CIS-TR-80-5

K-trees: representations and distances

Andrzej Proskurowski
Department of Computer and Information Science
University of Oregon, Eugene, Oregon 97403

Abstract

K-trees can be defined by an iterative construction process where in every step a "new" vertex is added adjacent to some k mutually adjacent "old" vertices. Such a process may be represented by the recursive labeling in which the consecutive integers are associated with the vertices of a k-tree as they are added in the construction. Other, more visually appealing structures representing a particular construction process are the marked tree and the spatial tree. Nodes of these trees correspond to (k+1)-cliques of the k-tree. Using the notion of a cable of k vertex-disjoint paths between two vertices of a k-tree, we define the k-cable distance. We discuss the values of the shortest-path and the k-cable distances between vertices of a k-tree and present algorithms to compute them in terms of the different representations of k-trees.

1. Introduction

K-trees form a relatively restricted class of graphs which has received recently a substantial amount of interest generated in different research areas: complexity [1], centrality [6], reliability of networks [3]. K-trees can be defined by an iterative construction process where in every step a "new" vertex is added adjacent to some k mutually adjacent "old" vertices. Such a process may be represented by the recursive labeling in which the consecutive integers are associated with the vertices of a k-tree as they are added in the construction. Other, more visually appealing structures representing a particular construction process are the marked tree and the spatial tree. Nodes of these trees correspond to (k+1)-cliques of the k-tree. We will discuss these representations of k-trees and their suitability for combinatorial algorithms. In particular, we will present algorithms to compute the values of two distance functions: the shortest-path and k-cable distance, defined in [7].

2. Definitions

A graph with $n \geq k$ vertices is a <u>k-tree</u> iff it either (i) is a k-complete graph K_k, or (ii) can be obtained from a k-tree Q with n-1 vertices by adding a vertex adjacent to some k mutually adjacent vertices of Q. This recursive definition establishes a k-tree as a representative of all possible <u>iterative constructions</u> which can be expressed by so called <u>recursive labelings</u> of the k-tree. In such a recursive labeling, the "base" K_k subgraph of the k-tree has vertices labeled $1,\ldots,k$ and the label m, $k < m \leq n$, is assigned to a vertex adjacent to k vertices labeled $i_1,\ldots,i_k < m$.

A <u>recursive representation</u> of a k-tree Q associates with every label m labels $i_1,\ldots,i_k$, as above, according to a particular recursive labeling of vertices of Q. This generalized "father array" [4] is a partial adjacency list of vertices of Q. The remainder of the adjacency information is contained in the labeling itself.

The "tree-like" structure of a k-tree is immediately apparent when we represent the k-tree by a <u>marked tree</u> [2]. In this representation, we drop irrelevant implications of the recursive labeling (the total ordering of vertices) but retain the essential dependencies. The nodes of a marked tree represent (k+1)-cliques (K_{k+1} subgraphs) of a k-tree, and the edges represent K_k subgraphs. The nodes are marked by k+1 different "marks" (say, numbers 1,...,k+1). Each node corresponds to a non-basic vertex of Q and thus we may talk about marking of vertices of Q. The vertices of the base in the particular iterative construction of Q are marked 1,...,k. The corresponding edge of T becomes the "root edge". Each vertex added in the construction process is marked by the element missing among the marks of the parental K_k and the correspondingly marked node is made incident with the parental edge in T. The parent node in T represents the "youngest" - in terms of the iterative construction - parental vertex in Q. If we allow more than one edge to represent a given (interior [7]) K_k, this defines the parent-child relation between nodes of the tree.(*)

Another visualization of the recursive representation of k-trees is the spatial tree. Its nodes lie on the points of the (k+1)-dimentional grid and represent cliques of the k-tree Q. Addition of a vertex marked i in a given iterative construction of Q, is reflected in the spatial tree by a new node displaced in the direction of the i-th coordinate from its parent node ("mark" and "parent" are used in the sense of the marked tree). All the vertices adjacent to the base K_k are represented as separate nodes at the origin and are incident to edges along the (k+1)st coordinate representing the base of Q. Different nodes may be placed in the same grid point without any implication as to the relation between them. It is the way they "got to" that point (the sequence of edges from a common ancestor) that determines this relation (e.g., the distance between the vertices).

(*) Phyllis Chin suggested that "syblings through the same K_k" should be represented as incident to the same hyperedge. It follows from the definition of k-trees, that the hypergraph defined above is cycle-less.

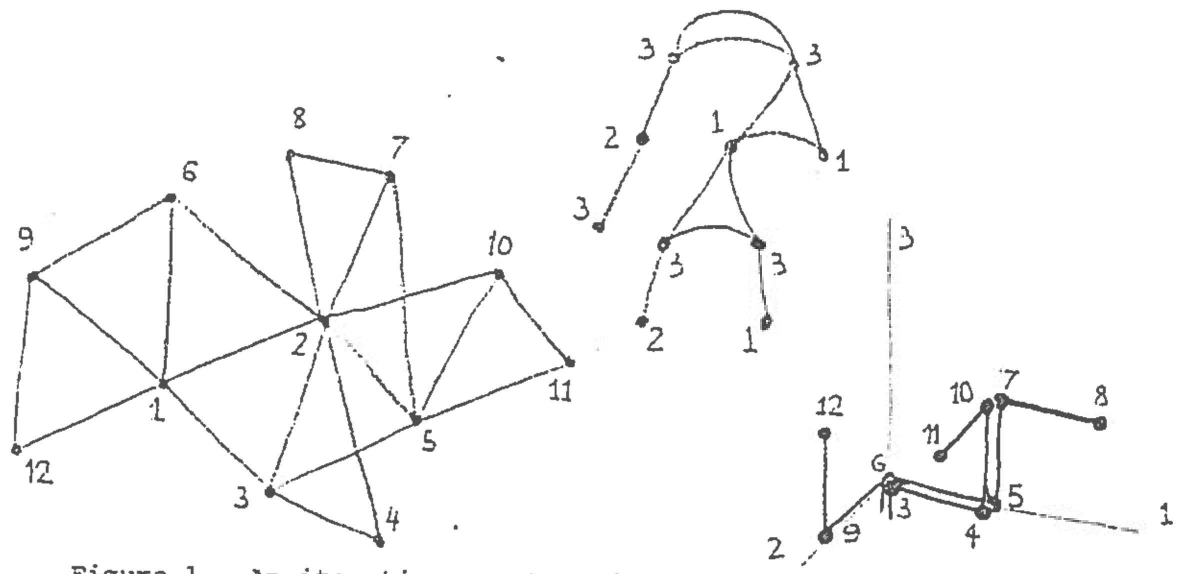

Figure 1. An iterative construction of a 2-tree indicated by its labeling, and the corresponding marked and spatial trees.

The view of k-trees as basic acyclic graphs redundantly stocked with edges for the reason of greater connectivity invites discussion of distances in these graphs. The usual shortest-path distance may not suit the needs of k-connectivity. In [7] we have introduced the notion of a k-cable, which is a collection of k vertex-disjoint paths between two vertices. The length of a cable is equal to the length of its shortest path. The k-cable distance between two vertices of a k-tree is the length of a shortest k-cable between them. We notice that the cable distance can differ from the shortest-path distance between the same vertices by an arbitrarily large amount.

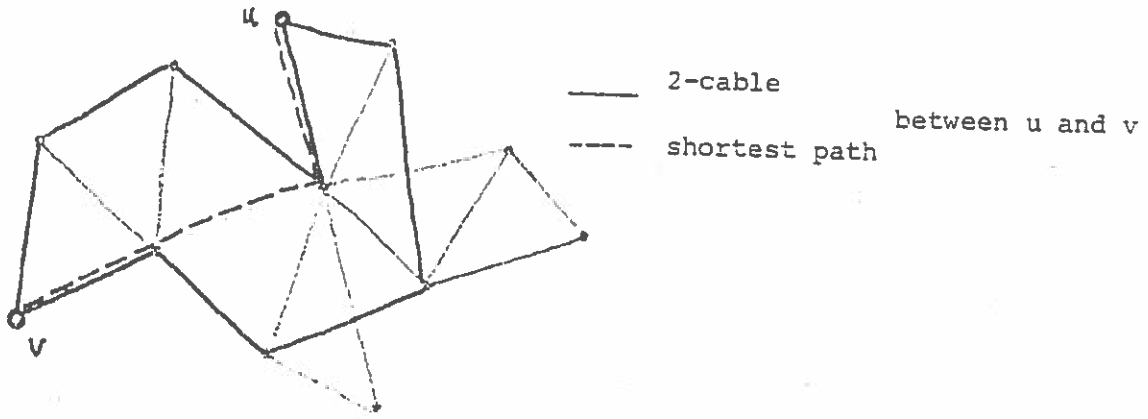

Figure 2. A 2-cable in a 2-tree and the cable- vs. path-distance.

3. Computing the shortest path distance

For the purpose of this presentation, we define the distance from a non-basic vertex to the base, i.e., to a closest vertex of the base. To discuss the distance between two arbitrary vertices of a k-tree, we can use a solution to the problem above with the base strategically placed. This requires a fast "rerooting" algorithm changing the representation according to the redefined base subgraph [5].

Given an iterative construction of a k-tree Q and a non-basic vertex v of Q, the shortest path from v to the base of Q is given by the "shortest-path tree". In this tree, nodes represent non-basic vertices of Q and their adjacency is determined by the adjacency of a "new" vertex v and the "oldest" vertex in the parental K_k of v. This oldest vertex has the least label among the vertices corresponding to v in the recursive representation of Q. Hence, the recursive representation of the shortest-path tree is given by the array of the least entries in the recursive representation of the k-tree. In this array, the labels of the base vertices all denote the base and could thus be replaced by the same symbol, say k.

A shortest path from v to the base can thus be recovered from recursive representation of Q in time proportional to the length of the path.

Algorithm 1.

```
Input:   Recursive representation R of Q and the label of v.
Output:  Labels of vertices of a shortest path from v to the
         base of Q.
Method:  {Let p be the label of the current vertex on the path.}
         Set p to the label of v;
         While $p > k$ do
           output p; set p to the least entry in R[p].
```

For a marked tree (or its spatial embedding) representation of a k-tree, the oldest parent of a vertex v represented by a node p is represented by the "youngest" ancestor q of p such that the set of marks of the nodes on the path q,...,p completes the mark set $M=\{1,...,k+1\}$. If no such q exists, the oldest parent of v is obviously among the base vertices. Let us define a hyperplane x_i=const to cover a path in the grid iff it contains all edges of the path. The following proposition follows immediately from the discussion above.

Proposition 1. Given a spatial tree S representing a k-tree Q and a non-basic vertex v, the shortest-path distance from v to the base of Q is equal to the minimum number of hyperplanes covering the path from the node representing v to the origin, and the edge representing the base.

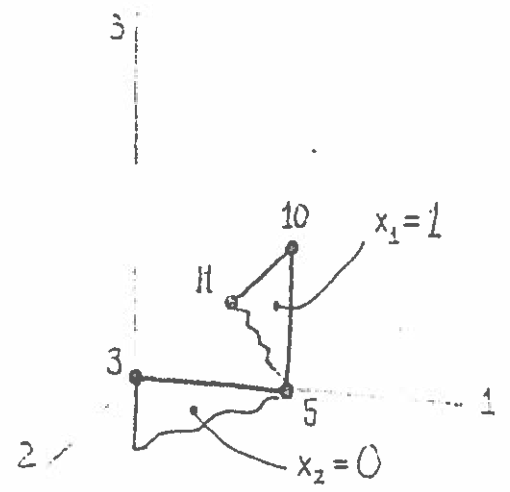

Figure 3. A set of hyperplanes representing a shortest path in the 2-tree in Figure 1.

4. Computing the cable distance

A cable between two vertices u and v of a k-tree Q has been defined as a collection of k vertex-disjoint paths between these vertices. In the following we will discuss computation of the k-cable distance between a non-basic vertex of Q and the base. This restriction, similar to the restriction of the preceding section, requires rerooting of Q in the case of arbitrary u and v, so that one of the vertices is adjacent to all vertices of the new base and is separated by these from the other vertex.

We have shown in [7] that a shortest cable between u and v is spanned on vertices of the k-complete subgraphs of Q separating u and v. Due to this observation, it is easy to recover the vertices of the cable between a given vertex and the base from a recursive representation of Q. The set of parental vertices of a given vertex v is an entry in this representation and induces a subgraph separating v from some vertices of the base.

The following algorithm computes the k-cable distance to the base of a k-tree.

Algorithm 2.

```
Input:   Recursive representation R of k-trees Q and the vertex
         v.
Output:  Labels of a shortest path of a k-cable from v to the
         base of Q
Method:  {Let p be the label of the current vertex}
         {Let C be an array of paths' labels}
         Set p to the label of v:
         Assign labels of R[p] to the k paths in C;
         While no base vertex has been reached do
           set p to the next highest "assigned" label;
           assign the only unassigned label in R[p]
                             to the path ending with p.
```

Using the same approach, an algorithm linear in the number of nodes can determine both the (k-1)-tree of subgraphs separating a vertex from the base and the corresponding shortest cable. A computer program (in Pascal) implementing this task is given in the Appendix.

The determination of a shortest cable using the marked or spatial tree representation can be done in the similar manner. Since it is necessary only to consider the nodes on the path to the root in these representations, the corresponding algorithms may be faster.

Proposition 2. Given a marked tree T representing a k-tree Q and k partially determined vertex-disjoint paths from a vertex v to the base of Q. Let E be the set of end-vertices of these paths and let $u \varepsilon E$ be the youngest vertex represented by a node p in T. Then the corresponding path can be extended from u by the edge (u,w), where w is represented by the youngest ancestor q of p whose mark completes M. If no such node exists then w is a base vertex.

Proof. We will show that the marks of end-vertices of the k vertex-disjoint paths are all different. The proof is conducted by induction on the total length of the paths. (i) The parent vertices of v have all different marks, by definition. (ii) Let u be a vertex as postulated above, represented by p. Then the youngest ancestor q of p completing the set of marks M represents the only parental vertex w of u which is available to extend the path from u toward the base. The node q replaces p in E and its mark differs from marks of all other elements of E. □

This proposition implies correctness of the following algorithm.

Algorithm 3.

```
Input:   Marked tree representation of a k-tree Q and a vertex v.
Output:  Nodes of a shortest cable from v to the base of Q.
Method:  {Let p be the current node}
         {Let C be an array of paths}
         {Let E be an array of end-nodes}
         Set p to the label of v;
         Enter nodes representing parental
                    vertices of v into C and E;
         While no base vertex has been reached do
            set p to its parent node;
            extend the path to p in C by p's youngest
                    ancestor q whose mark completes M;
            substitute q for p in E.
```

For spatial tree, the completeness of the set of marks is equivalent to covering of the path to the base by translates of m-dimensional subspaces ($m \leq k$) of the grid space. The following proposition follows from the observation about marks of nodes representing end-vertices of the cable paths.

Proposition 3. Given a spatial tree S representing a k-tree Q and k partially determined vertex-disjoint paths from a vertex v to the base of Q. Let u be the oldest end-vertex of these paths (other than v) and let $\hat{p}$ represent it in S. Define H as the translate of the smallest dimension subspace containing p and its ancestor node not yet assigned to any path. The youngest unassigned ancestor of p in H represents a vertex w which extends the path from u in a shortest cable from v to the base of Q.

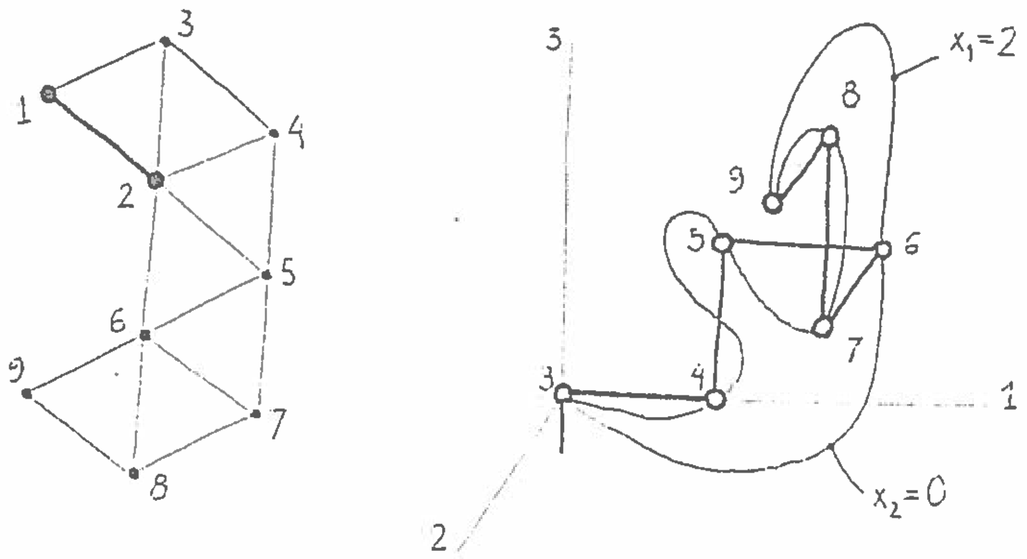

Figure 4. Spatial representation of a 2-tree and its cover defining a shortest cable.

5. Conclusions

We have introduced the spatial tree as a visualization of recursive representation of k-trees. It gives a different geometric interpretation of the shortest path and the shortest cable. It may be a convenient way of viewing k-trees. However, for its usefulness, the question of the effect of rerooting of the k-tree on its spatial tree representation should be answered. Some other problems for k-trees may have interesting interpretation in this representation. For instance, [7], what is the representation of the (k-1)-tree induced by vertices separating two given vertices of a k-tree? The spatial tree defines euclidean distance from the base to the nodes if we "grow" the nodes in positive directions of the coordinates. This induces distance-related questions like centrality.

6. References

[1] K.S. Booth, C.J. Colbourn, Problems polynomially equivalent to graph isomorphism, University of Waterloo, CS-77-04, June 1979.

[2] D. Corneil, M. Klawe, A. Proskurowski, Complexity of the isomorphism problem for "hook up" classes of graphs, in preparation.

[3] A. Farley, Independently reliable networks, University of Oregon CS-TR-79/21, submitted.

[4] D.E. Knuth, The Art of Computer Programming, vol. I, 2nd ed., Addison-Wesley, 1973.

[5] A. Proskurowski, Shortest paths in recursive graphs, University of Oregon, CS-TR-78/10, submitted.

[6] A. Proskurowski, Centers of k-trees, University of Oregon, CS-TR-79/9, submitted.

[7] A. Proskurowski, Separating subgraphs in k-trees, University of Oregon, CS-TR-79/18, submitted.

7. Appendix

```
{A procedure to compute the cable distance between vertex n and the base}
var R:  array [1..k,k+1..n] of integer; {recursive representation}
    S:  array [1..k-1,k..n] of integer; {separating (k-1)-tree}
    A:  array [k+1..n] of integer;      {path indicators for assigned
    E:  array [1..k] of integer;        {pointers to end nodes}  nodes}
    C:  array [1..1,1..n] of integer;   {paths of the cable}
    i,j,b,p:  integer;
begin for i:=k+1 to n do A[i]:=0;
      for j:=1 to k do
      begin A[R[j,n]]:=j; A[j]:=0; E[j]:=1 end;
      for i:=n-1 downto k+1 do
         if A[i]<>0 then
         begin b:=0;
            for j:=0 to k do
            begin p:=R[j,i];
               if A[p]<>0 then S[j-b,i]:=p
                          else begin b:=1; A[p]:=A[i];
                                     C[A[i],E[A[i]]]:=p;
                                     E[R[i]]:=E[A[i]]+1
                                     end
               end
            end
    end.
```