This work was partially funded by the National Science Foundation under grant number MCS-81/05894

Abstract

In this paper we discuss applications of the Steiner tree problem on graphs which are n-dimensional hypercubes. Such problems arise frequently as models of evolutionary systems in biology, archaeology, historical reconstruction, and other problems in hierarchical clustering. We show that the weighted problem is NP-complete; we then show, with an empirically observed assumption, that many of the above Steiner tree problems can be closely approximated by a fast heuristic, and we observe that the same bounds can be obtained by other heuristics of the type commonly cited in the applied clustering literature. This provides a partial explanation for the reported effectiveness of the these algorithms in this application domain.

THE STEINER TREE PROBLEM IN PHYLOGENY

Dan Gusfield

YALEU/DCS/TR- 332
September, 1984

1. Introduction

L.R. Foulds and R.L. Graham [FG] showed that the Steiner tree problem in phylogeny is NP-complete. Technically, they showed that the Steiner tree problem restricted to undirected unweighted graphs that are hypercubes is NP-complete. The phylogeny problem they discuss is a simplification of a more general phylogeny problem which is modeled as a Steiner tree problem on *weighted* hypercubes, and although the NP-completeness of an unweighted problem generally implies the NP-completeness of the weighted version, the proof in [FG] does not imply that the weighted Steiner tree problem on hypercubes is NP-complete. The resolution of this seeming contradiction involves the size of the input needed for these two problems. In the unweighted problem, the input consists of the nodes to be connected in the Steiner tree, but omits an explicit description of the hypercube in which they appear; only the dimension of the hypercube is input, since the form of a hypercube itself is assumed to be known to the program. Further, the number of input nodes used in the Foulds-Graham reduction is exponentially smaller than the number of nodes of the hypercube in which they are embedded. The case of the *weighted* Steiner tree problem in hypercubes, however, is different: all the edge weights need to be explicitly represented in the input. Hence the size of the input in the weighted case is exponentially larger than the size of the input used in [FG]. This leaves the possibility that the weighted case can be solved in polynomial time as a function of the input size.

In this paper we present a simple proof that the weighted Steiner tree problem on hypercubes is NP - complete. We then show that, with an empirically observed assumption, a known heuristic for the Steiner tree problem on arbitrary graphs is particularly effective for the phylogeny problem and other hierarchical clustering problems modeled either as weighted or unweighted Steiner tree problems. We finally note that commonly cited heuristics in the applied clustering literature also achieve the same provable bounds as this heuristic.

Definitions: Let G = (N,E) be an undirected graph on node set N and edge set E with a non-negative weight w(i,j) on each edge (i,j) in G. Let $X \subseteq N$ be a subset of nodes, and $|X| = x$. A Steiner tree in G is any subtree of G that contains all the nodes of X, although it may contain other nodes in N-X. The weight of a Steiner tree ST is the sum of the weights of all the edges in ST, and is denoted W(ST). The optimal Steiner tree is the Steiner tree of minimum weight, and the weighted Steiner tree problem is to find the optimal Steiner tree. When all edges have weight one, the problem is called the unweighted Steiner tree problem.

A hypercube of dimension k is an undirected graph with 2^k nodes, where the nodes are labelled with the integers between 0 and 2^k-1. Two nodes in the hypercube are adjacent if and only if the binary representation of their labels differs in exactly one bit. The weighted Steiner tree problem on hypercubes is the weighted Steiner tree problem where the graphs are hypercubes.

1.1. Motivations for the Steiner tree problem on hypercubes

The Steiner tree problem on hypercubes frequently models problems of hierarchical clustering that arise in biology, archaeology, and historical reconstruction, etc. (see [HKT] for numerous examples in archaeology and history, and see [W], [D], [F1], [F2], [FM] and [WA] for applications in biology). The basic model in each of these applications is that a set of objects (animals, gravesites, manuscripts, amino acid sequences, etc.) is described by a set of characteristics that are each either present or absent in each object. With a set of k characteristics, there are 2^k possible distinct objects, of which only a few are generally known to exist or to have existed. Two objects are directly related if their characteristic vector differs in exactly one characteristic. Hence the set of potential objects and the relations on them can be represented as a hypercube of dimension k, where each object is a node in the hypercube and each relation is an edge. Given two related objects (adjacent nodes) there may be a weight representing some measure of their difference. In the case of molecules this weight might represent the energy needed to transform one to the other; in evolution it might represent time needed for one species to evolve into the other; or in general it might represent some measure of dissimilarity or unlikliness of the relation. In each of the mentioned applications, the objective is to find the least weight tree in the hypercube that connects the known objects. This is exactly the weighted Steiner tree problem on hypergraphs. The significance of the optimal tree differs in each application, and is often controversial. In some applications the claim is made that the tree represents a possible history of how the objects evolved; in other applications it is used only to organize the objects; and in some applications it is only the weight of the optimal tree that is believed to be of value.

Foulds and Graham describe one of the classical and widely used applications of the above model: inferring an evolutionary tree, or phylogeny, from characteristics of certain amino acid sequences. Another concrete application (see [NA], [NI]) is the problem of reconstructing a genealogy of manuscripts. The problem is that there is a set X of copies of a given manuscript, which each vary from the other in some small ways. These variances are believed to have been introduced by errors created and propagated in hand copying of the the manuscripts over time.

In this copying process, errors are introduced, but may also be corrected. It is believed that there are lost copies of the manuscript that form part of the copying process. The characteristics that describe the set of manuscripts are all the locations where variances occur (each variance is assumed to be either there or not). The problem is to reconstruct a possible genealogy of the manuscripts, under the assumption that it forms a tree, and that the tree involves as few manuscripts, in X or not, as possible. This is exactly a Steiner tree problem on hypercubes where each of the nodes of the hypercube represents a characteristic vector of variances, and the known manuscripts form the set X.

2. The weighted Steiner tree problem on hypercubes in NP-complete

We reduce the unweighted Steiner tree problem on the grid to the weighted Steiner tree problem on a hypercube. The former problem has been shown (implicitly) to be NP-complete [GJ]. The idea of the reduction is that we will embed any mxn grid into a hypercube which has at most 4mn nodes. In particular, we will label the mn grid points with distinct k-bit labels so that two nodes on the grid are adjacent if and only if their labels differ in exactly one bit, and hence the adjacency relations on the grid will be mapped into the k dimensional hypercube. We will see that for any m and n, the number of nodes in the hypercube (2^k) will be at most 4mn.

In the above embedding, there will be adjacency relations on the hypercube that don't occur in the grid. To deal with these, we give a weight of 1 to every edge in the hypercube corresponding to an edge in the grid, and a weight of 2mn to every other edge in the hypercube. The effect is that for any Steiner tree problem in the grid, the only edges in the hypercube that will be used in the optimal solution are those of weight 1. Hence a Steiner tree problem on the grid has weight less than or equal to some number z if and only if the imbedded problem on the hypercube has a solution of weight z or less.

In order for this embedding to be an NP-completeness proof, we must verify that it requires only polynomial time in the size of the input to the Steiner tree problem on *grids*. This is a technical point requiring some care. The Steiner tree problem on grids that is (implicitly) shown to be NP-complete in [GJ] is best stated as follows: Given n integer coordinate pairs, X, and some target z, is there a Steiner tree connecting X of weight z or less which runs along the integer grid enclosing those n points. The key point is that in this grid problem, as in the unweighted hypercube problem, the description of the grid is not explicitly given as input; only the

coordinates of the nodes in X are input. These points are embedded in an some mxn grid, but the size of X might be exponentially smaller than mn. If this were the case, then the above reduction of grids into hypercubes does not qualify as a polynomial time reduction, even thought the size of the hypercube is at most four times that of the grid. Fortunately, in the reduction used in [GJ], the n points in X lie on a grid which has at most $|X|^2$ grid points. Hence the unweighted Steiner tree problem on grids, where the grid is explicitly described in the input, is also NP-complete and hence so is the weighted Steiner tree problem on hypercubes.

2.1. How to label the grid

We consider rectangular grids of size mxn, where m and n are each a power of two. Any other rectangular grid can be embedded in such a grid which has at most four times the number of nodes.

For m=1, $n = 2^p$ we know there is an appropriate labelling of the nodes with p bit numbers; this is just a Grey code numbering of the numbers between 0 and 2^p-1. Now for $m = 2^q$, assume there is an appropriate labelling of the nodes with $p+q = \log_2(mn)$ bit numbers. We can obtain a labelling for the (2m)xn grid using p+q+1 bits by flipping the mxn grid around the horizontal axis and appending it below a copy of the original mxn grid; we add one bit to the label of each node in the two mxn grids. In the top mxn grid we set this bit to zero, in the bottom mxn grid we set it to 1. In this step we have doubled the number of nodes in the grid, and only introduced one new bit. So for m and n both powers of two, there is an appropriate labelling using exactly $\log_2(mn)$ bits, and so general mxn grids can be embedded into hypercubes with at most 4mn nodes. Note that what we have constructed is a two dimensional version of a Grey code.

3. Approximations for the Phylogeny problem

In this section we discuss a heuristic algorithm for the Steiner tree problem which gives particularly good approximations for the Phylogeny problem and other similar clustering problems modeled as Steiner tree problems. We first briefly describe a heuristic developed in [KMB] that achieves a factor of two approximation of the optimal Steiner tree for arbitrary graphs. We then observe that the algorithm achieves better than a factor of two approximation for a certain class of Steiner tree problems; we believe that many clustering problems modeled as Steiner tree problems fall into this class. Finally, we indicate that these approximation bounds also hold for several Steiner tree heuristics commonly cited in the applied clustering literature.

This provides an partial explanation for the reported effectiveness of these algorithms in the applications discussed above.

Algorithm KMB: a factor of two approximation for Steiner tree

Briefly the algorithm given in [KMB] is:

1. Construct a complete weighted graph G′ on the nodes of G in X, and for each pair of nodes (i,j) in X, assign the weight d(i,j) to edge (i,j), where d(i,j) is the length of the shortest path between i and j in G.

2. Computer a minimum spanning tree, T, in G′ connecting the nodes of X. For each edge (i,j) in T, find a shortest path between nodes i and j in G. Let T′ be the graph formed by the union of all of these paths.

3. If T′ is a tree, then stop, else find the minimum spanning tree of T′, and then successively delete any leaves that are not in X. The result is a Steiner tree that is at most twice the weight of the optimal Steiner tree connecting node set X in G.

The factor of two actually applies to the minimum spanning tree T produced in step 2. The graph T′ has weight at most that of T, and this weight is maintained or further reduced in step 3. We will need to refer to the analysis in [KMB] of the factor of two bound for T. This is briefly as follows:

Consider the edges of T and for each edge (i,j) in T consider a shortest path in G between i and j. The weight of T is exactly the sum of the length of each of these paths (note that we are likely counting some edges more than once). Hence T defines a set of paths in G which connect the nodes of X, and in which every node of X is an endpoint of at least one of the paths. Any such set of paths is called a *connecting set* of paths. Then T, in fact, defines the least weight connecting set of paths in G, when the weight of an edge is counted each time it appears in a path.

Now let ST be the optimal Steiner tree connecting X in G. Consider a depth first traversal, P, of ST; P traverses every edge exactly twice. Now if we split P into paths that have an X node on each end, and contain no X nodes in between, then P can be considered to be the concatenation of x paths between the nodes of X. Therefore, P defines a set of paths that connect the nodes of X in G, and in which every node of X is the endpoint of at least one path. That is, P defines a connecting set of paths. But T defined to be the cheapest connecting set of paths, and so the

total length of the paths defined by T is at most the total length of the paths defined by P, which is exactly 2W(ST).

As observed in [KMB] the bound of two is too large. Let d be the maximum distance in ST between two nodes of X that are connected by a path containing no other nodes of X. Then the weight of T is at most 2W(ST) - d. This will follow as a special case of the analysis below.

3.1. Steiner trees in phylogeny and other clustering applications

In this section we modify the above analysis to include more information about the structure of ST. In particular, we observe the effect on the worst case bound when ST contains interior nodes from X [1]. We assume that the depth first search of ST begins with a node of X, and consider this node an interior node, even if it has degree one.

Definition: Let ST′ be defined as the directed tree obtained from ST by orienting each edge in ST in the direction it is first traversed in P. Let i be a node of X with outdegree d(i), and for each edge e directed out of i in ST′, let I(e) be the set of X nodes which are reachable from i by a directed path which traverses edge e, and which contains no X nodes other than the endpoints of the path; let X(e) be the node in I(e) which is farthest from node i. Let X(i) denote these d(i) paths from node i, and let w(i) be the total weight of the paths. See figure 1.

Theorem: The minimum spanning tree T of step 2 above has weight at most 2W(ST) -Σw(i).

Proof: The basic idea is that any connecting set of paths have weight at least that of T. We saw above that P defines a connecting set of paths that used every edge in ST at most twice. We want to extract a connecting set of paths from P that have a smaller total length; We will do this by modifying the order of the depth first search.

As before, we can think of P as a collection, C, of paths that run between each successive X node encountered on P. These paths form a connecting set of paths in G. But observe that we can delete any path in C which backs up to a node of X in P, and still have a connecting set of paths. For example in figure 2, the path from v to w in P can be deleted. The set of paths obtained in this way is still a connecting set of paths, and hence has total length at least as great as the weight of T, but has less than the total length of P. Therefore, the weight of T is bounded

---

[1] Reasoning of the type used here is more finely developed in [SU] to improve the general bounds of [KMB]

by 2W(ST) minus the total length of the paths removed.

Now the above argument holds for any depth first search, so the idea is to find a depth first search that maximizes the savings obtained by the above observation. We will show a depth first traversal, P′, of ST such that for each internal X node i, every path in X(i) backs up to node i in P′. That means that in P′, if i is any internal X node and e is any edge directed out of i, then node i is the first X node visited after X(e). Equivalently, X(e) is the last node in I(e) that is visited in P′. It is easy to modify the rules of the depth first search to achieve this for each internal X node: In ST′, if (u,v) is a directed edge from u to v, and on a path in X(i), then it should be traversed in the forward direction only after all other edges out of u have been traversed (twice). The existence of traversal P′ proves the theorem. □

The bound of ST -Σw(i) is of interest for the class of problems where Σw(i) is a large percentage of W(ST). Many of the clustering applications involving the evolution or derivation of a system of objects often exhibit this property. This is empirically observed, but is also suggested by the nature of the applications: the nodes of X form part of an evolutionary system in which one node spawns or generates others. In fact, for some of these problems, *most* of the nodes in X are interior nodes in ST. This happens when there are long chains of derivations along which there is no branching, i.e. each node (both X and non-X nodes) along the path has at most one descendant. Even in the case when the existing species are likely to be leaves of the evolutionary system, fossil remains may exist which form part of X and which these are often internal nodes of the optimal tree. This is also true for problems in architecture and archaeology where one type of building or tool precedes and leads to the development of other later objects. Further, in the clustering problems we have been discussing, the nodes of ST-X represent objects in the system for which all evidence has been lost. But in many instances, most of the relevant objects are known, so that the percentage of internal nodes not in X is small.

There are other characteristics of optimal Steiner trees arising in the above applications which further support the claim that algorithm KMB achieves a better bound than two for these problems. For example, in some problems |I(e)| is small, independent of the size of X. When |I(e)| $\leq$ 2, then T is within 1.5W(ST) for the weighted problem, and within 1.33W(ST) in the unweighted Steiner tree problem.

3.2. Other good heuristics

The approximation algorithm KMB discussed above did not originate in the clustering literature. In this section we suggest that many reasonable heuristics for Steiner tree will perform as well as the KMB algorithm. We look at one algorithm which is typical of the heuristics that appear in the applied clustering literature. The Steiner tree constructed is called H.

1. Find the two nodes, i,j in X of shortest distance in G, and a shortest path that connects them. The nodes and edges on this path are put into H. Nodes i and j are deleted from X.

2. While there are nodes in X not in H find the shortest path in G that connects a node v in X but not in H to a node in H. Add the nodes and edges on that path to H. Delete v from X.

This algorithm can be viewed an adaptive version of algorithm LMB KMB, when the minimum spanning tree in KMB is computed by Prim's algorithm, i.e. by always adding to a single connected component in the cheapest way. In KMB the weights for G' are computed once and fixed throughout the minimum spanning tree computation. Here the minimum spanning tree algorithm begins with the weights used in KMB, but, essentially, may reduce some of the weights after each iteration. These distance reductions amount to assuming that any edge in $G \cap H$ now has distance zero, so that if the edge weights in G' reflect the current distances in G, the weights in G' between X nodes not in H and X nodes in H may be reduced. The question is how the resulting tree, H, compares to T. It seems intuitive that H has weight no greater than T, and this is a consequence of the following two lemmas that we state without proof.

Lemma 1: If C_1 and C_2 are two weight vectors for the edges of a graph G, and $C_1 \leq C_2$ in each component, then the weight of the minimum spanning tree when the weights of G are given by C_1 is less than that when the weights are given by C_2.

Lemma 2: At any point in the running of Prim's algorithm, let G'' be the graph obtained from G' by contracting the edges already selected. Then the minimum spanning tree of G' consists of these edges plus the edges in the minimum spanning tree of G''.

Acknowledgement

Thanks to Gregory Sullivan and Leonard Pitt for helpful discussions.

4. References

[D] T. Duncan, "Cladistics for the Practicing Taxonomist - An Eclectic View", Systematic Botany, (1980) 5(2).

[F1] J. S. Farris, "Methods for Computing Wagner Trees", Systematic Zoology 19, 1970.

[F2] J. S. Farris, "Inferring Phylogentic Trees from Chromosome Inversion Data", Systematic Zoology 28, 1980.

[FM] W. M. Fitch and E. Margoliash, "The Construction of Phylogentic Trees", Science, 155, 1967.

[FG] L.R. Foulds and R.L Graham, "The Steiner Tree in Phylogeny is NP-Complete". Advances in Applied Math, Vol. 3, No. 1, March 1982.

[GJ] M.R. Garey and D.S. Johnson, "The Rectilinear Steiner Tree Problem is NP-Complete". Siam J. Applied Math, Vol. 32, No. 4, June 1977.

[HKT] Hodson, Kendall, and Tautu. Mathematics in the Archaeological and Historical Sciences. University Press, Edinburgh, 1970.

[KMB] L. Kou, G. Markowsky, and L. Berman, "A Fast Algorithm for Steiner Trees", Acta Informatica 15, 1981.

[NA] D. Najok, "Principles and Modifications of Local Genealogical Algorithms in Textual History", Computers and the Humanities 14 (1980).

[NI] S. C. Nita, "Establishing the Linkage of Different Variants of a Romanian Chronicle" in [HKT].

[SU] G. Sullivan, "Approximation Algorithms for Steiner Tree Problems". Department of Computer Science, Yale University, Technical Report 249, October 1982.

[W] E. O. Wiley, Phylogenetics: The Theory and Practice of Phylogentic Systematics. Wiley-Interscience 1981.

[WA] J.D. Watson. The Molecular Biology of the Gene. Benjamin Press, 1975.

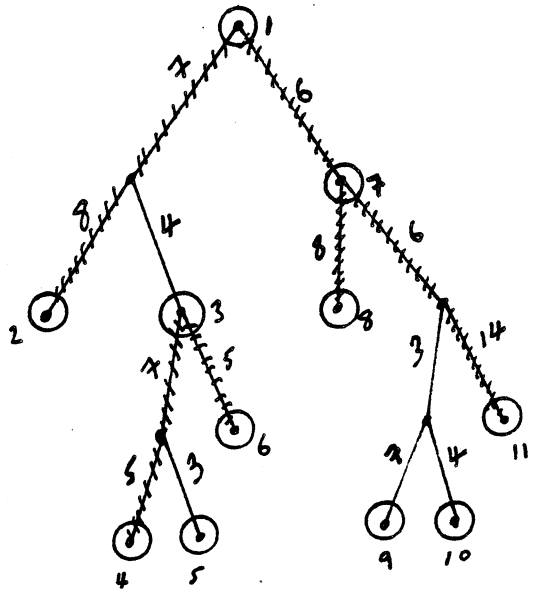

Figure 1: Optimal Steiner tree ST. The nodes in set X are circled. The node numbers give the order that the nodes are first visited by P. The dashed paths represent the paths in X(i) for i = {1,3,7}. W(ST) = 87, and Σw(i) = 66.

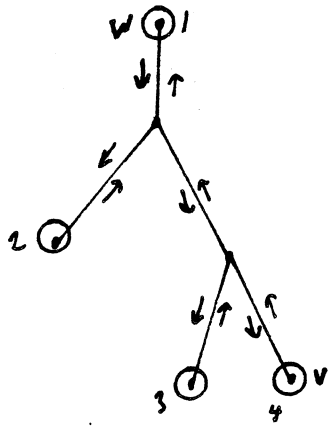

Figure 2: Optimal Steiner tree ST. The node numbers are given by P. The backword path from v (4) to w (1).