The Full Degree Spanning Tree Problem

Randeep Bhatia[*] Samir Khuller[†] Robert Pless[‡] Yoram J. Sussmann[§]

1 Introduction

In this paper we study the problem of computing a spanning tree T in a connected graph $G = (V, E)$ so as to maximize the number of vertices of *full* degree (FDST). These are vertices whose degree in the tree T is the same as their degree in the graph G. First mentioned by Lewinter [5], this problem is central in the work of Pothof and Schut [7], which has to do with water distribution networks. By placing water pressure meters at every vertex of a cotree (set of edges that are the complement of a tree), one can infer the water pressure in every pipe in the network. Thus we are looking for a cotree that is incident on as few vertices as possible (to minimize cost). The *savings* is exactly the number of full degree vertices in a spanning tree, since these vertices have zero degree in the corresponding cotree and we do not have to install pressure meters at these vertices. Hence we would like to maximize the number of vertices of full degree.

Previous Work: Pothof and Schut [7] suggest the following heuristic to maximize the number of full degree vertices. Define the weight of each edge to be the sum of the degrees of the incident vertices, and compute a minimum weight spanning tree. As discussed in their paper, there are cases when their algorithm does very poorly. Consider a wheel graph (a cycle with an additional central vertex with edges to all vertices on the cycle), then their algorithm may find no vertices of full degree! In fact, for this particular graph our full degree spanning tree algorithm (for arbitrary graphs) finds a spanning tree with $\frac{n}{3}$ vertices of full degree.

Our Results: We provide a brief summary of our results [1] in this extended abstract. This problem is NP-hard even when restricted to planar graphs by a reduction from the Maximum Independent Set problem, which is known to be NP-complete for planar graphs [3]. We then consider approximation algorithms for this problem — these are algorithms that have a polynomial running time, and produce a suboptimal solution. The ratio of the optimal solution to the solution we produce (for a maximization problem) is referred to as the approximation ratio ρ.

In Section 2 we provide an algorithm with an approximation ratio of $O(\sqrt{n})$ for the FDST problem. Our worst case approximation ratio cannot be improved significantly; we can show that if there is an $O(n^{\frac{1}{2}-\epsilon})$ approximation factor in polynomial time then $NP = coR$. For the case of planar graphs, we can show that there is a polynomial time approximation scheme (PTAS).

Section 3 gives experiments on various graphs; our algorithm always found a solution size within an additive factor of 2 of the optimal, and always outperformed the heuristic of Pothof and Schut [7].

A very recent paper independently uses identical methods to find a PTAS for planar graphs and gives algorithms computing exact solutions for the problem restricted to a variety of special graphs [2].

2 Full Degree Spanning Tree Approximation Algorithm

We now present the **Greedy Star-Insertion Algorithm** that finds a good solution to the FDST problem. We show how to implement it efficiently, and then prove an $O(\sqrt{n})$ bound for its worst case approximation factor.

High Level Description: The algorithm first sorts the vertices in nondecreasing order of degree, then considers them one-by-one to be added to the current solution. Let S be the full degree vertices (initially, S is empty) and F the edges in the spanning tree that we are constructing. At each step we insert a vertex into S, together with its incident edges (a "star"), as long as it does not create a cycle. In some sense the algorithm is similar to Kruskal's MST algorithm, except that we are inserting vertices (along with all their incident edges) rather than simply adding single edges at each step.

The main hurdle regarding an efficient implementation is to check if the vertex v_i we are considering can be legally inserted or not, without creating a cycle in F. Let $N(v_i)$ be the set of neighbors of vertex v_i. The

[*]Bell Labs, Lucent Technologies, Murray Hill, NJ 07974 E-mail:randeep@research.bell-labs.com.

[†]Computer Science Dept., University of Maryland, College Park, MD 20742. E-mail:samir@cs.umd.edu. Research supported by NSF CAREER Award CCR-9501355.

[‡]Computer Science Dept., University of Maryland, College Park, MD 20742. E-mail:pless@cs.umd.edu.

[§]Computer Science Dept., University of Maryland, College Park, MD 20742. E-mail:yoram@cs.umd.edu. Research supported by NSF CAREER Award CCR-9501355.

edges incident on v_i are in two categories: edges already in F and edges not in F as yet. If the set of vertices $\{v_i\} \cup \{u|(v_i,u) \notin F$ and $u \in N(v_i)\}$ all belong to distinct connected components of F, then we can insert v_i without creating a cycle in F. However, if two vertices belong to the same component, then adding the edges incident to v_i will create a cycle in F.

The algorithm uses the Union-Find data structure to maintain the connected components induced by the current spanning forest F. The algorithm scans the adjacency list of the current vertex v_i and, for each adjacent edge not already in the forest, checks to see which component the opposite vertex lies in. If all the components thus discovered along with the component in which v_i lies are distinct then v_i can be added to the current solution. Proofs of the following can be found in [1].

THEOREM 2.1. *The* **Greedy Star-Insertion Algorithm** *delivers a solution of size at least* $\frac{OPT}{2\sqrt{2n}}$ *where* OPT *is the size of the optimal solution. Moreover, the algorithm has a worst case time bound of* $O(m\alpha(m,n))$ *where* n, m *denote the number of vertices and edges in the graph, respectively and* $\alpha(m,n)$ *is the inverse Ackermann function.*

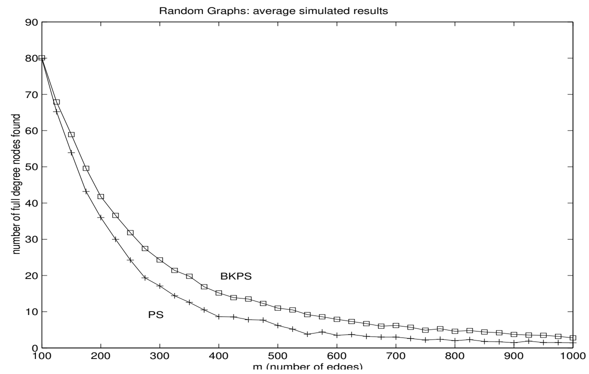
THEOREM 2.2. *Unless* $NP = coR$, *no polynomial time approximation algorithm exists for the FDST problem with an approximation ratio* $O(n^{(\frac{1}{2}-\epsilon)})$ *for any* $\epsilon > 0$.

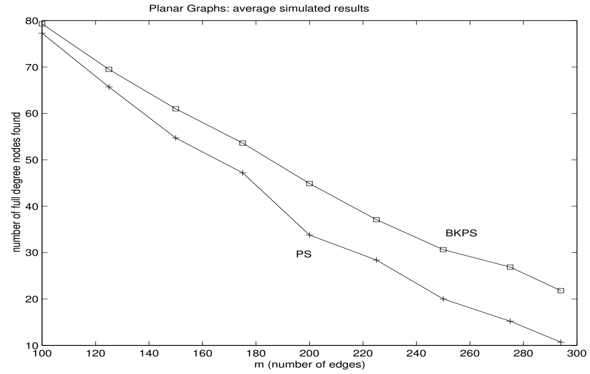
THEOREM 2.3. *For any fixed* $\epsilon > 0$, *there exists a linear time algorithm that computes a* $(1+\epsilon)$ *approximation to the FDST problem restricted to planar graphs.*

3 Experimental Results

We implemented both the Greedy Star-Insertion heuristic (labelled BKPS) and the heuristic of Pothof and Schut (labelled PS) [7]. These heuristics were run on random graphs generated using LEDA [6] and Stanford Graphbase [4]. The charts show the performance on random general and planar graphs of 100 vertices. The x axis records the number of edges in the graph, and the y axis records the number of full degree nodes found by the algorithms.

We also computed the optimal solution by writing an integer program capturing the FDST constraints, and ran it through an IP solver. It is computationally feasible to use this to compute exact solutions for graphs of less than 30 vertices. A pruning trick allows an efficient brute force strategy to search graphs of up to 100 nodes efficiently for graphs with small optimal solution sizes. For random planar graphs, the BKPS algorithm found the optimal size solution in 34 of 35 tests on random planar graphs and returned the best possible solution half of the time for arbitrary random graphs. When the BKPS algorithm failed to give the optimal solution, the solution size was within 1 of optimum (except one instance where it missed by two).





References

[1] R. Bhatia, S. Khuller, R. Pless and Y. J. Sussmann, "The Full Degree Spanning Tree Problem", Computer Science Department, Technical Report CS-TR-3931, Univ. of Maryland, (1998).

[2] H.J. Broersma, A. Huck, T. Kloks, O. Koppius, D. Kratsch, H. Müller, H. Tuinstra, "Degree-preserving forests", *Mathematical Foundations of Computer Science*, (1998) LNCS 1450, pp. 713–721.

[3] M. R. Garey and D. S. Johnson, "Computers and intractability: A guide to the theory of NP-completeness", Freeman, San Francisco, (1979).

[4] Donald E. Knuth. "The Stanford Graphbase", ACM/Addison Wesley, (1993).

[5] M. Lewinter, "Interpolation theorem for the number of degree-preserving vertices of spanning trees", *IEEE Trans. Circ. Syst.*, CAS-34, (1987), page 205.

[6] K. Mehlhorn and S. Näher, "LEDA: A platform for combinatorial and geometric computing", *Communications of the ACM*, Vol 38(1), (1995), pp. 96–102. `http://mpi-sb.mpg.de/LEDA/leda.html`.

[7] I. W. M. Pothof and J. Schut, "Graph-theoretic approach to identifiability in a water distribution network", *Memorandum No 1283*, Universiteit Twente (1995). Submitted to *Elsevier Science*.