Weight-Biased Edge-Crossover in Evolutionary Algorithms for Two Graph Problems

Bryant A. Julstrom
Department of Computer Science
St. Cloud State University
720 Fourth Avenue South
St. Cloud, MN 56301 USA
julstrom@eeyore.stcloudstate.edu

Günther R. Raidl
Institute of Computer Graphics and Algorithms
Vienna University of Technology
Favoritenstraße 9–11/1861
1040 Vienna, Austria
raidl@ads.tuwien.ac.at

Keywords
Graph problems, crossover, cost-based heuristics, traveling salesman problem, constrained minimum spanning trees

ABSTRACT
Many optimization problems on weighted graphs seek a subset of the graph's edges that has minimum weight and satisfies the problem's constraints. Two examples are the traveling salesman problem (TSP) and the degree-constrained minimum spanning tree problem (d-MSTP). Heuristics like evolutionary algorithms often construct candidate solutions to such problems iteratively, repeatedly including an edge selected from those currently eligible. Not surprisingly, low-weight edges usually predominate in good and optimal solutions, an observation we confirm empirically for the TSP and the d-MSTP. This suggests that any process that builds candidate solutions should, with higher probability, select edges of lower weight. We incorporate into crossover operators and compare in a genetic algorithm four edge-selection techniques: random, greedy, according to probabilities inversely proportional to the edges' weights, and 2-tournament. Tests on instances of the TSP and the d-MSTP indicate that with the weight-biased techniques, the GA identifies better solutions faster than with random edge-selection.

1. INTRODUCTION
An *undirected graph* $G(V, E)$ consists of a finite set V of *vertices* or *nodes* and a set E of unordered pairs of vertices, called *edges*. In a *weighted* undirected graph, a function $w : E \to \Re^+$ associates a *weight* with each edge in E. Many problems in graphs seek a subset $S \subseteq E$ of edges that satisfies the problem's particular constraints and minimizes the total weight of the edges in S:

$$w(S) = \sum_{e \in S} w(e).$$

The constraints S must satisfy characterize a specific problem; examples include the following:

- S forms a Hamiltonian path or Hamiltonian tour in which each vertex is visited exactly once (the *traveling salesman problem*).
- S forms a spanning tree of G. Often, additional constraints are imposed: In the *degree-constrained minimum spanning tree problem* [7, 8], the number of edges adjacent to each vertex is restricted to some upper bound $d > 1$. Other variants restrict edge capacities [14], the number of leaves, or the diameter of the spanning tree.
- S forms a k-edge or k-vertex connected network [11].
- S forms a Steiner tree connecting a subset of vertices from V.

Some of these problems are computationally easy. For example, several efficient algorithms exist for identifying an unconstrained minimum spanning tree. Most, however, are $\mathcal{NP}$-hard and it is unlikely that fast algorithms exist for their exact solution. In these cases, we look to heuristic optimization techniques.

Many of these techniques, such as simple greedy algorithms, simulated annealing, ant colony optimization, and evolutionary algorithms, build candidate solutions to a target problem instance by repeatedly adding one more edge to an initially empty solution until a complete, feasible solution is obtained. *Edge-selection* chooses each next edge from a set $F \subseteq E$ of eligible *candidate edges* whose inclusion in the solution would not violate the problem's constraints.

In problems that seek to minimize the total weight $w(S)$, it is not surprising that low-weight edges predominate in good and optimal solutions. This suggests that in building candidate solutions, edge-selection should favor edges in F of lower weight. In a genetic algorithm (GA), this idea may be applied in the creation of initial solutions or as crossover and mutation build new solutions [2, 6, 13, 14].

This paper examines weight-biased crossover operators in GAs for graph problems like those listed above. We investigate four edge-selection strategies—random, greedy, according to probabilities inversely proportional to edges' weights,

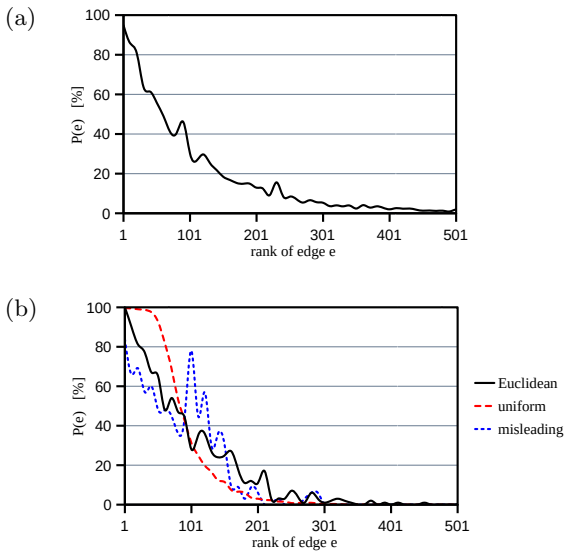

Figure 1: For test instances of (a) the TSP and (b) the 3-MSTP, the probability that an edge appears in a near-optimal solution as a function of its rank.

and 2-tournament—in the crossover operators of GAs for the traveling salesman problem (TSP) and the degree-constrained minimum spanning tree problem (d-MSTP).

The next section of this paper investigates the ranks of edges that appear in near-optimal solutions of instances of the two problems. Section 3 describes the four edge-selection strategies. Section 4 describes a genetic algorithm and codings and variation operators for the two problems. Section 5 summarizes tests on six representative instances of each problem with each edge-selection strategy implemented in the crossover operators. We confirm that the weight-biased edge-selection techniques perform far better than unbiased (random) edge-selection.

2. THE EDGES IN GOOD SOLUTIONS

It is reasonable that low-weight structures in weighted graphs should consist mostly of low-weight edges. The following empirical investigation confirms this observation in the traveling salesman and degree-constrained minimum spanning tree problems.

For the TSP, we generated 100 independent instances, each of $n = 100$ distinct vertices randomly located in a square region of the plane. Edges' weights were the Euclidean distances between their vertices, and for each instance the edges were sorted by weight, with ties resolved arbitrarily. An adequate GA identified a near-optimal tour on each instance and recorded the ranks of the edges that appeared in these tours. Figure 1(a) summarizes these tests; it shows the probability that an edge appears in a near-optimal tour as a function of the edge's rank in its instance.

About 90% of the edges in the good tours had ranks below $260 = 2.6n$ (out of $100(100-1)/2 = 4950$); that is, they were among the shortest 5.25% of edges. 96% of the edges in the good tours had ranks no greater than $400 = 4n$; that is, they were among the shortest 8.1% of edges in their instances. The highest rank of any edge that appeared in a good tour was 1651. These results, of course, hold for random problems of a particular size using Euclidean distances. Other distributions might hold for TSP instances with particular structures, of other sizes, or under other metrics.

A similar examination of d-MSTP did consider instances of three different structures. 100 instances consisted of $n = 100$ vertices randomly scattered in the plane, under Euclidean distance as in the TSP instances. 100 instances consisted of $n = 100$ vertices with edge weights assigned at random. Three instances, again of $n = 100$ vertices, had edge weights chosen to mislead simple greedy heuristics. These instances were due to Knowles and Corne [7], who named them `m100n1`, `m100n2`, and `m100n3`. The genetic algorithm from [13] identified near-optimal spanning trees of degree $d \leq 3$ on each of these instances. Figure 1(b) summarizes the results of these tests, as in Figure 1(a).

The distributions for the three classes of d-MSTP problems differ. The curve for the Euclidean instances resembles that for the TSP instances. The curve for the instances with random weights is the smoothest and most heavily favors low-weight edges. The curve for the misleading instances shows a peak just above rank 100, which is due to their special structure. Nonetheless, in all three cases, 98% of the edges that appear in near-optimal trees have ranks less than $300 = 3n$; that is, they are among the shortest 6.1% of all edges in their instances.

These results support the intuition that meta-heuristics like genetic algorithms should favor low-weight edges when building candidate solutions to graph problems like TSP and d-MSTP, and they raise the question of how to implement this favoritism.

3. EDGE-SELECTION IN CROSSOVER

When a genetic algorithm's chromosomes represent graph structures like tours or trees, crossover often builds an offspring iteratively, beginning with the empty set and repeatedly selecting a new edge from a current set F of (mostly parental) edges eligible for inclusion. Four such crossover operators are distinguished by their edge-selection strategies:

Random crossover (*RX1*): Select each edge from F at random. This is the unbiased case.

Greedy crossover (*GX1*): Always select the edge in F with the smallest weight. If several share the smallest weight, select one of them at random.

Inverse-weight-proportional crossover (*IX1*): Select each edge from F according to probabilities inversely proportional to the eligible edges' weights.

2-tournament crossover (*TX1*): Select the edge of lower weight in a 2-tournament of contestants from F. Conduct the tournament with replacement, so that the worst eligible edge has a non-zero probability of being selected.

A good crossover operator replicates parental substructures—here, edges and sets of edges—in offspring. In par-

ticular, as Starkweather *et al.* observed, a GA might benefit if crossover emphasizes the inheritance of common features of parental solutions [18]. Thus, we extend each of the crossovers to select edges in F that are common to both parents whenever possible. These extended operators are called *RX2*, *GX2*, *IX2*, and *TX2*.

4. GENETIC ALGORITHMS

The eight crossover operators were compared in genetic algorithms for the TSP and the d-MSTP. The GA is the same in both cases. It is steady-state and does not allow duplicate chromosomes in its population. It initializes its population with distinct random chromosomes. It selects chromosomes to be parents for crossover in 2-tournaments without replacement, so that the two contestants in any tournament are different, though the same chromosome may win two consecutive tournaments. It generates every offspring chromosome via crossover and mutation, as described below. If a new chromosome differs from those currently in the population, it replaces the worst chromosome; otherwise, it is discarded. The GA runs through the generation of a fixed number of distinct new chromosomes.

For TSP and d-MSTP, the GA used the codings, initialization, and variation operators that the next two sections describe. In the tests of Section 5, on a problem instance of n nodes, the GA's population size was $2n$, and it ran through the generation of $1000n$ distinct new chromosomes.

4.1 Encoding and operators for the TSP

TSP tours are encoded in the most straightforward and intuitive way: as permutations of the cities; each city participates in the two edges to its predecessor and its successor. Two such chromosomes are the same if they list the cities in the same order, regardless of the starting city, though the GA does not consider the reversal of an existing tour to be a duplicate and will allow it in the population.

All eight crossover operators build a new permutation, representing a new tour, from two parents in the following way:

> Select the first vertex (*i.e.*, city) at random; this city is current.
> Repeat $n - 1$ times:
> Identify the parental edges that connect the current city to an unvisited one; these edges form the candidate set F.
> *Edge-selection:* If F is not empty, select an edge from it; otherwise select an edge connecting the current city to an unvisited city.
> Append the city at the other end of the selected edge to the tour; this city becomes current.

The details of the edge-selection step distinguish the eight crossover operators. The most aggressive of these, GX2, is the very greedy crossover described in [6].

Many researchers have described crossovers that conform to the outline above. Perhaps the first was proposed by Grefenstette *et al.* [4]; their heuristic crossover selects the shortest parental edge to an unvisited city, if such an edge exists. Otherwise, it chooses the next city at random from those not yet in the tour. Liepins *et al.* [10], Suh and Van Gucht [19], and Jog, Suh, and Van Gucht [5] implemented variations on this greedy theme.

As Grefenstette observed, operators like these may find the set F of parental edges to unvisited cities empty, and thus need to introduce non-parental edges into the offspring [4]. In order to reduce the number of offspring edges not inherited from the parents, Whitley, Starkweather, and Fuquay [20] described edge recombination (ER), which favors parental edges to cities with the fewest connections to other cities not yet in the new tour. We do not consider this approach here, since it is specific to the TSP. The introduction of non-parental edges by crossover is an implicit mutation; therefore, we apply the mutation operator, which reverses a random subtour, only to 30% of all offspring chromosomes.

4.2 Encoding and operators for the d-MSTP

Many techniques have been proposed for encoding spanning trees in evolutionary algorithms, including random keys [17], weighted codings [12, 15], and even Prüfer numbers [21]. Here, a chromosome represents the edges of a spanning tree directly in a hash table, as in [13].

The crossover operator adapts Kruskal's algorithm for identifying an unconstrained minimum spanning tree [9]. At each step, all parental edges that connect two separated components and would not violate the degree constraint form the set F, which is usually much larger than in the case of TSP. Only when F is empty is a non-parental edge introduced into the offspring; in contrast to the TSP, this happens rarely. [13] describes a linear-time implementation of this operator.

Since crossover often creates offspring consisting only of parental edges, mutation is important to maintain variety in the population. Mutation deletes a random edge and replaces it with one that joins the resulting two components and does not violate the degree constraint. This operator is applied to every offspring produced by crossover.

5. TESTS

The eight crossover variants were compared on six TSP instances, of 50 to 200 cities, from Reinelt's data base of such problems [16, pp. 211–213]. The GA was run 50 times with each crossover on each instance; its progress was measured by the average percentage gap between the lengths $w(S)$ of the trials' shortest tours and the known optimal tour lengths w^*: $gap = (w(S)/w^* - 1) \cdot 100\%$. Table 1 summarizes these tests. In it, each cell lists the average gap after the GA has generated n, $10n$, $100n$, and $1000n$ new chromosomes, as well as the standard deviation of these values after $1000n$ chromosomes. The best results in each row are bold.

For the d-MSTP, two test instances were due to Krishnamoorthy *et al.* [8]: a Euclidean instance of 100 nodes (`crd100`) and a "structured hard" instance of 30 nodes (`shrd30`); for these instances, the maximum degree was $d = 3$. Four "misleading" instances of 50, 100, 200, and 400 nodes were due to Knowles and Corne [7]; on these instances, $d = 5$. Again, the GA was run 50 times with each crossover on each instance. Performance was reported as the average percentage gap between the weights $w(S)$ of the

best trees found and the smallest weights known w^*. Table 2 summarizes these tests in the same way as Table 1.

On both the TSP and the d-MSTP, all the weight-biased crossover variants performed significantly better than did the random variants RX1/2. With the latter crossovers, the GA slowly identified generally poor solutions, and its performance deteriorated rapidly as the problem instances became larger and more difficult.

On the six TSP instances, the greedy crossovers GX1/2 were clearly the most effective. With one or the other of them, the GA found shorter tours more quickly than with any other crossover, except on `pr136`, after having generated $1000n = 136,000$ new chromosomes. Of the remaining crossovers, IX2, which selects edges according to probabilities inversely proportional to their weights and favors edges that appear in both parents, is better, though not in general competitive with the greedy crossovers. Figure 2 plots the GA's average performance with each crossover variant on the 100-city instance `kroA100`. The figure's graphs clearly show the superiority of the greedy variants and the inefficiency of random edge-selection.

In contrast, on the d-MSTP, the purely greedy crossover variants are best only on the relatively simple instances `crd100` and `shrd30`. On the four misleading instances, the GA identifies the shortest trees with TX2, which selects edges in 2-tournaments with replacement and favors edges that appear in both parents. On these instances, the TX variants provide better performance than do the IX variants, which in turn provide better performance than do the GX variants, and crossovers that favor edges belonging to both parents outperform those that do not. Figure 3 plots the GA's average performance with each crossover on the 100-node instance `m100n1`.

These results suggest that greedy strategies might be best on Euclidean and other "easy" instances of graph problems like TSP and d-MSTP or when the running time is limited, but that edge-selection in tournaments provides better results on "misleading" instances.

Finally, note that greedy edge-selection (GX1/2) and selection according to probabilities inversely proportional to edge weights (IX1/2) are computationally more expensive than random edge-selection (RX1/2) or edge-selection in tournaments (TX1/2). This difference is small in a GA for TSP, where the number of edges in F cannot exceed four, but it can be significant in a GA for d-MSTP. On the d-MSTP instance `m400n1`, the GA took up to five times as long using IX1/2 than when using RX1/2 or TX1/2.

6. CONCLUSION

Genetic algorithms are often used to seek structures of minimum total weight in weighted graphs. This paper has described an investigation of edge-selection strategies in the crossover operators of such GAs, as applied to the traveling salesman problem and the degree-constrained minimum spanning tree problem. Crossovers that favor low-weight edges allowed a GA to identify shorter tours and trees in fewer evaluations than did operators that select edges at random from those that satisfy the problem's constraints.

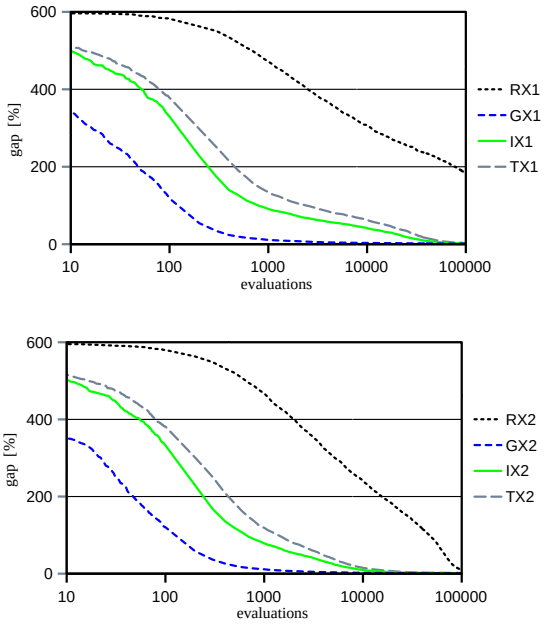

Figure 2: The percentage by which the shortest tour lengths exceeded the length of an optimal tour with each crossover variant on the TSP instance `kroA100`.

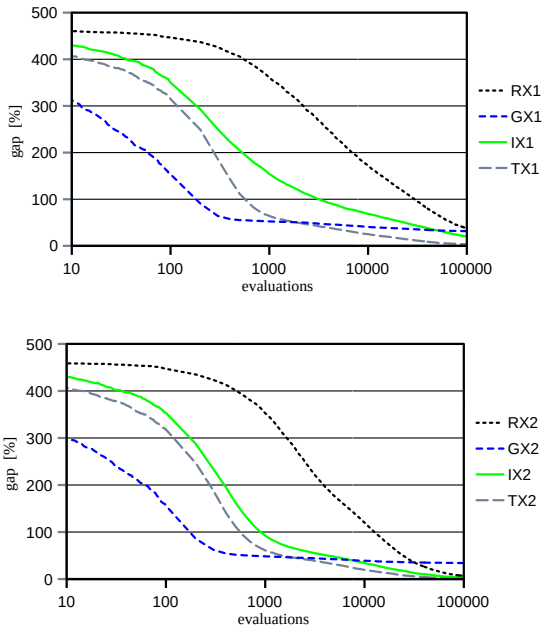

Figure 3: The percentage by which the shortest tree lengths exceeded the length of the known best solution, with each crossover variant on the 5-MSTP instance `m100n1`.

A purely greedy strategy was most effective when the GA sought short structures in the Euclidean plane. Misleading instances of the d-MSTP were better served by a less aggressive strategy that chooses edges in tournaments. Crossovers that favor edges common to both parents provided slightly better performance than those that do not.

Table 1: Average percentages by which the best tours the GA found exceeded the known optimal tour lengths, with each crossover variant on six TSP instances, after the generation of n, $10n$, $100n$, and $1000n$ new candidate tours.

Problem	Evaluations	RX1	RX2	GX1	GX2	IX1	IX2	TX1	TX2
`e50`	50	224.8	223.7	**62.7**	64.9	157.9	160.8	159.7	158.9
$n = 50$	500	166.6	158.1	9.9	**8.9**	51.8	42.4	52.5	44.9
$w^* = 425$	5000	86.5	40.1	2.2	**1.3**	12.0	4.2	7.8	3.4
	50000	12.0	2.4	**0.7**	**0.7**	1.2	1.3	1.0	1.3
	s(50000)	4.66	1.17	0.50	0.47	0.89	0.90	0.78	0.87
`e75`	75	299.3	299.0	72.4	78.1	206.8	83.9	**66.3**	210.0
$n = 75$	750	239.8	231.6	**6.3**	6.9	66.5	30.6	74.1	63.3
$w^* = 535$	7500	144.6	97.2	**0.9**	1.0	23.0	6.1	19.8	7.2
	75000	70.2	4.5	0.6	**0.4**	1.5	2.1	1.7	2.6
	s(75000)	8.0	1.6	0.2	0.6	1.0	1.3	1.2	1.3
`kroA100`	100	582.2	578.0	**118.1**	120.3	329.4	332.5	378.2	381.49
$n = 100$	1000	571.6	467.4	**11.0**	11.4	91.0	78.7	134.3	118.9
$w^* = 21282$	10000	405.5	239.4	3.2	**2.3**	41.7	9.6	62.7	15.1
	100000	283.7	10.1	1.4	**0.3**	2.0	1.5	3.0	1.9
	s(100000)	12.0	4.4	0.4	0.5	1.6	1.2	1.7	1.3
`pr136`	136	643.8	643.9	**109.0**	114.3	358.8	357.3	417.6	410.0
$n = 136$	1360	544.5	536.8	**15.3**	15.4	102.1	88.2	147.5	129.6
$w^* = 96772$	13600	366.5	312.4	9.0	**8.7**	59.3	19.3	78.1	27.0
	136000	246.2	104.9	7.0	6.6	11.3	**3.7**	15.0	4.5
	s(136000)	8.9	24.5	2.0	2.0	5.4	1.3	8.0	1.6
`kroA150`	150	744.3	745.4	128.8	**128.1**	402.8	410.1	476.5	478.3
$n = 150$	1500	629.7	624.0	**12.6**	13.2	110.2	95.9	171.9	153.2
$w^* = 26524$	15000	441.7	385.6	**3.8**	3.9	62.1	16.8	96.9	32.0
	150000	306.3	156.3	**2.1**	2.2	4.9	2.8	15.3	3.7
	s(150000)	10.0	24.9	0.6	0.8	4.9	1.6	11.1	1.3
`kroA200`	200	920.1	921.3	**142.7**	144.1	477.9	483.8	576.5	575.6
$n = 200$	2000	798.9	796.8	15.3	**15.2**	130.6	112.0	211.9	189.2
$w^* = 29368$	20000	584.5	536.6	3.7	**2.9**	82.8	29.8	131.5	56.4
	200000	435.5	300.9	1.6	**1.0**	21.7	2.7	62.7	4.1
	s(200000)	7.3	26.3	0.8	0.5	14.4	0.9	14.5	1.2

These results suggest, first, that the most effective edge-selection strategy depends on the target problem and instance and, second, that on instances of unknown or difficult structure, tournament edge-selection should be used for its simplicity, efficiency, and robustness.

7. REFERENCES

[1] C. Fonseca, J.-H. Kim, and A. Smith, editors. *Proceedings of the 2000 IEEE Congress on Evolutionary Computation*. IEEE Press, 2000.

[2] J. J. Grefenstette. Incorporating problem specific knowledge into genetic algorithms. In L. Davis, editor, *Genetic Algorithms and Simulated Annealing*, pages 42–60. Morgan Kaufmann, 1987.

[3] J. J. Grefenstette, editor. *Proceedings of the Second International Conference on Genetic Algorithms*. Lawrence Erlbaum, 1987.

[4] J. J. Grefenstette, R. Gopal, B. J. Rosmaita, and D. V. Gucht. Genetic algorithms for the traveling salesman problem. In J. J. Grefenstette, editor, *Proceedings of the First International Conference on Genetic Algorithms*, pages 160–168. Lawrence Erlbaum, 1985.

[5] P. Jog, J. Y. Suh, and D. V. Gucht. The effects of population size, heuristic crossover and local improvement on a genetic algorithm for the Traveling Salesman Problem. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 110–115. Morgan Kaufmann, 1989.

[6] B. A. Julstrom. Very greedy crossover in a genetic algorithm for the Traveling Salesman Problem. In K. M. George, J. H. Carroll, E. Deaton, D. Oppenheim, and J. Hightower, editors, *Proceedings of the 1995 ACM Symposium on Applied Computing*, pages 324–328. ACM Press, 1995.

[7] J. Knowles and D. Corne. A new evolutionary approach to the degree constrained minimum spanning tree problem. *IEEE Transactions on Evolutionary Computation*, 4(2):125–134, 2000.

[8] M. Krishnamoorthy, A. T. Ernst, and Y. M. Sharaiha. Comparison of algorithms for the degree constrained minimum spanning tree. Technical report, CSIRO Mathematical and Information Sciences, Clayton, Australia, 1999.

[9] J. B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematics Society*, 7(1):48–50, 1956.

[10] G. E. Liepins, M. R. Hilliard, M. Palmer, and M. Morrow. Greedy genetics. In Grefenstette [3], pages 90–99.

Table 2: Average percentages by which the best trees the GA found exceeded the shortest tree length found, with each crossover variant on six d-MSTP instances, after the generation of n, $10n$, $100n$, and $1000n$ new candidate tours.

Problem	Evals	RX1	RX2	GX1	GX2	IX1	IX2	TX1	TX2
`crd100`	100	637.1	635.5	139.6	**126.1**	443.3	430.5	355.9	359.1
$n = 100$	1000	479.8	454.7	**11.8**	12.5	98.9	53.2	47.2	51.5
$d = 3$	10000	164.9	114.2	**2.2**	4.1	31.2	14.4	11.8	15.0
$w^* = 6156$	100000	42.0	5.2	**0.0**	0.1	6.2	0.8	0.4	0.2
	s(100000)	4.6	1.2	0.0	0.1	1.5	0.4	0.2	0.2
`shrd30`	30	48.7	47.9	**11.5**	11.7	33.7	31.9	17.0	16.8
$n = 30$	300	27.2	23.5	**6.6**	7.3	18.0	15.2	8.7	7.9
$d = 3$	3000	6.2	3.3	**2.1**	3.3	4.6	2.2	3.6	1.4
$w^* = 2593$	30000	0.6	**0.0**	1.0	2.0	0.3	**0.0**	0.2	0.1
	s(30000)	0.7	0.1	1.5	2.1	0.3	0.1	0.3	0.1
`m050n1`	50	331.6	330.1	73.6	**72.9**	213.1	199.4	195.9	195.8
$n = 50$	500	257.8	242.2	53.0	**48.1**	102.3	72.9	54.6	54.7
$d = 5$	5000	87.9	50.1	40.1	38.6	30.9	19.9	12.8	**10.7**
$w^* = 6.601$	50000	12.3	2.2	29.1	29.2	5.0	1.7	1.5	**0.9**
	s(50000)	3.0	1.2	12.5	10.1	1.1	2.5	0.8	0.7
`m100n1`	100	449.7	447.8	160.1	**157.0**	355.6	351.9	319.3	321.8
$n = 100$	1000	361.0	352.6	52.5	**48.3**	154.3	92.4	64.4	61.0
$d = 5$	10000	171.9	119.9	40.7	39.0	68.4	34.1	25.1	**19.5**
$w^* = 11.083$	100000	37.2	6.6	31.7	34.2	20.2	3.5	3.4	**1.2**
	s(100000)	7.5	2.1	7.7	6.2	6.5	2.8	2.7	1.7
`m200n1`	200	553.1	550.1	**190.9**	192.4	427.7	431.9	368.0	363.6
$n = 200$	2000	471.6	466.0	48.8	**47.2**	177.4	89.5	76.7	78.2
$d = 5$	20000	270.3	210.0	**37.1**	39.2	76.1	32.3	43.5	37.9
$w^* = 18.335$	200000	118.1	24.9	27.2	30.2	36.1	7.4	9.8	**6.3**
	s(200000)	13.7	4.4	6.2	6.4	6.7	2.9	5.0	2.8
`m400n1`	400	356.2	355.6	**117.6**	118.0	268.8	268.5	235.0	234.8
$n = 400$	4000	320.1	317.9	53.8	52.6	140.0	83.0	43.2	**38.1**
$d = 5$	40000	200.0	146.9	52.8	51.5	76.4	43.3	31.3	**22.5**
$w^* = 56.942$	400000	122.2	43.1	49.5	48.9	49.6	25.6	18.9	**11.6**
	s(400000)	4.4	4.0	4.3	3.2	4.1	3.0	4.2	3.0

[11] I. Ljubic and J. Kratica. A genetic algorithm for the biconnectivity augmentation problem. In Fonseca et al. [1], pages 89–96.

[12] C. C. Palmer and A. Kershenbaum. Representing trees in genetic algorithms. In D. Schaffer, H.-P. Schwefel, and D. B. Fogel, editors, *Proceedings of the First IEEE Conference on Evolutionary Computation*, pages 379–384. IEEE Press, 1994.

[13] G. R. Raidl. An efficient evolutionary algorithm for the degree-constrained minimum spanning tree problem. In Fonseca et al. [1], pages 104–111.

[14] G. R. Raidl and C. Drexel. A predecessor coding in an evolutionary algorithm for the capacitated minimum spanning tree problem. In C. Armstrong, editor, *Late Breaking Papers at the 2000 Genetic and Evolutionary Computation Conference*, pages 309–316, Las Vegas, NV, 2000.

[15] G. R. Raidl and B. A. Julstrom. A weighted coding in a genetic algorithm for the degree-constrained minimum spanning tree problem. In J. Carroll, E. Damiani, H. Haddad, and D. Oppenheim, editors, *Proceedings of the 2000 ACM Symposium on Applied Computing*, pages 440–445. ACM Press, 2000.

[16] G. Reinelt. *The Traveling Salesman: Computational Solutions for TSP Applications*. Springer, Berlin, 1994.

[17] F. Rothlauf, D. Goldberg, and A. Heinzl. Network random keys – a tree network representation scheme for genetic and evolutionary algorithms. Technical Report No. 8/2000, University of Bayreuth, Germany, 2000.

[18] T. Starkweather, S. McDaniel, K. Mathias, C. Whitley, and D. Whitley. A comparison of genetic sequencing operators. In R. Belew and L. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 69–76. Morgan Kaufmann, 1991.

[19] J. Y. Suh and D. V. Gucht. Incorporating heuristic information into genetic search. In Grefenstette [3], pages 100–107.

[20] D. Whitley, T. Starkweather, and D. Fuquay. Scheduling problems and traveling salesman: The genetic edge recombination operator. In J. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 133–140. Morgan Kaufmann, 1989.

[21] G. Zhou and M. Gen. Approach to degree-constrained minimum spanning tree problem using genetic algorithm. *Engineering Design & Automation*, 3(2):157–165, 1997.