Efficient sets in partial k-trees

Jan Telle and Andrzej Proskurowski

CIS-TR-91-07
March 1991

Department of Computer and Information Science
University of Oregon

Efficient sets in partial k-trees

Jan Arne Telle
Andrzej Proskurowski
Department of Computer and Information Science
University of Oregon, Eugene, Oregon 97403, USA

Abstract

We generalize the result of Bernhard, Hedetniemi and Jacobs by providing a linear time algorithm that computes the efficiency number of a partial k-tree (given with its embedding in a k-tree).

1 Motivation

For a fixed value of the integer parameter k, partial k-trees are exactly subgraphs of those chordal graphs that have at most $k+1$ completely interconnected vertices (see, for instance, [3, 10, 12]). Thus, partial 1-trees are the acyclic graphs (forests), and partial 2-trees are the series-parallel graphs (graphs with no K_4 minors or homeomorphs).

The class of partial k-trees is identical to the class of graphs of tree-width k [11]. These graphs have been in the focus of attention in recent years because of their interesting algorithmic properties. Namely, for a large number of inherently difficult (on general graphs) discrete optimization problems, partial k-trees admit a linear time solution algorithm when the value of k is fixed and any partial k-tree is given with its embedding k-tree [4]. We exhibit this property by applying the standard for this approach Dynamic Programming methodology to the problem of determining the efficiency number of a graph. This parameter has been defined by Bernhard, Hedetniemi, and Jacobs [5] as the maximum number of vertices uniquely dominated by a subset of vertices in the graph. They proved the problem to be NP-complete even when restricted to bipartite graphs and provided a linear time algorithm computing the efficiency number of a tree. We generalize their result to partial k-trees. Because of the separation of series and parallel reductions, we will discuss our result restricted

to partial 2-trees (series-parallel graphs) first. We then use the introduced concepts to present the general result.

2 Definitions and terminology

We will use standard graph theory terminology, as found, for instance, in Bondy and Murty [7]. In addition, we define some basic concepts.

For a graph $G = (V, E)$ and a set $D \subseteq V$, we define the *efficiency* of D to be the number of vertices in $V \backslash D$ such that each vertex is adjacent to exactly one vertex in D; we say that such a vertex is *efficiently dominated*. We call the set D a *dominating* set, even though it does not necessarily dominate every vertex in $V \backslash D$. The efficiency of G, $\varepsilon[G]$, is the maximum efficiency of any subset of V. The efficiency of a disconnected graph is clearly the sum of the efficiencies of its components. For the purpose of exposition, we assign *states* to the vertices of G for a given $D \subseteq V$ as follows:

$$state(v) = \begin{cases} 0 & \text{if } v \text{ not dominated and } v \notin D \\ 1 & \text{if } v \text{ efficiently dominated} \\ 2 & \text{if } v \text{ dominated by more than one vertex and } v \notin D \\ 3 & \text{if } v \in D \end{cases}$$

We say a (partial) state assignment function $\Phi : V \rightarrow \{0, 1, 2, 3\}$ is *legal* if and only if it corresponds to an assignment of states to V for a particular $D \subseteq V$ according to the above.

A graph G is a k-tree if and only if there exists a *perfect elimination* ordering of its vertices, $peo = v_1, ..., v_n$, such that for every $1 \leq i \leq n - k$, the higher numbered neighbors of v_i form a k-clique K. We then call K the *base* of v_i, and v_i a *descendant* of K. In the $(k+1)$-clique induced by $\{v_i\} \bigcup K$, there are k other k-cliques, in addition to K. We call these the *faces* of v_i. Each of these faces may in turn be a base of other vertices, and at any time during the reduction process, the *reduced branches* of a k-clique K is the transitive closure (across faces) of descendants. The base of v_{n-k} is termed the *root* of the k-tree. A partial k-tree is obtained by removing edges from a k-tree; we call the latter an *embedding* k-tree.

3 Efficiency of partial 2-trees

In our algorithm, we will follow the structure of a given partial 2-tree as defined by a perfect elimination ordering of vertices determined by an embedding 2-tree. In doing so, we will combine optimal solutions to the efficiency subproblems restricted

to subgraphs of the original graph with given states of some vertices. We define an operation of collapsing two vertices of a given graph: $U(G; x_1, x_2)$ will denote the graph resulting from identifying the collapsing vertices x_1 and x_2 in the graph G. We will refer to the collapsed vertex as x. Furthermore, given a partial state assignment *state* to a set of vertices S of a graph G, we define $\varepsilon[G; state(S) = \vec{s}]$ to be the efficiency of G when the state of any vertex x in S is restricted to $state(x) = s_x$, *without* the possible contribution of any vertices in S as efficiently dominated. To justify our notation, $state(S) = \vec{s}$, we should actually view S as a sequence.

We initially view the 2-tree as a disjoint collection of initialized 2-cliques (edges) and combine solutions by collapsing corresponding vertices. Our intermediate goal is to establish the value of $\varepsilon[G', state(x) = s]$ for the graph $G' = U(G; x_1, x_2)$ given the efficiency $\varepsilon[G, state(x_1) = s_1, state(x_2) = s_2]$ for a partial state assignment $state(x_1) = s_1$ and $state(x_2) = s_2$. We will relate the state of the collapsed vertex x to the states of the collapsing vertices x_1, x_2 through the function τ defined below:

$$state(x) = \tau(s_1, s_2) = \begin{cases} 0 & \text{if } (s_1, s_2) = (0,0) \\ 1 & \text{if } (s_1, s_2) \in \{(0,1),(1,0)\} \\ 2 & \text{if } (s_1, s_2) \in \{(0,2),(2,0),(1,2),(2,1),(1,1),(2,2)\} \\ 3 & \text{if } (s_1, s_2) = (3,3) \\ \uparrow & \text{otherwise} \end{cases}$$

Before giving the algorithm for computing the efficiency of partial 2-trees we establish two theorems needed for correctness analysis.

Lemma 1: For two vertices x_1 and x_2 in G, not sharing a neighbor, and a legal partial state assignment $\vec{s}$ to a subset of vertices $S \subseteq V \backslash \{x_1, x_2\}$,

$$\varepsilon[U(G; x_1, x_2); state(x) = s, state(S) = \vec{s}] =$$

$$\max_{s_1, s_2} \{\varepsilon[G; state(x_1) = s_1, state(x_2) = s_2, state(S) = \vec{s}]\}$$

where $s = \tau(s_1, s_2)$.

Proof: Let $G' = U(G; x_1, x_2)$. Let any dominating set D in G' yield $state(x) = s$ and let v be a vertex of $G' \backslash \{x\}$ with d neighbors in D.

For the case $x \notin D$ (s=0,1 or 2), consider G with the same dominating set D. Any dominating set of G not including x_1 or x_2 will be considered as D varies in G'. Since G and G' only differ by edges with end vertex x, the vertex v has d neighbors in D also in the graph G. Hence v's contribution to the efficiency value is preserved. Recall that the efficiencies of the lemma do not count the contributions of x_1, x_2 or x. The states of vertices x_1 and x_2 vary depending on the number of neighbors they

have in D. Since x_1 and x_2 share no neighbors, it can be argued easily that the only legal states of these vertices are those given by the τ function. Thus the lemma holds for this case.

For the case $x \in D$ (s=3), consider G with the dominating set $D\backslash\{x\} \cup \{x_1, x_2\}$. Again the efficiency contribution of an arbitrary vertex $v \in G'\backslash\{x\}$ is preserved in G since x_1 and x_2 do not share neighbors. The state pair $s_1 = s_2 = 3$ is the only one considered by τ. Any dominating set of G that includes x_1 and x_2 will be considered as D varies in G'. Hence the lemma holds. □

A collapsed vertex contributes to the efficiency value of the graph if and only if its state is 1. We describe this contribution through the function σ: $\sigma(0) = \sigma(2) = \sigma(3) = 0$ and $\sigma(1) = 1$.

Theorem 1: For v_1 and v_2 not sharing a neighbor in G, and the state of some other two vertices of G, x and y, restricted to s_x and s_y, respectively,

$$\varepsilon[U(G; v_1, v_2); state(x) = s_x, state(y) = s_y] =$$

$$\max_{state(v)=\tau(s_1,s_2)} \{\varepsilon[G; state(v_1) = s_1, state(v_2) = s_2, state(x) = s_x, state(y) = s_y)] + \sigma(state(v))\}$$

Proof: Follows from Lemma 1 and the fact that when $state(v) = 1$ we must account for its contribution to the efficiency value. □

Let us extend the operation U to several sets of collapsed vertices, so that collapsing vertex x_1 with vertex x_2 and vertex y_1 with vertex y_2 is denoted $U(G; x_1, x_2; y_1, y_2)$.

Theorem 2: For x_1,x_2 and y_1,y_2, two disjoint pairs of vertices in G, such that vertices of neither pair share neighbors, and $(x_2, y_2) \notin E(G)$

$$\varepsilon[U(G; x_1, x_2; y_1, y_2); state(x) = s_x, state(y) = s_y] =$$

$$\max_{s_{x_1}, s_{x_2}, s_{y_1}, s_{y_2}} \{\varepsilon[G; state(x_1) = s_{x_1}, state(x_2) = s_{x_2}, state(y_1) = s_{y_1}, state(y_2) = s_{y_2}]\}$$

where $s_x = \tau(s_{x_1}, s_{x_2})$ and $s_y = \tau(s_{y_1}, s_{y_2})$.

Proof: By reversing the process of collapsing the two pairs of vertices, we can make use of Lemma 1. Let $G' = U(G; x_1, x_2)$. Note that y_1 and y_2 do not share a neighbor in G' since they do not share neighbors in G and $(x_2, y_2) \notin E(G)$. Hence, by Lemma 1,

$$\varepsilon[U(G'; y_1, y_2), state(y) = s_y] = \max_{s_{y_1}, s_{y_2}} \{\varepsilon[G'; state(y_1) = s_{y_1}, state(y_2) = s_{y_2}]\}$$

By the definition of G' and applying Lemma 1 again, the last expression is equal to

$$\max_{s_{y_1},s_{y_2}} \{ \max_{s_{x_1},s_{x_2}} \{\epsilon[G; state(x_1) = s_{x_1}, state(x_2) = s_{x_2}, state(y_1) = s_{y_1}, state(y_2) = s_{y_2}]\}\}$$

We thus get the desired result by the associativity of the maximum function.□

In the following algorithm, we use the above theorems. The right-hand side of the formulas in the theorems are computed using the fact that the efficiency of a disconnected graph is the sum of the efficiencies of its components.

Algorithm 1: Efficiency of partial 2-trees
Input: $G = (V, E)$, a partial 2-tree, given with an embedding 2-tree $G' = (V, E \cup E')$ and a corresponding vertex elimination ordering $\omega(V)$. Note that $E \cap E' = \emptyset$.
Output: $\epsilon[G]$

1. Initialize efficiencies of all edges $(x_i, x_j) \in E \cup E'$ as follows: If (s_i, s_j) is a legal state assignment (for the edge viewed as a disjoint component), then $\epsilon[(x_i, x_j), state(x_i) = s_i, state(x_j) = s_j] := 0$ and otherwise $\epsilon[(x_i, x_j), state(x_i) = s_i, state(x_j) = s_j] := -\infty$. The legality of a state assignment depends on whether $(x_i, x_j) \in E$ or $(x_i, x_j) \in E'$.

2. For $i = 1$ to $|V| - 2$ do *Reduction step:*

   Let v be such that $\omega(v) = i$. In G', let v have faces (v, a) and (v, b), so that (a, b) is its base. Let the disjoint representatives of these edges be the 2-cliques $(v_1, a_1), (v_2, b_2)$ and (a_3, b_3), respectively. Each of these may represent some previously reduced branches.

   First, compute the efficiency of the graph resulting from collapsing v_1 (in the subgraph reduced onto the face (v_1, a_1)) and v_2 (in the subgraph reduced onto the face (v_2, b_2)). This corresponds to a series reduction, and the result follows from the formula given in Theorem 1.

   Then, use this result to update the efficiency values of the base (a_3, b_3) as the efficiency of the graph resulting from collapsing the pair a_1, a_3 and the pair b_2, b_3. This corresponds to a parallel reduction, and the result follows from the formula given in Theorem 2.

   The total effect is the inclusion of v into the reduced branches while updating the optimal efficiency values of its base according to the optimal efficiency values of its faces.

3. Let (r, s) be the root of G'. Return the maximum efficiency value associated with this edge, while accounting for vertices r and s if they are efficiently dominated.

Timing and correctness

The algorithm follows the standard approach for solving discrete optimization problems on partial k-trees [4]. Note that an illegal partial state assignment will always have the efficiency value $-\infty$ by virtue of the initialization step and the rules for combining efficiencies of components. Correctness thus follows from the theorems. Finding an embedding 2-tree and a vertex elimination ordering can be done in linear time [12] and each iteration of the *Reduction* step involves a constant amount of work. Hence, the linear time complexity of the algorithm.

4 Efficiency of partial k-trees

For general k, the algorithm computing the efficiency of a partial k-tree again follows a perfect elimination ordering of its vertices as determined by an embedding k-tree. As before, we initially view the partial k-tree as a disjoint collection of initialized k-cliques, and combine solutions with the aid of a theorem that involves collapsing sets of corresponding vertices. This theorem updates the efficiency of the base of a vertex in terms of the efficiencies of the separate faces of the vertex, thereby including it in the reduced branches of its base. We call this process *pruning* the vertex. The theorem, for $k > 2$, is complicated by the fact that, in the embedding k-tree, the faces of a vertex share edges, which may or may not be edges of the partial k-tree (in 2-trees, they only share the pruned vertex). Pruning involves collapsing each of $k+1$ sets of corresponding vertices. To facilitate presentation, we state a lemma dealing with collapsing one such set. Even when collapsing only one set, the function τ that gives the state of the collapsed vertex x depends not only on the states of the collapsing vertices $x_1, x_2, ..., x_k$, but also on the presence of *shared* dominators. A vertex is a shared dominator if it efficiently dominates 2 or more of the collapsing vertices. Such a shared dominator is critical when two or more of the collapsing vertices are efficiently dominated (have state 1) and all the rest have state 0. The state of the collapsed vertex x will then depend on whether all the efficiently dominated collapsing vertices share the same dominator (in which case x is efficiently dominated, $state(x) = 1$) or not (which would imply $state(x) = 2$). Formally, let $D(v)$ be the dominator of an efficiently dominated vertex v. Let N be a set of vertices, with fixed states, that includes all shared dominators of $x_1, ..., x_k$, the vertices to be collapsed. Let

$state(N) = \vec{n}$, and $state(x_i) = s_i$. We define $state(x) = \tau(s_1, ..., s_k, N, \vec{n})$ to be:

$$\tau(s_1, ..., s_k, N, \vec{n}) = \begin{cases} 0 & \text{if } \forall i(s_i = 0) \\ 1 & \text{if } \exists i \forall j(s_i = 1 \wedge (s_j = 0 \vee (s_j = 1 \wedge D(x_j) = D(x_i)))) \\ 2 & \text{if } \forall i(s_i \neq 3) \wedge (\exists j(s_j = 2) \vee (\exists l, m(s_l = s_m = 1 \wedge D(x_l) \neq D(x_m)))) \\ 3 & \text{if } \forall i(s_i = 3) \\ \uparrow & \text{otherwise} \end{cases}$$

Note that τ requires information about shared dominators only when 2 or more of the collapsing vertices are efficiently dominated and the rest have state 0.

Lemma 2: Consider a graph G, a set of non-adjacent vertices $x_1, x_2, ..., x_k$ of G, and a subset of vertices $N \subseteq V - \{x_1, ..., x_k\}$ such that any shared neighbors of $x_1, ..., x_k$ is in N. For a legal partial state assignment $\vec{n}$ of N,

$$\varepsilon[U(G; x_1, x_2, ...x_k); state(x) = s, state(N) = \vec{n}] =$$
$$\max_{s_1, ..., s_k} \{\varepsilon[G; state(x_1) = s_1, ..., state(x_k) = s_k, state(N) = \vec{n}]\}$$

where $s = \tau(s_1, ..., s_k, N, \vec{n})$

Proof: Follows the strategy of the proof of Lemma 1 by considering cases that depend on the state s of the collapsed vertex. For each case one can show that the function $\tau^{-1}(s)$ considers exactly those configurations of G that could yield this state and that any non-collapsing vertex preserves the number of neighbors in a dominating set.

Pruning a vertex involves collapsing $k+1$ sets of k vertices each, where each face (or k-clique) is viewed as a disjoint component. According to τ, when collapsing a vertex in the dominating set the information about shared dominators is not needed. Hence our 2-step approach of collapsing the dominating vertices first. This provides the information about shared dominators needed to subsequently collapse the remaining vertices.

More formally, consider a graph G containing $k+1$ disjoint components, $C_1, ..., C_{k+1}$, with vertices $x_i^j \in C_j$ for all $i \neq j$ $(1 \leq i, j \leq k+1)$. Each pair of vertices with identical subscript has the same adjacency relation within each component (for our application to k-trees, x_{k+1} is the vertex to be pruned, C_{k+1} is its base, $C_1, ..., C_k$ are its faces and the neighborhood relation is determined by adjacencies in the partial k-tree).

Our goal is to establish the value of

$$\varepsilon[U(G; x_1^2, x_1^3, ..., x_1^{k+1}; x_2^1, x_2^3, ..., x_2^{k+1}; ...; x_{k+1}^1, x_{k+1}^2, ..., x_{k+1}^k), state(x_1) = s_1, ..., state(x_k) = s_k]$$

$$\max_{s_1,...,s_k} \{\varepsilon[G_{0123}; state(v_1) = s_1, ..., state(v_k) = s_k; state(V_{012}) = \vec{v}_{012}, state(V_3) = \vec{v}_3]\}$$

with the shared dominators determined by adjacencies in G_{0123}. Note that the dominators of G_{0123} are the same as the dominators of G_3. Combining this with the inductive assumption we get

$$e = \max_{s_1,...,s_k} \{\varepsilon[G_{0123}; state(v_1) = s_1, ..., state(v_k) = s_k; state(V_{012}) = \vec{v}_{012}, state(V_3) = \vec{v}_3]\} =$$

$$\max_{s_1,...,s_k} \{\max_{\vec{c}_{012}} \{\varepsilon[G_3; state(C_{012}) = \vec{c}_{012}, state(v_1) = s_1, ..., state(v_k) = s_k; state(V_3) = \vec{v}_3]\}\}$$

Since all the shared dominators of $v_1, ..., v_k$ are in G_3 and since the maximum function is associative, we can include the vertex v in the set V_{012} and redefine G_{0123} accordingly. This establishes the inductive step and hence the lemma.□

Our goal is to prune the vertex x_{k+1}. We do this by expressing the efficiency of its base, for a particular state sequence, maximized over all possible states of x_{k+1}. As with the other vertices, if the pruned vertex is viewed as belonging to the dominating set then its collapsing must occur in the first stage, otherwise it must occur in the second stage. Since we need to maximize the resulting efficiency over all states of the pruned vertex, we must consider each of these possibilities.

Theorem 3: The efficiency value of the base of a pruned vertex, for a particular legal state sequence of vertices in the base, is given by the the maximum efficiency over all 4 states of the pruned vertex resulting from the 2 steps:

1) Using Lemma 3, collapse corresponding vertices of each face (and base) that belong to the dominating set.

2) Using Lemma 4, collapse the remaining vertices based on the adjacencies resulting from step 1.

Proof: By maximizing over all states of the pruned vertex, we satisfy the maximality condition in the definition of efficiency. The theorem follows from the lemmata. □

Algorithm 2: Efficiency of partial k-trees
Input: $G = (V, E)$, a partial k-tree, given with an embedding k-tree $G' = (V, E \cup E')$ and a corresponding vertex elimination ordering $\omega(V)$. Note that $E \cap E' = \emptyset$.
Output: $\varepsilon[G]$

1. Initialize efficiencies of all state vectors of all k-cliques in G' as follows: For a legal state assignment to vertices in a k-clique K , $\varepsilon[K, state(K) = \vec{k}] := 0$. For an illegal state assignment , $\varepsilon[K, state(K) = \vec{k}] := -\infty$ (legality according to the states of vertices in the subgraph of G induced by the vertices of K).

2. For $i = 1$ to $|V| - k$ do *Reduction step:*

   Let v be such that $\omega(v) = i$. Include v into the reduced branches by updating the optimal efficiency values of its base based on the optimal efficiency values of its faces according to the 2-step formula outlined in Lemma 2 and Theorem 3. In doing so, we also make use of the fact that the efficiency of a disconnected graph is the sum of the efficiencies of its components.

3. Return the maximum efficiency value associated with the root of G' while accounting for vertices in the root if they are efficiently dominated.

Timing and correctness

The algorithm follows the standard approach for solving discrete optimization problems on partial k-trees [4]. Note that an illegal partial state assignment will always have the efficiency value $-\infty$ by virtue of the initialization step and the rules for combining efficiencies of components. Correctness thus follows from Theorem 3. A straightforward implementation of the formula given for the *Reduction* step would imply a linear time algorithm involving a constant exponential in k^2. Note that our algorithm assumes that an embedding in a k-tree is given with the partial k-tree. Efficient computation of this embedding is a subject of active research. We know that, for any k, there exists a linear time algorithm recognizing partial k-trees [1], but it requires the explicit knowledge of the set of minimal forbidden minors. On the other hand, there are $\mathcal{O}(nlog^2n)$ algorithms finding k-tree embeddings [6]. For $k \leq 3$, there are linear time algorithms [1, 9].

5 Conclusions

We have constructed an algorithm that, for a partial k-tree given with an embedding k-tree computes the efficiency of the graph in linear time. A suitable modification of the algorithm will allow for a construction of the corresponding dominating set.

The methodology used here, (introduced in [4]) can be applied to many other dominating set problems (cf. [8]). The fact that such linear time algorithms exist follows from the existence of a linear EMSOL (Extended Monadic Second Order Formula) description of the corresponding problems [2]. For instance, that a set X is efficiently dominated by a set D can be expressed as a MSOL formula $EDom$:

$$EDom(X, D) \equiv \forall v.v \in X \rightarrow \exists u.(u \in D \wedge Adj(v, u) \wedge \forall w.w \in D \rightarrow (u = w \vee \neg Adj(v, w)))$$

where Adj is the vertex adjacency relation in a given graph.

References

[1] S. ARNBORG, B. COURCELLE, A. PROSKUROWSKI AND D. SEESE, An algebraic theory of graph reduction. *Technical Report 90-02, Laboratoire Bordelais de Recherche en Informatique, Bordeaux, 1990.*

[2] S. ARNBORG, J. LAGERGREN AND D. SEESE, Problems easy for tree-decomposition and related problems, *FOCS '90, 173-182.*

[3] S. ARNBORG AND A. PROSKUROWSKI, Characterization and recognition of partial 3-trees, *SIAM J. Alg. and Discr. Methods 7 (1986) 305-314.*

[4] S. ARNBORG AND A. PROSKUROWSKI, Linear time algorithms for NP-hard problems on graphs embedded in k-trees, *Discr. Appl. Math. 23 (1989) 11-24.*

[5] P.T. BERNHARD, S.T. HEDETNIEMI AND D.P. JACOBS, Efficient sets in graphs, manuscript (1990).

[6] H.L. BODLAENDER AND T. KLOKS, Better algorithms for the pathwidth and treewidth of graphs, manuscript (1990).

[7] J.A. BONDY AND U.S.R. MURTY, *Graph Theory with Applications*, North Holland (1976).

[8] S.T. HEDETNIEMI AND R. LASKAR, Survey of domination problems in graphs, preprint.

[9] J. MATOUSEK AND R. THOMAS, Algorithms finding tree-decompositions of graphs, preprint (1988).

[10] A. PROSKUROWSKI, Recursive graphs, recursive labelings and shortest paths, *SIAM J. Computing 10 (1981) 391-397.*

[11] N. ROBERTSON AND P.D. SEYMOUR, Graph minors II: algorithmic aspects of tree-width, *J. Algorithms 7 (1986) 309-322.*

[12] A. WALD AND C.J. COLBOURN, Steiner trees, partial 2-trees, and minimum IFI networks, *Networks 13 (1983), 159-167.*