Local Search Heuristics for the Hop-Constrained Minimum Spanning Tree Problem

Luís Gouveia, *University of Lisbon, CIO, legouveia@fc.ul.pt*
Ana Paias, *University of Lisbon, CIO, ampaias@fc.ul.pt*
Dushyant Sharma, *University of Michigan, dushyant@umich.edu*

Keywords: hop-constrained spanning trees, local search, heuristics, neighborhoods

Introduction

The Hop-constrained Minimum Spanning Tree Problem (HMST) is defined as follows: given a graph G = (V,E) with node set V = {0,1,…,n} and edge set E as well as a cost c_e associated with each edge e of E and a natural number H, we wish to find a spanning tree T of the graph with minimum total cost and such that the unique path from a specified root node, node 0, to any other node has no more than H hops (edges) .

The HMST is NP-hard because it contains as a particular case (the case with H = 2) a NP-Hard version of the Simple Uncapacitated Facility Location problem (see Gouveia [i] and Dahl[ii]). Manyem and Stallmann [iii] have shown that the HMST is not in APX, i.e., the class of problems for which it is possible to have polynomial time heuristics with a guaranteed approximation bound. The HMST models the design of centralized telecommunication networks with quality of service constraints. The root node represents the site of a central processor (computer) and the remaining nodes represent terminals that are required to be linked to the central processor. The hop constraints limit the number of hops (arcs) between the root node and any other node and guarantee a certain level of service with respect to some performance constraints (see Woolston and Albin[iv]). Centralized terminal networks are also usually implemented with multidrop lines for connecting the terminals with the center. In such networks, node processing times dominate over queuing delays and fewer hops mean, in general, lower delays.

Lower bounding schemes for the HMSTP based on network-flow models have been suggested in Gouveia [v] [vi] and Gouveia and Requejo [vii]. More recently, Dahl et al [viii] propose a formulation involving only natural design variables and exponential sized set of constraints and propose a lower bound based on an appropriate Lagrangean relaxation. Dahl, Gouveia and Requejo [ix] summarizes these approaches. The models described in these papers view the HMSTP problem as defined in the original graph (although some of these models view the underlying hop-constrained shortest path problem as defined in an appropriate layered graph). Recently, Simonetti, Uchoa and Gouveia [x] propose a modeling approach for the HMST that views the whole problem as Steiner tree problem defined in an appropriate layered directed graph.

Apart from the work by Stefan Voss (see [xi]) which proposes a Tabu Search algorithm based on an edge exchange neighborhood for a Steiner version of the HMSTP, and a more recent work by Fernandes, Gouveia and Voss [xii] which propose and analyze the use of repetitive heuristics for the HMSTP not much has been done in terms of generating good heuristic solutions for this problem.

In this paper, we develop and study local search heuristic methods that are based on two different and "complementary" neighborhood structures, one based on edge exchanges and the other based on node-level exchanges. We also present a new dynamic programming formulation for the HMSTP. In the talk, following Ergun and Orlin [xiii], we will show that restricted polynomial versions of the dynamic program will give a

systematic way of searching neighborhood structures based on one of the neighborhood structures.

Neighborhoods and Local Search Methods for the HMSTP

A neighborhood structure for a given combinatorial optimization problem is a mapping N: S 2^S (where S denotes the set of feasible solution of the given problem) which returns a subset N_x of feasible solutions for each feasible solution x. A solution x is locally optimal with respect to a neighborhood N, if N(x) does not contain a better solution than x. A neighborhood structure N is exact for a given problem if a locally optimal solution with respect to N is optimal for the entire problem.

A neighborhood search algorithm is an iterative procedure that starts with an initial solution x and at each iteration, searches for a better solution in a neighborhood N(x). If a better solution y is found, the current solution x is replaced by y and the algorithm continues. If there is no better solution in N(x), then the algorithm terminates with the current solution.

In order to define and implement a neighborhood search algorithm, one needs to define a neighborhood structure. Furthermore, one also needs to "choose" the neighborhood structure according to some criteria, for instance: i) we need to know whether we could obtain the best solution in N(x) in polynomial time (or more loosely, find a better solution in N(x) in polynomial time) and ii) we also need to have an idea of how good it is, that is, have an idea of the quality of the solutions obtained by searching over the neighborhood. The thumb rule provided by Ahuja et al [xiv], "the larger is the neighborhood of each solution, the better is the quality of the locally optimal solutions", works well in practice. However, one should point that the rule may not always work and in fact, Orlin and Sharma [xv] have given examples of pairs of neighborhoods, one much larger than the other, with exactly the same set of locally optimal solutions.

The typical neighborhood structure for improving a spanning tree solution x (which may involve additional constraints) is defined by the set of feasible solutions that is obtained from x by adding an edge x and subsequently removing an edge which is included on the single cycle formed by the previous inclusion. It is well known that for the spanning tree problem (without any additional constraints) this neighborhood is exact. Thus, we should expect that a neighborhood search algorithm based on this neighborhood, should perform well for the HMST when the value of H is large.

To describe the node-level neighborhood we start by introducing some notation. A node is in level k in a solution, if the number of arcs from the root to that node is at most k in that solution. Consider, now, the following remark about solutions for the HMSTP problem (note that the remark also holds for the unconstrained directed spanning tree problem):

Remark: If, for each node in V\{0}, we know its level, then the best solution is uniquely determined by linking each node in level *l,* for $l = 1, \ldots, H$, to its closest node in the levels $0, 1, \ldots, l - 1$.

Clearly, this remark also holds if the given node-levels correspond to the node levels of the optimal solution. This node-level characterization suggests the following general neighborhood. Let S be a subset of V. We change the levels associated to the nodes in S. Of course we can specify additional constraints such as that the node levels may not vary beyond a certain limit. As specified above, the new solution can be easily obtained and in this case we need not check for feasibility as the node levels satisfy the hop constraints.

Clearly, the idea just described is still too vague as it is not clear yet how to choose the set S neither how to choose the new node levels. We introduce, next, three specific neighborhoods based on this idea:

Shift Neighborhood – Given a solution x, N(x) contains all solutions such that the node levels differ from the node levels of x in at most one node. Searching on this neighborhood has complexity O(nH) (this neighborhood has been proposed by Gunther [xvi] in the context of a slightly different problem, the bounded diameter minimum spanning tree).

Swap Neighborhood - Given a solution x, N(x) contains all solutions such that differ from x by swapping the levels of two nodes. Searching on this neighborhood has complexity $O(n^2)$.

Shift-Swap Neighborhood - Given a solution x, N(x) contains all solutions such that differ from x either by shfting the level of node or by swapping the levels of two nodes. Searching on this neighborhood has complexity $O(n^2)$.

As mentioned in the Introduction, we will show in the talk that searching over Shift, Swap and Shift-Swap is equivalent to search over a restricted dynamic program for the HMSTP. This dynamic program will be introduced in the next Section.

A Dynamic Programming Formulation for the HMSTP

To motivate the dynamic programming formulation described in this paper we use the observation made in the previous section concerning node levels. Suppose we have the optimal solution for an HMSTP instance with depth H. If we remove the nodes that are assigned level H (let S denote this node set), and the arcs incident into them, the resulting solution must be optimal for the HMSTP defined on the set V\S with depth H – 1. This optimality property indicates that the HMSTP can be formulated as a dynamic programming model. Consider a H-partition $\{S_1, S_2, \ldots, S_H\}$ of $V \setminus \{0\}$, i.e., $S_i \cap S_j = \varnothing$ and $S_1 \cup \ldots \cup S_H = V \setminus \{0\}$. We do not require every subset S_k to be non-empty. However, we require that if $S_k \neq \varnothing$ than $S_{k'} \neq \varnothing$ for $1 \leq k' < k$. Let S_0 denote the set $\{0\}$. We use the partition to represent a tree such that the nodes in subset S_1 are directly connected to node 0, and for k = 2, 3…, H, each node i in subset S_k is connected to a single node j in $S_0 \cup S_1 \cup \ldots \cup S_{k-1}$ such that $c_{ij} \leq c_{ij'}$ for $j' \in S_0 \cup \ldots \cup S_{k-1}$ with ties broken arbitrarily. We denote by $f(S_k,S)$ the minimum cost of connecting all the nodes in the set S_k to the closest nodes in the set S. That is $f(S_k,S) = \sum_{i \in S} c_{i,i(S)}$ where i(S) denotes the node in S closest to a node i in S_k.

In order to derive a dynamic program for the HMST, let z(S, *k*) be the cost of the optimal tree spanning the nodes in the set S and such that it has depth at most *k* . Using this definition we can derive a dynamic program for the HMSTP with depth H. The states in the k^{th} stage are given by (S, *k*) for all $S \subseteq V$. The decision at the state (S, *k*) is to determine a set S_{k+1} such that $f(S_{k+1},S)$ plus the cost z(S,*k*) of the tree associated to state (S, *k*) is minimum. The dynamic program can now be stated as follows:

$$z(S, 1) = \sum_{i \in S} c_{0i} \qquad S \subseteq V$$

$$z(S, k) = \min_{\{S_k \subseteq S\}} \{ f(S_k, S \setminus S_k) + z(S \setminus S_k, k-1) \} \qquad k = 2,\ldots,H,\ S \subseteq V$$

The optimal value for the HMSTP, v(HMSTP), is equal to z(V,H) .

The number of states for this dynamic program is $O(H.2^n)$ since the number of pairs (S, *k*) for each k, is bounded by $O(2^n)$. It is also easy to see that dynamic program can be solved in time $O(H.2^n)$ time. The dynamic program is clearly intractable even for small size problems ($n \sim 10$). But as we shall show in the talk, restricted polynomial versions of the dynamic program will give a systematic way of searching neighborhood structures based on node-level exchanges described in the previous section.

Computational Results

In this section we compare the quality of the solutions obtained by local search methods based on the previously described neighborhoods. Three of the methods are based on using the three neighborhoods defined in Section 3.3 and for simplicity, these methods will also be denoted Shift, Swap and Shift-Swap. Since searching the Shift neighborhood is computationally less expensive than searching the Swap neighborhood, the Shift-Swap method first makes some Shift moves until a solution that is locally optimal for the Shift neighborhood is found and only then proceeds by performing one "swap" move. Either there is no improvement and the solution is also locally optimal for the more complex Shift-Swap neighborhood or a better solution was found and the method proceeds, again, with a sequence of Shift moves which alternates with a Swap move and terminates when no improvement is found. We also tested the standard edge-exchange neighborhood previously mentioned as well as combined method that uses edge exchanges, shift and swap moves. This methods starts with a sequence of edge exchange moves (the motivation is, as before, to start with the less computationally expensive method), then after no improvement is possible, proceeds with a sequence of Shift moves and with a Swap move as described before. This is repeated until we obtain a solution that cannot be improved by performing any of the three types of moves. The solution obtained is locally optimal for the edge exchange and shift-swap neighborhoods.

For each method we will perform a sequence of 1000 iterations. Each local search iteration is based on random first improvement, that is, the set of nodes or edge that are candidates to be exchanged is randomly searched and the exchange is performed as soon as an improvement is found.

For this comparison we use the complete graph instances with 40 and 80 nodes described in Gouveia and Requejo [vii]. Each set contains two classes of Euclidean cost instances, depending on the location of the root. One class has the root located in the center of the grid, instances TC, and the other has the root located on a corner of the grid, instances TE. Each class contains 5 instances. The hop parameter H was set to 3, 4 and 5 in every case.

The computational results were obtained in a Pentium IV with 2.4 GHz and the heuristics where implemented in C. These results are shown in Table 1. The first column, denoted by Prob, identifies the type of instance (TC or TE) and gives the number of nodes of the corresponding graph. The second column gives the value of H. The next pairs of columns give the results obtained by the several methods. Two columns correspond to each method. The first column gives, for each class of instances, the average gaps calculated as 100*(UB – OPT)/OPT where OPT indicates the value of the corresponding optimal solutions. With the exception to the TE instances with n = 80, the optimal values of these instances were known for some time (obtained by using CPLEX and the model described in [vi]). The TE n = 80 instances were solved quite recently by the method described in

[x] (and which quite easily solves the previously solved cases). All the methods were run for 1000 iterations except for column 'Combo Search^{+}', where we have considered 10000 iterations.

Table : Average results

		Shift_Swap		Shift_LS		Swap_LS		Edge_Exchange		Combo_Search		Combo_Search^{+}	
Prob	H	% GAP	CPU	% GAP	CPU	% GAP	CPU	% GAP	CPU	% GAP	CPU	% GAP	CPU
TC 40	3	0,47	0,56	1,43	0,22	3,41	0,83	4,78	0,08	0,23	0,81	0,00	8,34
	4	1,48	0,57	1,89	0,27	4,31	0,78	4,86	0,09	0,73	0,81	0,33	8,20
	5	3,10	0,59	3,30	0,31	5,63	0,75	4,45	0,10	0,97	0,76	0,42	7,91
TE40	3	0,08	0,62	1,20	0,25	6,80	0,97	10,18	0,09	0,09	0,91	0,00	9,24
	4	0,89	0,64	1,91	0,30	7,08	0,91	7,28	0,10	0,46	0,88	0,31	9,03
	5	2,89	0,68	4,94	0,33	7,52	0,86	5,92	0,11	1,97	0,88	0,63	9,01
TC80	3	1,28	3,72	2,53	0,81	7,44	6,50	11,77	0,23	1,26	4,53	0,80	46,22
	4	2,91	3,30	4,68	0,92	7,63	5,32	8,91	0,24	2,17	4,20	1,68	43,02
	5	4,96	7,94	6,84	1,95	7,13	4,55	8,24	0,25	2,85	3,86	2,03	99,46
TE80	3	1,03	4,17	2,06	0,94	23,64	7,79	26,13	0,28	1,04	5,59	0,55	57,38
	4	2,63	3,83	4,03	1,12	15,16	6,80	16,93	0,28	2,32	5,26	1,73	53,21
	5	5,72	9,29	7,25	1,26	15,88	5,95	13,67	0,27	3,44	5,04	2,28	127,18

The main conclusions about the results are:

- Clearly, Shift is faster than Swap. In general, Shift obtains better solutions than Swap. Although the Swap neighborhood is bigger, these results might be explained by the fact that Swap is much more restrictive than Shift.
- Both Shift and Swap produce, in general, better solutions than the Edge Exchange method produces. However, the edge exchange method is the less computationally expensive method. Note also, that the edge exchange method performs better results when H = 5 (this follows, from the fact that edge exchange gives the optimal solution for the unconstrained minimum spanning tree problem) but deteriorates when H is small (this follows from the previously made observation that the edge exchange is exact when H is large).
- The other four methods obtain better results for small values of H.
- In general, improvements of the combined Shift and Swap method over Shift are bigger than the improvements of the Combo method over Shift and Swap. We should, note however, that last improvements are more hard to obtain, but we should also emphasize that the Combo method reasonably improves on Shift and Swap when H = 5.
- The gaps are worse for the TE80 instances as in this case we are only providing lower bounding values (and not optimal solutions).

Conclusions

In this paper we studied and tested several local search methods based on two complementary neighborhoods for the HMSTP. We have presented a new formulation for the HMSTP which is based on an exponential sized dynamic program. Current work is on showing that restricted polynomial versions of the dynamic program will give a systematic way of searching neighborhood structures based on the node-level neighborhood structures.

References

[i] Gouveia, L., "Using the Miller-Tucker-Zemlin Constraints to Formulate a Minimal Spanning Tree Problem with Hop Constraints," Computers and Operations Research, Vol 22, pp. 959-970, 1995.

[ii] Dahl, G., "The 2-Hop Spanning Tree Problem," O.R. Letters, Vol 23, pp 21-26, 1998.

[iii] Manyem, P., and Stallmann, M., "Some Approximation Results in Multicasting", Working paper, North Carolina State University, 1996.

[iv] Woolston, K., and Albin, S., "The Design of Centralized Networks with Reliability and Availability Constraints",Computers and Operations Research, Vol. 15, pp 207-217, 1988.

[v] Gouveia, L., "Multicommodity Flow Models for Spanning Trees with Hop Constraints", European Journal of Operational Research, Vol. 95, pp 178-190, 1996.

[vi] Gouveia, L., "Using Variable Redefinition for Computing Lower Bounds for Minimum Spanning and Steiner Trees with Hop Constraints", INFORMS Journal on Computing, Vol. 10, pp 180-188, 1998.

[vii] Gouveia, L., and Requejo, C., "A new Lagrangian Relaxation Approach for the Hop-constrained Minimum Spanning Tree Problem", European Journal of Operational Research, Vol. 132, pp 539-552, 2001.

[viii] Dahl, G., Flatberg, T., Foldnes, N., and Gouveia, L., "The Jump Formulation for the Hop-Constrained Minimum Spanning Tree Problem", Working Paper nº5, C.I.O, University of Lisbon, 2004.

[ix] Dahl, G., Gouveia, L., and Requejo, C. , "On Formulations and Methods for the Hop-Constrained Minimum Spanning Tree Problem", Handbooks of Telecommunications, Eds. Panos Pardalos and Mauricio Resende, Springer, pp 493-515, 2006.

[x] Simonetti, L., Uchoa, E., and Gouveia, C., "Modelling the hop-constrained minimum spanning tree problem over a layered graph", paper presented at the Mathematical Programming Symposium, Rio de Janeiro, 2006.

[xi] Voβ, S., "The Steiner Tree Problem with Hop Constraints", in Advances in Combinatorial Optimization edited by Sharaiha, Y., Beasley, J., Annals of Operations Research, Vol 86, pp 271-294, 1999.

[xii] Fernandes, M. Gouveia, L. and Voss, S. "Determining hop-constrained heuristics with repetitive heuristics" paper presented at the 8th INFORMS Telecom Conference, Dallas, 2006.

[xiii] Ergun, O., and Orlin, J., "Two Dynamic Programming Methodologies in Very Large Scale Neighborhood Search Applied to the Traveling Salesman Problem", MIT Sloan School of Management Working Paper 4463-03, 2004.

[xiv] Ahuja, R., Ergun, O., Orlin, J., and Punnen, A., "A Survey of Very Large Scale Neighborhood Search Techniques", Discrete Applied Mathematics 23 (2002), 75-102.

[xv] Orlin, J. and Sharma, D., "Extended Neighborhood: Definition and Characterization", Mathematical Programming 101, 537-559, 2005.

[xvi] Gruber, M., van Hemert, J., Raidl, G.R., "Neighbourhood Searches for the Bounded Diameter Minimum Spanning Tree Problem Embedded in a VNS, EA and ACO", Proceedings of GECCO'06, Seattle Washington, 2006.