Edge Coloring A K-Tree Into Two Smaller Trees

Dilip Chhajed
Department of Business Administration
University of Illinois
Edge Coloring A K-Tree Into Two Smaller Trees

Dilip Chhajed
Department of Business Administration
Edge Coloring A K-Tree Into Two Smaller Trees

Dilip Chhajed
Department of Business Administration
University of Illinois
350 Commerce West
Champaign, IL 61820
email: chhajed@vmd.cso.uiuc.edu

September 1993

Abstract
The problem of edge coloring partial k-tree into two partial p- and q-trees with p,q<k is considered. An algorithm is provided to construct such a coloring with p+q=k. Usefulness of this result in a Lagrangian decomposition framework to solve certain combinatorial optimization problems is discussed.

Keywords: Networks/Graphs: Theory; Facilities Planning: Location; Analysis of Algorithms: Suboptimal Algorithms

Acknowledgments: I would like to thank Refael Hassin and Charles Blair for their comments on an earlier draft of this paper.
We consider a special case of the following edge coloring problem in graphs (Heath, 1991): Given an graph \( G(V, E) \) and a class of graphs \( Q \), color the edges of \( G \) using \( u \) colors so that each monochromatic graph is isomorphic to some graph in \( Q \). Other names used for this problem are partitioning the edges of \( G \) into graphs of \( Q \) or covering edges of \( G \) with graphs in \( Q \) (Elmallah and Colbourn, 1988). In this paper we consider the case when \( G \) is a partial k-tree, \( Q \) consists of partial \( k' \)-trees with \( k' < k \) and \( u \) is 2.

It is well known that when \( G \) is Halin graph, its edges can be assigned two colors to form a monochromatic tree and a monochromatic cycle. Elmallah and Colbourn (1988) provide the solution to the problem of \( G \) as a planar graph and \( Q \) consisting of partial 3-trees. Heath (1991) improves this result to show that a planar graph can be partitioned into two outerplanar graphs.

Adapting the definition of Elmallah and Colbourn (1988), we say that \( G \) is a partial \((p,q)\)-tree if the edges of \( G \) can be assigned two colors so that, one monochromatic colored subgraph is a partial \( p \)-tree and the other is a partial \( q \)-tree. In this paper we show that a partial \( k \)-tree is a partial \((p,q)\)-tree with \( p+q=k \) and \( p,q>0 \).

Our motivation for this problem came from a desire to exploit existing polynomial time algorithms for several problems on partial \( k \)-trees (for small \( k \)) to solve larger and more general problems. We discuss this further in section 3. To this end, we wanted to know the class of graphs that can be edge colored by two partial 2-trees. This question is still unsettled although we know that planar graphs (Heath, 1991), partial 3-trees and partial 4-trees (results of this paper) are members of this class.

The outline of this paper is as follows: in section 1 we introduce some definitions and known properties of partial \( k \)-trees. Section 2 contains the algorithm and a proof of its correctness. A discussion of these results is provided in section 3. Finally we end with a conclusion.
1. Preliminaries

We first introduce some definitions. $G' (V', E')$ is a subgraph of $G$ if $V \supseteq V'$ and $E \supseteq E'$. $G'$ is a partial graph if $V = V'$ and $E \supseteq E'$. A $k$-clique is a complete graph on $k$ vertices ($k \geq 1$ and integer). A $k$-tree is a generalization of trees and is recursively defined as follows. A $k$-clique is a $k$-tree. Given a $k$-tree and a subgraph of the $k$-tree that is a $k$-clique, the graph obtained by introducing a new vertex and connecting it to every vertex of the $k$-clique is again a $k$-tree. A partial graph of a $k$-tree is a called a partial $k$-tree.

A node with degree $k$ is a $k$-leaf if all of the $k$ nodes adjacent to it (neighbors) induce a $k$-clique. A $k$-leaf along with its neighbors forms a $(k+1)$-clique. If we eliminate a $k$-leaf of a $k$-tree, the resulting graph is again a $k$-tree. Thus, repeated elimination of $k$-leaves will result finally in a graph which is a $k$-clique. Every $k$-tree with greater than $k+1$ nodes has at least two non-adjacent $k$-leaves any one of which can start the elimination process (Rose, 1974). For a $k$-tree with $n > k$ vertices, let the set $Y = [y_n, \ldots, y_{n-k}]$ denote the ordering of vertices in an elimination sequence of $k$-leaves and let the $k$ remaining nodes of the $k$-tree be arbitrarily numbered $y_1, \ldots, y_k$. Thus $y_n$ represents the node which is eliminated first, followed by $y_{n-1}$, and so on. Such an elimination sequence can be found by a simple scheme that keeps track of the degree of every node.

Partial $k$-trees form a rich class of graphs and contain several well known graph families. A tree, defined in the usual sense, is a 1-tree. Series-parallel graphs, circuits, outer-planar graphs, and cactus graphs are all partial 2-trees. A Halin graph is a partial 3-tree. Many combinatorial problems have been solved when restricted to partial $k$-trees (Takamizawa, Nishizeki and Saito, 1982; Corneil and Keil, 1987; Fernandez-Baca, 1989; Arnborg and Proskurowski, 1989; Chhajed and Lowe, 1993).

Although efficient algorithms exist for several problems when restricted to partial $k$-trees, recognizing whether a graph is a partial $k$-tree remains a difficult problem. Arnborg, Corneil, and Proskurowski (1987) have show that finding an embedding of a partial $k$-tree into a $k$-tree with the same node set is NP-hard for arbitrary $k$ but can be
solved for a fixed $k$ in $O(n^{k+2})$. Lagergren (1990) has given, for a fixed $k$, an $O(\log^3 n)$ parallel algorithm for finding the tree-decomposition of a partial $k$-tree, which can be implemented in $O(n\log^2 n)$ time in a sequential manner.

In the remaining discussion we assume that $G$ has been embedded into a $k$-tree and thus we assume without loss of generality that $G$ is a $k$-tree. Renumber the nodes of $G$ so that $[y_n,...,y_{n-k}]$ is an elimination sequence and the final $k$-clique obtained consists of nodes $\{y_1,...,y_k\}$. Let $D_t$ be the clique to which node $y_t$ is attached when it was removed. Our algorithm colors the edges of $G$ by reconstructing $G$.

2. The Algorithm

Let positive integers $p$ and $q$ be such that $p+q=k$. The algorithm now follows.

Algorithm 2ColorK-TREE

Given: Embedding of a graph on a $k$-tree along with positive integers $p$ and $q$ such that $p+q=k$. An elimination sequence with nodes renumbered so that $[y_n,...,y_{n-k}]$ is the elimination sequence. The algorithm rebuilds the $k$-tree by following the elimination sequence in reverse order.

Step 1: Start with a $k$-clique on nodes $\{y_1,...,y_k\}$. Color the edges of $p$-clique formed by nodes $\{y_1,...,y_p\}$ blue. Assign red color to the remaining edges.

Set $t:=k+1$.

Step 2: In the node set $D_t$, identify a blue $p$-clique and a red $q$-clique with nodes $B$ and $R$, respectively such that $R\cup B=D_t$. Add node $y_t$ and new edges connecting $y_t$ to each node in $D_t$. Color the new edges connecting node $y_t$ to nodes in $R$ with red color and the edges with end nodes $y_t$ and nodes in $B$ with blue color.

Step 3: If $t=n$ then stop, else set $t:=t+1$ and go to step 2.

We now proceed to show the correctness of the algorithm. A few more definitions
are needed. A subgraph (clique) consisting of red edges will be called a red subgraph (red clique). A k-clique with edges colored either red or blue has the \textit{2color-subcliques property} if its nodes can be partitioned into two sets \{R,B\} such that the subgraph induced by R is a red q-clique and the subgraph induced by B is a blue p-clique.

\textbf{Lemma 1:} Given a k-clique with nodes \(\{y_1,\ldots,y_k\}\), edges corresponding to the clique on nodes \(\{y_1,\ldots,y_p\}\) colored blue and the remaining edges assigned red color. Then the k-clique has \textit{2color-subcliques} property. Moreover, the blue subgraph is a p-tree and the red subgraph is a q-tree.

\textbf{Proof:} The blue colored subgraph is a p-clique and so is a p-tree. The red subgraph has all the k nodes and contains a q-clique with nodes \(\{y_{p+1},\ldots,y_k\}\). In addition, this graph has edges between node \(y_i\) and every node in \(\{y_{p+1},\ldots,y_k\}\) for \(i=\{1,\ldots,m\}\). Thus the red subgraph is a q-tree. «»

Consider a k-clique with node set C with \textit{2color-subcliques} property. Let \(C^o\) be the graph obtained by adding a new node \(y^o\) and connecting it to every node in C with edges \((y^o,y), y \in B\) colored blue and red color assigned to edges \((y^o,y), y \in R\).

\textbf{Lemma 2:} Given graph \(C^o\) as defined above, then every subgraph induced by a subset of k nodes of \(C \cup \{y^o\}\) is a k-clique with \textit{2color-subcliques} property.

\textbf{Proof:} Since C is a k-clique and node \(y^o\) is connected to every node in C, the graph \(C^o\) is a \((k+1)\)-clique. Thus every graph induced by a subset of nodes of this is a clique. There are \((k+1)\) distinct subsets of k nodes of \(C^o\). One of these subsets is C which satisfies the lemma. Each of the other k subsets will contain the node \(y^o\), and

a) \((q-1)\) nodes from R and \(p\) nodes from B or

b) \(q\) nodes from R and \((p-1)\) nodes from B.

We only consider case (a) as the proof for (b) is similar. Consider a subset of k nodes
satisfying (a). The p nodes of B form a blue clique. y° is connected to every node of R with red edges, thus (q-1) nodes of R with y° form a red q-clique. This shows that the 2color-subcliques property is satisfied by every k-clique of C°. «»

**Lemma 3:** The clique D_t in Step 2 of Algorithm 2ColorK-TREE has the 2color-subcliques property.

**Proof:** We prove this by induction on the number of nodes of G. Step 2 is first encountered with t=k+1. D_{k+1} is the set of nodes \{y_1,...,y_k\}, and by Lemma 1, it satisfies the property. Let G_t be the graph obtained at the end of step 2 of the algorithm, i.e., after adding node y_t along with the arcs. By Lemma 2, every clique of G_{k+1} with k-nodes also satisfies the property.

We now assume the Lemma for (n-1) nodes. Consider D_n and let \tau be the largest of indices of nodes in D_n. From the graph G_{n-1}, if we delete all the nodes with indices greater than \tau then the resulting graph is the same as G_{\tau}. In the graph G_{\tau} also node y_{\tau} is adjacent to nodes in D_n\{y_{\tau}\} and thus when y_{\tau} is added, it is connected to D_n\{y_{\tau}\} and one additional node. This shows that D_{\tau} \supseteq D_n\{y_{\tau}\}. By induction hypothesis and Lemma 2, every k-clique of D_{\tau}\cup\{y_{\tau}\} has the 2color-subcliques property and thus clique D_n has the 2color-subcliques property.

**Theorem:** Algorithm 2ColorK-TREE correctly assigns 2 colors to the edges of a given k-tree to give a blue p-tree and a red q-tree.

**Proof:** When there are only k-nodes, the theorem follows from Lemma 1. So consider the case when n>k. When node y_{k+1} is added, D_{k+1} satisfies the 2color-subcliques property and by the definition of a k-tree, the red and blue edges continue to form p-tree and q-tree, respectively. By induction suppose that the theorem is true for n-1 nodes. When the n^{th} node is added, by induction, the blue and red graphs are each p-tree and q-tree, respectively and by the nature of addition of node n and the edges, the new blue (red)
subgraph formed with one additional node is again a p-tree (q-tree). «»

3. Discussion

k-trees have the k-clique separator property (Rose, 1974) and many algorithms have been designed to exploit this property in constructing polynomial time algorithms on k-trees (for fixed k). We believe that the edge coloring property of a k-tree can provide another vehicle to device polynomial time algorithms to solve combinatorial problems on a k-tree.

The other application of this result is in utilizing some of the existing algorithms in a Lagrangian decomposition framework (Guignard and Kim, 1987) to find bounds more efficiently. To motivate this consider the following Quadratic Location Problem (Chhajed and Lowe, 1992) that has applications in location theory, computer module allocation, product design, etc.

Problem $P$ \[ \min \sum_{(u,v) \in E} \sum_{k=1}^{w} \sum_{r=1}^{w} C_{ukvr}x_{uk}x_{vr} \]
Subject to:
\[ \sum_{k=1}^{w} x_{uk} = 1 \ \forall \ u \in V \]
\[ x_{uk} = \{0,1\} \ \forall \ k = 1, \ldots, w; \ u \in V. \]

If edges of $G$ can be partitioned into two graphs $G_1(V_1,E_1)$ and $G_2(V_2,E_2)$ with nodes $V_1 \cap V_2 = V_{12}$ common between them then we can rewrite the above as:

Problem $P'$ \[ \min \sum_{(u,v) \in E_1} \sum_{k} \sum_{r} C_{ukvr}x_{uk}^1x_{vr}^1 + \sum_{(u,v) \in E_2} \sum_{k} \sum_{r} C_{ukvr}x_{uk}^2x_{vr}^2 \]
Subject to:
\[ \sum_{k} x_{uk}^1 = 1 \ \forall \ u \in V_1 \] (1)
\[ \sum_{k} x_{uk}^2 = 1 \ \forall \ u \in V_2 \] (2)
\[ x_{uk}^1 = x_{uk}^2 \ \forall \ u \in V_{12}, k = 1, \ldots, w \] (3)
Here the \( x \) variables corresponding to the nodes in \( G_1 \) (\( G_2 \)) are denoted by \( x^1 \) (\( x^2 \)). Note that constraints (3) require that the corresponding \( x^1 \) and \( x^2 \) variables take on the same value.

A Lagrangian relaxation of \((P')\) is obtained by multiplying each of the constraints (3) by a multiplier \( \lambda_{uk} \) and adding them to the objective function. This gives us

**Problem \( P(\lambda) \)**

\[
Z^*(P(\lambda)) = \min \sum_{(u,v) \in E_1} \sum_k \sum_r C_{ukrv} x^1_{uk} x^1_{vr} + \sum_{u \in V_{12}} \sum_k x^1_{uk} \lambda_{uk}
\]

\[
+ \sum_{(u,v) \in E_2} \sum_k \sum_r C_{ukrv} x^2_{uk} x^2_{vr} - \sum_{u \in V_{12}} \sum_k x^2_{uk} \lambda_{uk}
\]

Subject to: (1), (2) and (4).

The best choice of multipliers is the one which maximizes \( Z^*(P(\lambda)) \). This is given by the following dual problem which is a lower bound to \( P \):

**Problem \( P_d \)**

\[
Z^*(P_d) = \max_{\lambda} Z^*(P(\lambda))
\]

When \( G \) is a \( k \)-tree, \( P \) (as well as the version with some linear terms in \( x \)) can be solved in \( O(w^{k+1}(k+1)n) \) time (Fernandez-Baca, 1989, Chhajed and Lowe, 1993). \( P(\lambda) \) can be separated into two subproblems and each of them represent a problem of the form of \( P \) (with some linear terms in \( x \)) but on graphs different from the original graph. In solving \( P(\lambda) \), a feasible solution can also be generated which provides an upper bound to the problem. In the decomposition, instead of solving one problem with \( k \)-tree, several iterations of \( P(\lambda) \) will be performed, each with new values of the multipliers and in every iteration two problems with \( p \)- and \( q \)-trees will be solved. This may however, result in savings in computational time in many cases. To illustrate this, we assume that \( k \) is even.
to simplify the notation in the following arguments and set $p=q=k/2$. Let the time taken for exact solution is $Z_1 = C_1[w^{k+1}(k+1)n]$ and the time taken by the decomposition approach to get a "good" lower bound with $N$ iterations is $Z_2 = 2NC_2[w^{k/2+1}(k/2+1)n]$.

The ratio $Z_1/Z_2$ provides a measure of the effectiveness of the decomposition approach with a value greater than 1 providing greater support to the decomposition approach. Assuming $C_1=C_2$ we get $\frac{Z_1}{Z_2} = \frac{w^{k/2}(k+1)}{N(k+2)}$. Thus, for problems that require low number of iterations, large value of $w$ and/or $k$, the decomposition can be an attractive alternative.

4. Conclusion

In this paper we have shown that a k-tree is a partial (p,q)-tree. This result is useful in solving problems on partial k-trees with large $k$ by decomposing it into problems with smaller $k$. This research also brings us a step closer toward characterizing the class of graphs that are partial (p,q)-trees.
References


