4	11026.2	127844.0	1051.2	13759.4	1065.6	26818.6
115RAT575	2414.8	48481.0	5706.0	29366.8	12684.8	177752.8	2436.4	23506.2	2442.4	46834.4
131P654	27508.2	32672.0	5909.4	11381.4	10344.2	173235.6	27439.0	17903.0	27448.4	46996.8
132D657	22599.0	132243.6	8681.2	66083.8	22186.6	218083.4	22624.0	59046.8	22857.6	58449.8
145U724	17370.6	161815.2	9921.6	62581.8	22077.2	223626.6	17681.8	58994.0	17806.2	59625.2
157RAT783	3300.2	152147.0	12421.8	48479.2	19752.6	231742.4	3330.8	68056.2	3341.0	89362.4
201PR1002	114582.2	464356.4	26940.6	89278.4	27910.8	339429.2	116058.4	295209.2	117421.2	332406.2
207SI1032	22388.8	242366.0	19397.2	32418.6	52047.0	253389.2	22415.2	126962.4	22515.2	135431.0
212U1060	108390.4	594637.4	30281.4	106204.4	30766.4	352773.6	109519.8	239453.2	110158.0	216999.8
217VM1084	131884.6	562040.6	32193.6	78499.2	32331.6	331020.4	133563.4	290765.6	133743.4	390115.6

Further research could be conducted on the effects of the novel improvements on the schemata theorem, the basic theoretical support for GAs. Additional research could consider the use of an entirely different crossover (such as the edge recombination crossover) in conjunction with a rotational inversion mechanism, or the effectiveness of a slightly modified mrOX GA on other transportation problems.

5 Appendix

In Table 4, the "Value" column contains the mrOX GA fitness value, the "Time (ms)" column contains the total mrOX GA runtime including time both before and after the population merge, the "Merge Time (ms)" column contains the mrOX GA's runtime until it merges isolated populations, the "Swaps" column contains the mrOX GA number of swaps, the "2-opts" column contains the number of mrOX GA 2-opts, the "Crossovers" column contains the number of mrOX GA crossovers, the "50-Gen value" column contains the mrOX GA fitness value for the 50-generation termination run, the "50-Gen Time (ms)" column contains the total mrOX GA runtime for the 50-generation termination run, the "S+D GA Value" column contains the S+D GA fitness value, and the "S+D GA Time (ms)" column contains the S+D GA's runtime. S+D values and runtimes are from this paper's coding of the heuristic. Fractional values are the effects of averaging results from 5 trial runs.

References

1. J.R. Current, D.A. Schilling. The median tour and maximal covering tour problems: Formulations and heuristics, *European Journal of Operational Research* 73: 114–126, 1994.
2. L. Davis. Applying Adaptive Algorithms to Epistatic Domains. *Proceeding of the International Joint Conference on Artificial Intelligence*, 162-164, 1985.
3. M. Fischetti, J.J. Salazar-Gonzalez, P. Toth. A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. *Operations Research* 45 (3): 378–394, 1997.
4. D.E. Goldberg and R. Lingle. Alleles, loci and the traveling salesman problem. In: *J.J. Grefenstette (ed.), Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, pp. 154-159. Lawrench Erlbaum Associates, Hillsdale, N.J., 1985.
5. A.L. Henry-Labordere. The record balancing problem: A dynamic programming solution of a generalized traveling salesman problem. *RAIRO* B2: 43-49, 1969.
6. G. Laporte, A. Asef-Vaziri, C. Sriskandarajah. Some Applications of the Generalized Traveling Salesman Problem. *Journal of the Operational Research Society* 47: 1461-1467, 1996.
7. Z. Michalewicz. Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag, Charlotte, NC, 1999.
8. H. Mühlenbein, M.G. Schleuter, and O. Krämer. Evolution algorithms in combinatorial optimization. *Parallel Computing* 7: 65-85, 1988.
9. C.E. Noon. The generalized traveling salesman problem. Ph. D. Dissertation, University of Michigan, 1988.
10. G. Reinelt. TSPLIB—A traveling salesman problem library. *ORSA Journal on Computing* 4: 134–143, 1996.

11. J. Renaud, F.F. Boctor. An efficient composite heuristic for the symmetric generalized traveling salesman problem. *European Journal of Operational Research* 108 (3): 571–584, 1998.
12. C.S. Revelle, G. Laporte. The plant location problem: New models and research prospects. *Operations Research* 44 (6): 864–874, 1996.
13. J.P. Saksena. Mathematical model of scheduling clients through welfare agencies. *CORS Journal* 8: 185-200, 1970.
14. L. Snyder and M. Daskin. A random-key genetic algorithm for the generalized traveling salesman problem. *European Journal of Operational Research* 17 (1): 38-53, 2006.
15. H.-K. Tsai, J.-M. Yang, Y.-F. Tsai, C.-Y. Kao. Some issues of designing genetic algorithms for traveling salesman problems. *Soft Computing* 8: 689-697, 2004.