Greedy Heuristics and Evolutionary Algorithms for the Bounded Minimum-Label Spanning Tree Problem

Arindam Khaled and Bryant A. Julstrom
Dept. of Computer Science, St. Cloud State University
St. Cloud, MN 56301 USA
khar0001@stcloudstate.edu, julstrom@stcloudstate.edu

ABSTRACT

Given an edge-labeled, connected, undirected graph G and a bound $r > 1$, the bounded minimum-label spanning tree problem seeks a spanning tree on G whose edges carry the fewest possible labels and in which no label appears more than r times. Two greedy heuristics for the unbounded version of the problem are adapted to the bounded version. Two genetic algorithms for the problem encode labeled spanning trees as permutations of G's edges. A simple GA performs poorly, but the addition of local search enables consistently good results.

Categories and Subject Descriptors: G.2.1 [Mathematics of Computing]: Discrete Mathematics—*Combinatorics*; I.2.8 [Problem Solving, Control Methods, and Search]: Heuristic Methods

General Terms: Algorithms

Keywords: Labeled spanning trees, bounded labels, greedy heuristics, genetic algorithms, local search

1. INTRODUCTION

Let G be a connected, undirected graph, each of whose edges bears a label. The labels are not in general unique; each label may be attached to several edges. A minimum-label spanning tree (MLST) on G is one whose edges carry the smallest possible number of labels; the minimum-label spanning tree problem seeks a MLST on G. If no label may be used more than $r > 1$ times, we have the *bounded* minimum-label spanning tree (BMLST) problem. Figure 1 shows a labeled graph and a BMLST on it with bound $r = 3$.

Chang and Leu [2] described the MLST problem, showed that it is NP-hard, and described an A* algorithm and two heuristics for it. The Maximum Vertex Covering Algorithm was particularly effective. An evolutionary algorithm by Xiong, Golden, and Wasil [6] encoded solutions as sets of labels. It implemented a local search step that added a random label and removed redundant labels. Another, by Nummela and Julstrom [4], encoded solutions as permutations of the labels and implemented several heuristic steps.

Brüggemann, Monnot, and Woeginger [1] described the bounded problem and showed that it is NP-hard for $r \geq 3$. To our knowledge, the evolutionary algorithm below is the first for the BMLST problem. Like the MLST problem, the bounded problem finds applications in network design.

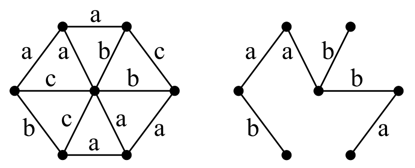

Figure 1: A labeled graph (left) and a bounded minimum-label spanning tree on the graph corresponding to the bound $r = 3$ (right).

2. TWO GREEDY HEURISTICS

Chang and Leu [2] proposed two greedy heuristics for the MLST problem that we adapt to the bounded problem. The Edge Replacement Algorithm (ERA) begins with an arbitrary spanning tree and swaps edges into and out of the tree. One step considers an edge e not in the tree and its label $\ell(e)$. It identifies the cycle created by including e in the tree and within that cycle, an edge e' whose label appears least often. If $\ell(e') \neq \ell(e)$ and if $\ell(e)$'s count in the tree is between 1 and $r - 1$, e replaces e' in the tree.

The maximum-vertex covering algorithm (MVCA) begins with an empty tree. At each step, it identifies a label whose edges cover the largest number of uncovered vertices; it includes these edges in the tree. If more than r edges carry the chosen label, r of them are selected at random.

3. TWO GENETIC ALGORITHMS

Permutations of G's edges can represent bounded-label spanning trees via a Kruskal-based [3] decoder. Given a chromosome `c[·]`, the decoder examines G's edges in their order in `c[·]`. It builds a tree by including each edge e unless e completes a cycle or the tree already contains r edges with e's label. The chromosome's fitness is the number of labels on the tree's edges, which we seek to minimize. If no tree is identified, fitness is an arbitrarily large value.

Cycle crossover [5] divides the symbols in two parent chromosomes into two groups whose sets of positions are disjoint. It copies the first group into the same positions in the offspring, then fills the offspring with the symbols in the second group, in order. This operator tends to preserve parental symbol positions and thus is appropriate here. Mutation swaps two symbols at random adjacent positions.

A local search operator scans the graph's edges in chromosome order to identify the first edge e whose label appears once in the tree and other labels whose counts in the tree are non-zero but less than r. It examines the non-tree edges

Table 1: Results of the trials of the five algorithms on the 27 BMLST problem instances: For each instance, its number of vertices n, density d, number of edges $|E|$, and number of labels $|L|$; for the A* algorithm, ERA, and MVCA on each instance, the numbers of labels in the trees they identified and the times t in seconds they required to find them; for GA and HGA on each instance, the best and mean numbers of labels in 30 trials, the standard deviation s of these values, and the mean time $\bar{t}$ that each ran.

Instance				A*		ERA		MVCA		GA				HGA			
n	d	$\|E\|$	$\|L\|$	Labels	t	Best	t	Best	t	Best	Mean	s	$\bar{t}$	Best	Mean	s	$\bar{t}$
10	0.4	20	5	2	0.1	3	0.01	4	0.0002	2	2.7	0.4	6.3	2	2.0	0.0	8.1
10	0.6	30	5	2	0.1	2	0.01	4	0.0010	2	2.6	0.5	8.6	2	2.0	0.0	9.4
10	0.8	40	5	2	0.1	3	0.02	3	0.0003	2	2.5	0.5	11.1	2	2.0	0.0	12.4
15	0.4	45	8	3	0.2	5	0.05	4	0.0004	4	4.5	0.5	15.0	3	3.0	0.0	17.1
15	0.6	67	8	3	0.2	5	0.09	4	0.0005	3	4.2	0.5	22.6	3	3.0	0.0	27.1
15	0.8	90	8	3	0.3	4	0.12	5	0.0056	4	4.1	0.2	29.3	3	3.0	0.0	40
20	0.4	80	10	4	2.3	6	0.12	5	0.0020	5	6.1	0.4	31.6	4	4.0	0.2	40.7
20	0.6	120	10	4	2.8	6	0.18	4	0.0007	5	5.9	0.5	45.4	4	4.0	0.0	67
20	0.8	160	10	4	3.3	5	0.23	4	0.0035	5	5.7	0.5	56.5	4	4.0	0.0	100.1
25	0.4	125	12	6	6729.5	10	0.20	7	0.0031	7	7.9	0.3	63.1	6	6.0	0.2	94.3
25	0.6	187	12	6	6695.6	9	0.23	7	0.0015	7	7.9	0.5	76.9	6	6.0	0.0	146.8
25	0.8	250	12	6	6643.3	9	0.43	7	0.0017	7	8.0	0.5	88.2	6	6.0	0.0	216.7
30	0.4	180	15			11	0.30	8	0.0091	9	10.3	0.6	83.5	8	8.0	0.0	142.4
30	0.6	270	15			12	0.57	8	0.0030	8	10.2	0.8	106.8	8	8.0	0.0	247.6
30	0.8	360	15			13	0.91	8	0.0028	8	10.7	0.8	118.6	8	8.0	0.0	365
35	0.4	245	18			12	0.49	10	0.0025	11	12.5	0.7	118.5	9	9.1	0.2	228.7
35	0.6	367	18			13	1.05	9	0.0019	11	13.0	0.9	144.0	9	9.1	0.2	401.4
35	0.8	490	18			13	1.74	10	0.0060	12	14.0	0.8	170.5	9	9.1	0.3	623.1
40	0.4	320	20			14	0.83	11	0.0171	13	14.7	1.0	160.6	10	10.9	0.4	347.9
40	0.6	480	20			15	1.75	11	0.0023	14	15.9	0.9	202.2	10	10.5	0.5	598.7
40	0.8	640	20			12	2.58	10	0.0143	14	16.3	1.1	152.3	10	10.6	0.6	902.9
45	0.4	405	25			17	1.39	12	0.0074	17	18.7	1.0	211.1	12	12.7	0.6	512.1
45	0.6	607	25			18	2.42	12	0.0381	17	19.2	1.1	283.5	11	12.5	0.8	866.1
45	0.8	810	25			19	4.17	12	0.0208	18	20.0	1.0	316.5	12	12.7	0.6	1321.8
50	0.4	500	28			19	1.95	13	0.0111	19	21.3	1.0	270.6	13	14.4	0.8	679.4
50	0.6	750	28			19	3.82	13	0.0130	20	22.2	1.2	328.0	13	14.4	0.8	1190.6
50	0.8	1000	28			18	6.41	13	0.0140	21	23.0	1.3	418.0	13	14.5	0.9	1811.5

for an edge that bears a different label found in the tree and that can replace e. Swapping such an edge with e in the chromosome reduces the tree's label count by one. If the tree contains no appropriate edges, the chromosome is unchanged.

Two versions of a generational genetic algorithm apply the coding and operators above. The GA initializes its population with random permutations of the graph's edges. It selects chromosomes to be parents in 2-tournaments, and it generates each offspring by applying crossover or mutation. It is 1-elitist and runs through a fixed number of generations. The simple GA (SGA) does not use local search; the heuristic GA (HGA) applies local search to every new chromosome after the initial population.

SGA and HGA were implemented in C++ and executed on a Pentium 4 processor running at 2.4GHz under Fedora Core 2.6.19. In the comparisons below, their population sizes were 250, and the probability that crossover generated each offspring was 0.85, that of mutation 0.15.

4. COMPARISONS AND CONCLUSION

An exact A* algorithm, the two greedy heuristics, and the two GAs were run on 27 instances of the BMLST problem with from 10 to 50 vertices and densities—the proportion of all possible edges—of 40%, 60%, and 80%. Their numbers of labels ranged from five to 28. The A* algorithm was run once and the remaining algorithms 30 independent times on each BLMST problem instance. ERA and MVCA were run multiple times because they involve random choices, but their results never varied, so only their single results are reported. Note that A* becomes intractable as n grows.

ERA performs well on the smaller instances, but on the larger instances, it is inferior to MVCA, which always does well. SGA also does well when n is small, but it is slower than the non-evolutionary heuristics, and as n grows, it returns poorer results even than ERA. Of the heuristics, HGA returns the best results. It finds optimum solutions where A* finds them, and does at least as well as MVCA. The local search is effective, though it increases the GA's time.

5. REFERENCES

[1] T. Brüggemann, J. Monnot, and G. J. Woeginger. Local search for the minimum label spanning tree problem with bounded color classes. *Operations Research Letters*, 31:195–201, 2003.

[2] R.-S. Chang and S.-J. Leu. The minimum labeling spanning trees. *Information Processing Letters*, 63:277–282, 1997.

[3] J. B. Kruskal. On the shortest spanning subtree and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7:48–50, 1956.

[4] J. Nummela and B. A. Julstrom. An effective genetic algorithm for the minimum-label spanning tree problem. In M. Keijzer et al., editors, *Proceedings of the 2006 Genetic and Evolutionary Computation Conference*, volume 1, pages 553–557, New York, 2006. ACM Press.

[5] I. M. Oliver, D. J. Smith, and J. R. C. Holland. A study of permutation operators on the Traveling Salesman Problem. In J. J. Grefenstette, editor, *Proceedings of the Second International Conference on Genetic Algorithms*, pages 224–230, Hillsdale, NJ, 1987. Lawrence Erlbaum.

[6] Y. Xiong, B. Golden, and E. Wasil. A one-parameter genetic algorithm for the minimum labeling spanning tree problem. *IEEE Transactions on Evolutionary Computation*, 9(1):55–60, 2005.