Covering a Tree by a Forest

Fanica Gavril and Alon Itai

Department of Computer Science, Technion - IIT, Haifa, Israel
gavril@cs.technion.ac.il, itai@cs.technion.ac.il

Abstract. Consider a tree T and a forest F. The paper discusses the following new problems: The *Forest vertex-cover problem (FVC)*: cover the vertices of T by a minimum number of copies of trees of F, such that every vertex of T is covered exactly once. The *Forest edge-cover problem (FEC)*: cover the edges of T by a minimum number of copies of trees of F, such that every edge of T is covered exactly once. For a solution to always exist, we assume that F contains a one vertex (one edge) tree.

Two versions of Problem FVC are considered: ordered covers (OFVC), and unordered covers (UFVC). Three versions of Problem FEC are considered: ordered covers (OFEC), unordered covers (UFEC) and consecutive covers (CFEC). We describe polynomial time algorithms for Problems OFVC, UFVC and CFEC, and prove that Problems OFEC and UFEC are NP-complete.

Keywords: vertex-cover of a tree by a forest, edge-cover of a tree by a forest, graph algorithms.

1 Introduction

In the present paper we consider only rooted trees. The root of a tree t is denoted by $root(t)$. For a vertex v of t, we denote by $p_t(v)$ the father of v, by $Ch(v)$ its set of children and by $deg(v)=|Ch(v)|$ their number. For a subset $X\subseteq V(t)$, we denote $Ch[X]=\cup_{v\in X}Ch(v)$. We denote by t_v the subtree of t rooted at v and containing v and all its descendants. The number of edges in the unique path between two vertices x,y of t is called *distance between x and y* and is denoted by $dist(x,y)$; $height(t)$ is the distance from $root(t)$ to the farthest leaf. *Isomorphism* between two rooted trees t, f is denoted by $t \approx f$. The connected components $f_1,\ldots,f_q$ of a forest F are rooted trees; for simplicity, we denote by F also the family $\{f_1,\ldots,f_q\}$. For a forest F, we denote by $V(F)$, $E(F)$, $L(F)$ its set of vertices, edges and leaves, respectively.

A *forest vertex-cover* (*forest edge-cover*) of a tree T by a forest F, is a partition of T into vertex (edge) disjoint subtrees $t_1,\ldots,t_k$ such that each t_i is isomorphic to some $f_j\in F$. A *minimum forest vertex-cover* is one which uses a minimum number of copies of trees of F.

We define the following two new problems:

FVC: Find a minimum *forest vertex-cover* of a tree T by a forest F.
FEC: Find a minimum *forest edge-cover* of a tree T by a forest F.

M. Lipshteyn et al. (Eds.): Golumbic Festschrift, LNCS 5420, pp. 66–76, 2009.

To ensure that a cover exists, we assume throughout the paper that F contains a unique tree f_1 consisting of a single vertex, for vertex covers, and of a single edge, for edge covers.

A rooted tree is *ordered* if there exists an order between the children of every vertex. A cover is *ordered* if the trees t_i and f_j are isomorphic as ordered trees. A cover without this restriction is *unordered*. We discuss two versions of Problem FVC: *Problem OFVC* – ordered forest vertex-cover (see Fig. 1), and *Problem UFVC* – unordered forest vertex-cover. Likewise, for edges we have *Problem OFEC* – ordered forest edge-cover, and *Problem UFEC* – unordered forest edge-cover.

In an ordered tree T, let $\{u_1,\ldots,u_{deg(u)}\}$ be the children of a vertex u . We say that the children of u are *covered consecutively* by the children of a vertex x of F if for some $1 \le i \le deg(u)-deg(x)$, the children $u_i,\ldots,u_{i+deg(x)-1}$ of u are all covered by the children of x. A *cover* of T by F is *consecutive* if for every vertex u in T which is covered by a vertex x of F, u's children are covered consecutively by x's children. *Problem CFEC* is to find a minimum consecutive edge-cover of an ordered tree T by a forest F. Note that *Problem CFVC* – to find a minimum consecutive vertex-cover – is a restricted case of OFVC.

For a vertex-cover $t_1,\ldots,t_k$ of a tree T by a forest F, let FC be the multiset of non-trivial subtrees t_j in the forest vertex-cover fulfilling $|V(t_j)|>1$. Let $r=|FC|$: FVC is equivalent to finding a forest vertex-cover which minimizes $r+(|V(T)|-\sum|V(t_j)|)$, that is, it maximizes $\sum|V(t_j)|-r$. The problem is new, having a flavor of both max and min: for a constant $\sum|V(t_j)|$, it minimizes r, while for a constant r, it maximizes $\sum|V(t_j)|$. Therefore, FVC is equivalent to packing into T a set of copies of trees in $F-\{f_1\}$, where $|V(f_1)|=1$, such that $\sum|V(t_j)|-r$ is maximized. When the trees in $F-\{f_1\}$ are of equal cardinality, the problem is of maximizing r, becoming equivalent to the maximum packing problem. When more than one set of trees covers the same number of vertices in T, the problem becomes one of minimizing the number r of trees. Similarly, Problem FEC is equivalent to finding a forest edge-cover which minimizes $r+(|E(T)|-\sum|E(t_j)|)$, that is, maximizes $\sum|E(t_j)|-r$.

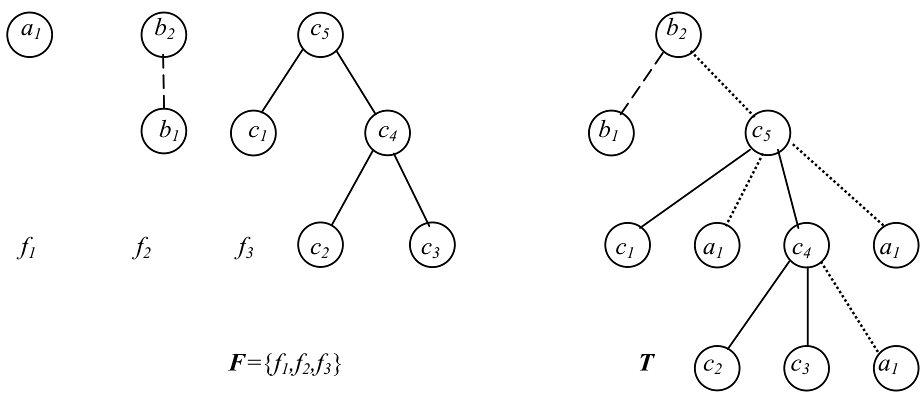

Fig. 1. An ordered forest vertex-cover of size 5: three copies of f_1 and one of f_2 and f_3

The above problems are related to the *subtree isomorphism problem*. Algorithms for the subtree isomorphism problem were given in [6,7,8], while the problem of subgraph isomorphism of a forest F into a tree T is NP-complete [3]. In the present paper we describe two polynomial time algorithms to solve Problems OFVC, UFVC and CFEC, and prove that Problems OFEC and UFEC are NP-complete. One algorithm called MAP-CHILDREN is similar to the algorithm for graph isomorphism and its complexity depends on $|V(F)|$. The other algorithm called MAP-LEAVES seems to be new, and works by replacing in F every maximal directed path in which the internal vertices have only one child, by a single edge; its complexity depends on $|L(F)|$, being very efficient when $|L(F)|$ is much smaller than $|V(F)|$, for example when the trees in F are paths. The algorithms can be extended to unrooted trees by considering copies of T rooted at each one of its vertices and extending the family $F=\{f_1,\ldots,f_q\}$ to contain copies of every f_i rooted at each one of its vertices.

Problem FVC in a restricted form was discussed by Golumbic [4] for the factorization of a tree Boolean function as a read-once (fan-out) function, and by Levin and Pinter [5] for the realization of a tree Boolean function using a minimum number of logic circuits. An additional application is in translation: we wish to cover a syntax tree of a source language sentence by a minimum number of phrases, each of which has an optimal translation to the target language.

In Sections 2,3 we describe polynomial time algorithms to solve Problems OFVC and UFVC: in Section 2 the complexity depends on $|V(F)|$, while in Section 3, the complexity depends on $|L(F)|$. In Section 4 we describe similar algorithms to solve maximum packing problems of copies of trees of F into T. In Section 5 we prove that Problems OFEC and UFEC are NP-complete. In Section 6 we describe a polynomial time algorithm to solve CFEC.

2 Algorithm MAP-CHILDREN for Forest Vertex-Cover

Consider a tree T and a forest F. We shall describe how to extend covers of a subtree T_u of T, consisting of u and all its descendants, to a complete cover of T. Let $u\in V(T)$, $f\in F$ and $x\in V(f)$. An $[T_u, f_x]$ *forest vertex-cover of* T_u is an $F\cup\{f_x\}$ forest vertex-cover of T, such that $u=root(T_u)$ is covered by $x=root(f_x)$ and when $x\neq root(f)$, f_x is used only once in the cover. Note that if a vertex u is covered by a vertex x of $f\in F$, $x\neq root(f)$, then the parent $p_f(x)$ of x must cover the parent $p_T(u)$ of u. Let $W(u,x)$ be the number of trees in a minimum $[T_u, f_x]$ forest vertex-cover of T_u. Let $W(u)$ denote the number of trees in a minimum forest vertex-cover of T_u. Clearly $W(u)=min_{f\in F}\{W(u,root(f))\}$.

Consider first the case where x is a leaf of f, i.e., $V(f_x)=\{x\}$. Then in any $[T_u, f_x]$ forest vertex-cover of T_u, u is covered by x and each of its children is covered by the root of a tree of F. Hence, $W(u,x) = 1 + \sum_{v\in Ch(u)} W(v)$.

In the general case, consider an $[T_u, f_x]$ forest vertex-cover of T_u; let $X\subseteq V(T)$ be the set of vertices of the subtree rooted at u of T_u covered by f_x. $T_u - X$ is a set of disjoint subtrees of T_u, each subtree rooted at a vertex in $Ch[X] - X$ and covered by a forest vertex-cover of F. Let $T[X]$ denote the vertex subgraph of T induced by X. Hence,

$$W(u,x)=1+min_X\{\textstyle\sum_{z\in Ch[X-\{u\}]-X}W(z)+\sum_{v\in Ch(u)-X}W(v) : X\subseteq V(T_u),\ T[X]\approx f_x,\ root(T[X])=u\}. \quad (1)$$

The above equation requires us to find all isomorphic copies of f_x rooted at u and hence might lead to an exponential time algorithm. To get a polynomial time algorithm we will show how to compute $W(u,x)$ from the values of $W(v),W(y),\ v\in Ch(u),\ y\in Ch(x)$.

An $[T_u, f_x]$ forest vertex-cover of T_u exists only if $deg(x)\leq deg(u)$. We construct the cover of T_u from optimal covers of the children of u: $deg(x)$ of u's children are covered by the $deg(x)$ trees in $\{f_z : z\in Ch(x)\}$ and the remaining children of u are covered by trees of F. A *matching* is an injection $\mu : Ch(x)\to Ch(u)$; let $M[Ch(x),Ch(u)]$ denote the set of possible matchings of $Ch(x)$ into $Ch(u)$. Therefore,

$$W(u,x) = 1 - deg(x) + min_{\mu\in M[Ch(x),Ch(u)]}\{\textstyle\sum_{1\leq j\leq deg(x)} W(\mu(x_j),x_j) + \sum_{v\in Ch(u)-\mu(Ch(x))} W(v)\} \qquad (2)$$

Finally, $W(u)=min_{f\in F}\ W(u,root(f))$, and the size of a minimum cover is $W(root(T))$.

We describe a generic dynamic programming algorithm which is conducted by a postorder traversal of T and F, that is, it considers the children of u and x before considering u and x. This ensures that when evaluating the left side of equation (2), the values of the right side have been evaluated.

The dynamic programming algorithm traverses T and F in postorder and for every $u\in V(T), f\in F$ and $x\in V(f)$, it finds the size of a minimum $[T_u, f_x]$ forest vertex-cover of T_u. To obtain $W(u)$ the algorithm finds the $f\in F$ which minimizes $W(u) = min_{f\in F}\ \{W(u,root(f))\}$. Since the trees of F are disjoint, once a vertex x is chosen, the tree $f\in F$ to which it belongs and the vertex of T covered by $root(f)$ are uniquely determined. When x is a leaf of f, hence $f_x=\{x\}$, the algorithm already evaluated for T_u the minimum cover of every tree T_v, $v\in Ch(u)$, thus $W(u,x) = 1 + \sum_{1\leq i\leq deg(u)} W(u_i)$.

When x is not a leaf of f, the algorithm simulates a minimum weight isomorphism algorithm of f_x into T_u, by assuming that the corresponding ancestor of u, will be covered by $root(f)$. Thus, by optimally covering the children of u by the children of x, it carries to u (and to $root(f)$) the sizes of the minimum covers. This is done as follows:

A matching μ *between a sequence of vertices* $(u_1,\ldots,u_m)$ and $(x_1,\ldots,x_r)$ *is non-crossing* when every pair x_i,x_{i+1} is matched to a pair u_j,u_k fulfilling $j<k$. For each subproblem P we will restrict the permitted matchings to a subset $MP[Ch(x),Ch(u)]$ of all possible matchings. For OFVC, MP is the set of non-crossing matchings and for UFVC, MP is the set of all matchings. To every pair $[u_i,x_j]$ we assign a cost equal to $W(u_i,x_j)$. Hence, a minimum cost matching μ^* of $Ch(x)$ into $Ch(u)$ in equation (2), will give us the optimal way of covering the children of u by the children of x. This, together with the postorder traversal ensures by induction that the dynamic programming algorithm is correct.

Equation (2) is rewritten below as a minimum cost matching $\mu^*\in MP[Ch(x),Ch(u)]$ of a complete bipartite graph $[Ch(x),Ch(u)]$, in which the cost of every edge (u_i,x_j) is $W(u_i,x_j)-W(u_i)$:

$$W(u,x) = 1 - deg(x)+\textstyle\sum_{1\leq i\leq deg(u)} W(u_i) + \sum_{1\leq j\leq deg(x)} [W(\mu^*(x_j),x_j) - W(\mu^*(x_j))]\ . \qquad (3)$$

We denote $WS(u) = \sum_{1\leq i\leq deg(u)} W(u_i)$ and $cost(\mu^*)=\sum_{1\leq j\leq deg(x)} [W(\mu^*(x_j),x_j) - W(\mu^*(x_j))]$.

The algorithm keeps pointers along the minimum cost matchings to retrace the minimum forest vertex-cover when $u=root(T)$, and does not have to remember the intermediate forest vertex-covers.

Algorithm MAP-CHILDREN
for every $u \in V(T)$ in postorder $W(u)=\infty$;
  if u is a leaf **then** $W(u)=1$, $WS(u)=0$;
  if u is not a leaf **then** $WS(u) = \sum_{v \in Ch(u)} W(v)$;
  for every $x \in V(F)$ in postorder
    if u is a leaf
      if x is a leaf **then** $W(u,x)=1$ **else** $W(u,x)=\infty$;
    if u is not a leaf
      if x is a leaf **then** $W(u,x) = 1+WS(u)$;
      if x is not a leaf
        Let μ^* be a minimum cost matching in $MP[Ch(u),Ch(x)]$;
        $W(u,x) = 1 - deg(x) + WS(u) + cost(\mu^*)$ // equation (3)
    if $x=root(f)$, for some $f \in F$ **then** $W(u)=min\,\{W(u,x), W(u)\}$;
  if $u=root(T)$ **then** $W(u)$ is the size of the minimum cover;
end

In order to use the generic algorithm MAP-CHILDREN we need to specify how to calculate equation (3). For OFVC, MP is the set of non-crossing matchings. Assuming that $W(u_i)$ and $W(u_i,x_j)$ ($u_i \in Ch(u)$ and $x_j \in Ch(x)$) have been computed, we need to find a minimum cost non-crossing matching μ^*. We use dynamic programming again:

Let $S_{Ch(u)Ch(x)}[i,j]$ be the value of the minimum cost non-crossing matching between $u_1,\ldots,u_i$ and $x_1,\ldots,x_j$ where $S_{Ch(u)Ch(x)}[1,1]=W(u_1,x_1)$ and $S_{Ch(u)Ch(x)}[i,j]=\infty$ if $i<j$. Then

$$S_{Ch(u)Ch(x)}[i,j]=min\,\{S_{Ch(u)Ch(x)}[i-1,j-1]+W(u_i,x_j), S_{Ch(u)Ch(x)}[i-1,j]+W(u_j)\} \text{ for } i>1,\, j\leq i\,. \quad (4)$$

To compute all $S_{Ch(u)Ch(x)}[i,j]$'s for given vertices u,x it takes $O(deg(u)deg(x))$ time. For all vertices, the required time is

$$\textstyle\sum_{u \in V(T)} \sum_{x \in V(F)} deg(u)deg(x) = \sum_{u \in V(T)} deg(u) \sum_{x \in V(F)} deg(x) < |V(T)||V(F)|\,. \quad (5)$$

To find a minimum unordered cover UFVC, when the tree T and the forest F are unordered, we also apply MAP-CHILDREN. The only difference is that in equation (3) we drop the constraint that the matching be non-crossing, i.e., MP is the set of all matchings between $Ch(x)$ and $Ch(u)$.

To find the matching μ^* in the complete bipartite graph $MP[Ch(x),Ch(u)]$ we follow [1,7,8] and employ the maximal flow minimum cost algorithm of [2] to yield an algorithm that requires $O(\sum_{f \in F}|V(f)|^{1.5}\, |V(T)|\, log|V(T)|)$ time.

3 Algorithm MAP-LEAVES for Forest Vertex-Cover

In this section we construct an algorithm for OFVC and UFVC whose complexity depends on the number $|L(F)|$ of leaves of F, which may be much smaller than the number of vertices of F. The key step is to cover the vertices of T by the leaves of F: covering a vertex $u\in V(T)$ by a leaf x of f determines how the path from x to $root(f)$ covers vertices of T. However, in order not to scan each vertex of f separately, we shall replace the tree f by its *skeleton* – the tree $skel(f)$ – resulting by replacing every maximal directed path in which the internal vertices have only one child, by a single edge. Now, every internal vertex of the tree $skel(f)$ has at least two children. Thus, $|V(skel(f))|\leq 2|L(f)|$. Let $SKEL = \{skel(f) : f\in F\}$ and let $SKEL^+$ be the set containing the vertices of $SKEL$ and their children in F. Since every edge of $SKEL$ gives rise to one child of F, $|SKEL^+|\leq|2V(SKEL)|=O(|L(F)|)$.

Consider covering the vertex $u\in V(T)$ by a vertex $x\in V(skel(f))$, $skel(f)\in SKEL$. If u is a leaf then it can be covered only by leaves of $SKEL$. If u is not a leaf and x is a leaf, then u is covered by x, each of its children $v\in Ch(u)$ is covered by the root of a tree of F, and T_v is covered by a minimum forest vertex-cover with trees of F. If neither u nor x is a leaf, then each child of x in f must cover a child of u. The edge in $skel(f)$ connecting x to a child y_x, corresponds in f to a path $(x,x_j,\ldots,y_x)$. Assume that x_j covers u_i. Then u_i must have a descendant at distance $dist(x_j,y_x) = dist(x,y_x)-1$ which is covered by y_x.

Let $W(u)$, $WS(u)$ and $W(u,x)$ be as defined in Section 2. If u' is a descendant of u, let $Path_T(u,u')$ denote the path in T from u to u'. Let $CPath_T(u,u')$ be the set of children of vertices in $Path_T(u,u')-\{u'\}$, children which are not in $Path_T(u,u')$, and let $WP(u,u')=\sum_{v\in CPath(u,u')}W(v)$. We compute $WP(p_T(u),u')$ from $WP(u,u')$ by

$$WP(p_T(u),u') = WP(u,u') + \sum\{W(v) : v\in Ch(p_T(u))\} - W(u) = WP(u,u') + WS(u) - W(u). \quad (6)$$

To compute $W(u,x)$ for $u\in V(T)$, $x\in V(f)$, $f\in F$, we need to decide which children of u should be covered by the children of x in $f(x)$. Let (x,y_x) be an edge in $skel(f)$, let x_j be the child of x on the path in f from x to y_x and let $d = dist(x_j,y_x)$; to every child x_j of x corresponds exactly one y_x and one d. Let $\mathcal{D}=\{dist(x_j,y_x) : x_j\in SKEL^+, p_f(x_j)=x\}$. Since to every child x_j of x corresponds exactly one y_x it follows that $|\mathcal{D}|\leq|SKEL^+|=O(|L(F)|)$. Let $D[u_i,d]$ be the list of descendants of $u_i\in V(T)$ at distance d from u_i. Thus, for children u_i of u and x_j of x we have

$$W(u_i,x_j) = min_{u'}\ \{W(u',y_x) + WP(u_i,u') : d = dist(x_j,y_x),\ u'\in D[u_i,d]\}\ . \quad (7)$$

Now, according to equation (3), rewritten below as (8), we need to find a minimum cost matching $\mu^*\in MP[Ch(u),Ch(x)]$ of a complete bipartite graph $(Ch(u),Ch(x))$, where the cost of every edge (u_i,x_j) is $W(u_i,x_j) - W(u_i)$ and evaluate:

$$W(u,x) = 1 - deg(x) + \sum_{v\in Ch(u)} W(v) + \sum_{z\in Ch(x)}[W(\mu^*(z),z) - W(\mu^*(z))]\ . \quad (8)$$

By equations (5-7) we do not have to compute $W(u,x)$ for all vertices $x\in V(F)$, but only for vertices x in $SKEL^+$. Thus we need to compute only $O(|V(T)||L(F)|)$ such values.

In the preprocessing stage we discard from F the trees whose height exceeds $height(T)$ and prepare $SKEL$, $SKEL^+$ and $\mathcal{D}$; this requires $O(|V(F)|)$ time. Also, we prepare $D[u,d]$ for all $u\in V(T)$ and $d\in\mathcal{D}$. Let $d_{max}=max\ \{d : d\in\mathcal{D}\}$; clearly

$d_{max} \leq height(T)$. In the worst case, each vertex appears in the list of all its ancestors, and so this requires $O(|V(T)|\, d_{max})$ time. The algorithm traverses T in postorder, and at vertex u it examines the cost of covering u by every $x \in SKEL^+$.

Algorithm MAP-LEAVES
for every $u \in V(T)$ in postorder $W(u)=\infty$;
  if u is a leaf **then** $W(u)=1,\ WS(u)=0$;
  if u is not a leaf **then** $WS(u) = \sum_{v \in Ch(u)} W(v)$;
    for every $v \in Ch(u)$ // *compute WP*
      for all descendants w of v at distance at most d_{max}
        $WP(u,w) = WS(u) - W(v) + WP(v,w)$;
  for every $x \in SKEL^+$ in postorder
    if u is a leaf
      if x is a leaf **then** $W(u,x)=1$ **else** $W(u,x)=\infty$;
    if u is not a leaf
      if x is a leaf **then** $W(u,x) = 1 + WS(u)$;
     else if $x \in V(SKEL)$
        Let μ^* be a minimum cost matching in $MP[Ch(u),Ch(x)]$
        $W(u,x) = 1 - deg(x) + WS(u) + cost(\mu^*)$ // equation (7)
        if $x \in SKEL^+ - V(SKEL)$
          let y_x be the closest descendant of x that belongs to a tree in $SKEL$;
           $d = dist(x,y_x)$;
           $W(u,x) = min_{w \in D[u,d]} \{W(w,y_x) + WP(u,w)\}$; // y_x is unique for x (9)
  if $x=root(f), f \in F$ **then** $W(u)=min\ \{W(u,x), W(u)\}$;
if $u=root(T)$ **then** $W(u)$ is the size of the minimum cover;
end

The computation of $WP(u,w)$ for every u,w requires $O(|V(T)|\, d_{max})$ time, since each vertex w appears only in the list of its ancestors. The vertex $w \in V(T)$ in equation (9) is considered for its ancestor u at distance $d = dist(x,y_x)$. Thus for each $x \in SKEL^+$, w appears in $O(|SKEL^+|)$ computations of the minimum in (9). Hence, over all $w \in V(T)$, the number of vertices considered in the computation of all the minima in (9) is $O(|V(T)||SKEL^+|)=O(|V(T)||L(F)|)$. Since the matching can be found as in Section 2, we obtain: For OFVC, Algorithm MAP-LEAVES requires $O(|V(T)|d_{max}+|V(T)||L(F)|) \leq O(|V(T)|height(T) + |V(T)||L(F)|)$ time. For UFVC, Algorithm MAP-LEAVES requires $O(|V(T)|d_{max}+\sum_{f \in F}|L(f)|^{1.5}|V(T)|\ \log|V(T)|) \leq O(|V(T)|height(T)+\sum_{f \in F}|L(f)|^{1.5}|V(T)|\log|V(T)|)$ time.

4 Algorithms for Maximum Packing of a Forest in a Tree

Algorithms MAP-CHILDREN and MAP-LEAVES can be used for many other optimization problems on T and F. For example, finding a maximum packing of vertex disjoint copies of trees of F into T, can be solved in polynomial time; here we assume that F contains no single vertex tree, otherwise the problem is trivial. This problem has two versions, one to maximize the number of packed trees and another to maximize the number of covered vertices of T. Denote by $W(u)$ the number of trees in a maximum forest packing of T_u. An $[T_u, f_x]$ *forest packing of* T_u is an $F\cup\{f_x\}$ forest packing of T_u, such that $u=root(T_u)$ is covered by $x=root(f_x)$ and when $x\neq root(f)$, f_x is used only once in the packing. Let $W(u,x)$ be the number of trees in a maximum $[T_u, f_x]$ forest packing of T_u. Then, similarly to equation (1),

$$W(u,x)=1+max_X\{\textstyle\sum_{z\in Ch[X-\{u\}]-X}W(z)+\sum_{v\in Ch(u)-X}W(v)\ X\subseteq V(T_u),\ T[X]\approx f_x,\ root(T[X])=u\}. \quad (10)$$

Note that if a vertex u is covered by a vertex x of $f\in F$, $x\neq root(f)$, then $p_T(u)$ must be covered by $p_f(x)$. $W(u,x)$ can be evaluated as a maximum weight matching $\mu^*\in MP[Ch(x),Ch(u)]$ of a complete bipartite graph $[Ch(x),Ch(u)]$, in which the weight of every edge (u_i,x_j) is $W(u_i,x_j)-W(u_i)$:

$$W(u,x) = 1 - deg(x) + \textstyle\sum_{1\leq i\leq deg(u)} W(u_i) + \sum_{1\leq j\leq deg(x)} [W(\mu^*(x_j),x_j) - W(\mu^*(x_j))]. \quad (11)$$

Clearly $W(u)=max\ \{\sum_{1\leq i\leq deg(u)} W(u_i)\ ,\ max_{f\in F}\{W(u,root(f))\}\}$ and the size of a maximum packing is $W(root(T))$; when u is a leaf, $W(u)=0$.

For a packing covering a maximum number of vertices of T, equation (11) is replaced by: $W(u,x) = 1+ \sum_{1\leq i\leq deg(u)} W(u_i) + \sum_{1\leq j\leq deg(x)} [W(\mu(x_j),x_j) - W(\mu(x_j))]$.

The complexity of the algorithms is similar to the complexity of the algorithms in Sections 2, 3.

5 Covering the Edges by a Minimum Ordered or Unordered Forest Edge-Cover

Consider a tree T and a forest F, $F=\{f_i : i=1,..,k\}$. For $u\in V(T)$ and $x\in V(F)$, since we are looking for an edge-disjoint cover, the edge from the parent $p_T(u)$ of u to u must be covered by exactly one edge of the forest F.

We prove that the Problems OFEC and UFEC are NP-complete, by reducing to them the NP-complete problem of Exact Cover by 3-Sets (X3C) [3].

Problem: Exact Cover by 3-Sets (X3C)
Instance: A set $X=\{v_1,...,v_n\}$ and a family of 3-subsets $S=\{s_1,...,s_k\}$ of X.
Question: Is there a subfamily $S'\subseteq S$ s.t. every $v_i\in X$ is contained in exactly one set in S' ?

Theorem 1: The problems of exact covering of the edges of a tree T by a minimum ordered or unordered forest edge-cover are NP-complete.

Proof. We show that the problem X3C is reducible to the Problems OFEC and UFEC, i.e., for each instance of X3C we show an instance of the edge-cover problem that has a cover of size $n/3$ if and only if X3C has a solution.

Consider a set $X=\{v_1,...,v_n\}$ and a family of 3-subsets $S=\{s_1,...,s_k\}$ of X. We construct a tree T (Fig. 2a) with root v whose children are $v_1,...,v_n$ and at every v_i we attach a subtree T_i defined as follows (Fig. 2b): T_i's root is v_i, v_i has i children which are leaves and v_i has attached a path with $n-i+1$ vertices. Clearly, every T_i has exactly $n+2$ vertices and no two T_i's are isomorphic. For every $s_j=\{v_a,v_b,v_c\}\in S$, $a<b<c$, we

define a tree f_j (Fig. 2c) with root s_j, children v_a, v_b, v_c from left to right, and copies of T_a, T_b, T_c attached as subtrees. Let $F=\{f_1, f_2, \ldots, f_k\}\cup\{e\}$, where e is the tree consisting of two vertices and an edge between them. Consider an ordered or unordered forest edge-cover of size $n/3$ of T; such a covering is minimum since all vertices s_j's are mapped on v. Then, for every child v_i of v in T, there exists some f_j with child v_i of s_j, covering the child v_i of v. Thus, a forest edge-cover of size $n/3$ of T by trees in F gives an exact covering of X by subsets in S.

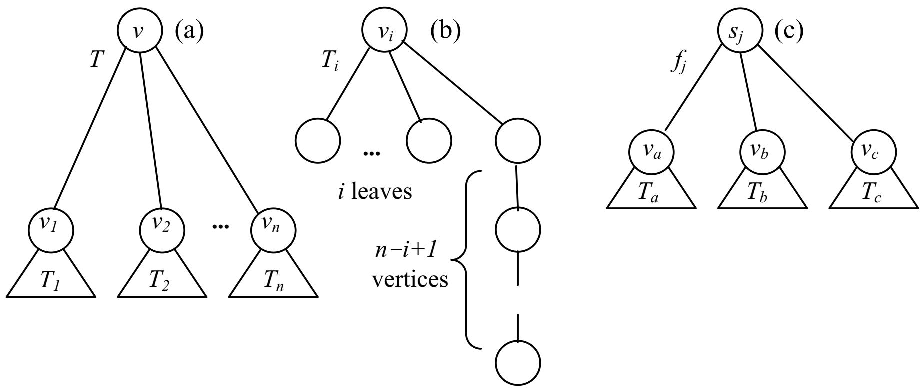

Fig. 2. An instance of forest edge-cover problem corresponding to an instance of X3C

Conversely, an exact covering of X by subsets $s_{i,1},\ldots,s_{i,n/3}\in S$ will give a cover $f_{i,1},\ldots,f_{i,n/3}\in F$ of the edges of T. Note that the order of the edges in the f_i's is compatible to that of T. Thus any exact cover is an ordered cover. □

By the same reduction, the maximum packing problems of edge disjoint copies of trees of F into a tree T, are NP-complete.

6 An algorithm for a Minimum Consecutive Forest Edge-Cover

Problem CFEC, finding a minimum consecutive edge-cover, assumes that T is an ordered tree and if $u\in V(T)$ is covered by $x\in V(F)$, then the children of u covered by the children of x, are consecutive in the order of T. Let the edge from the parent of a vertex v to v in a tree t be denoted by $p_t(v)\rightarrow v$. For non-roots $u\in T$ and $x\in V(F)$, since we are looking for an edge-disjoint cover, the edge $p_T(u)\rightarrow u$ should be covered by exactly one edge of F, the edge $p_f(x)\rightarrow x$. Let us denote $f_x^{+}= f_x\cup\{p_f(x)\rightarrow x\}$, $T_u^{+}=T_u\cup\{p_T(u)\rightarrow u\}$. For non-roots u, x, let $[T_u^{+}, f_x^{+}, j]$ denote a *consecutive edge-cover* of the subtree T_u^{+} by the forest $F\cup\{f_x^{+}\}$ such that the edge $p_f(x)\rightarrow x$ covers the edge $p_T(u)\rightarrow u$, the children $\{x_1,\ldots,x_{deg(x)}\}$ of x cover the children $u_{j-deg(x)+1},\ldots,u_j$ of u, and the tree f_x^{+} is used only once in the cover.

Let $T_u(i,j)$ denote the subtree of T_u containing u (as root), the children $u_i,\ldots,u_j$ of u and all the children's descendants. The algorithm is based on the observation that in a

$[T_u^+, f_x^+, j]$ consecutive edge-cover, the subtrees $T_u(1,j-deg(x))$ and $T_u(j+1,deg(u))$ have consecutive edge-covers by F, that is, u is covered only by roots of trees in F, while $T_u(j-deg(x)+1,\ j)\cup\{p_T(u)\to u\}$ has a consecutive edge-cover by the forest $F\cup\{f_x^+\}$, using the tree f_x^+ only once.

Let $\mu_{u,x,j}$ be a minimum cost matching of the complete bipartite graph $[Ch(x),\{u_{j-deg(x)+1},\ldots,u_j\}]$ in which the cost of every edge (v,z), $v\in\{u_{j-deg(x)+1},\ldots,u_j\}$, $z\in Ch(x)$, is the size of a minimum $[T_v^+, f_z^+, j]$ consecutive edge-cover. Let $W(u,i,j)$ be the cardinality of a minimum consecutive edge-cover of $T_u(i,j)$ by F. For every u and j we will evaluate

$$W(u,1,j) = min_{f\in F}\{W(u,1,\ j-deg(root(f)) + cost(\mu_{u,root(f),j})\} \tag{12}$$

$$W(u,j,deg(u)) = min_{f\in F}\{W(u,\ j+deg(root(f),deg(u)) + cost(\mu_{u,root(f),j+deg(root(f)-1)})\}\ . \tag{13}$$

For non-roots u, x, let $WR(p_T(u)\to u,p_f(x)\to x)$ be the size of a minimum among all j, $1\le j\le deg(u)$, of a $[T_u^+, f_x^+, j]$ consecutive edge-cover, Therefore

$$WR(p_T(u)\to u,p_f(x)\to x)=min_{deg(x)\le j\le deg(u)}\{W(u,1,j-deg(x))+W(u,j+1,deg(u))+cost(\mu_{u,x,j})\}. \tag{14}$$

Algorithm MAP-EDGES

for every $u\in V(T)$ in postorder
  for every $f\in F$
   for $j=1,\ldots,deg(u)$ set $W(u,1,j)=W(u,j,deg(u))=\infty$;
  if u is not a leaf **then**
   for every $j=deg(u)-deg(root(f)+1,\ldots,deg(u)$
    find a minimum cost matching $\mu_{u,root(f),j}$;
    $W(u,1,j)=min_{f\in F}\{W(u,1,\ j-deg(root(f)) + cost(\mu_{u,root(f),j})\}$;
   for every $j=deg(u),deg(u)-1,\ldots,\ deg(u)-deg(root(f)+1$
    $W(u,j,deg(u))=min_{f\in F}\{W(u,\ j+deg(root(f),deg(u)) + cost(\mu_{u,root(f),j+deg(root(f)-1)})\}$;
 if $u\ne root(T)$ **then**
  for every $x\in f\in V(F)$, $x\ne root(f)$, in postorder
   if u is a leaf **then**
    if x is a leaf **then** $WR(p_T(u)\to u,p_f(x)\to x)=1$ **else** $WR(p_T(u)\to u,p_f(x)\to x)=\infty$;
    if u is not a leaf **then**
    if x is a leaf **then** $WR(p_T(u)\to u,p_f(x)\to x)=1+W(u,1,deg(u))$;
    if x is not a leaf **then**
     Let $\mu_{u,x,j}$ be a minimum cost matching of the complete bipartite graph $MP[Ch(x),\{u_{j-deg(x)+1},\ldots,u_j\}]$;
    $WR(p_T(u)\to u,p_f(x)\to x)=min_{deg(x)\le j\le deg(u)}\{W(u,1,j-deg(x))+W(u,j+1,deg(u))$ $+cost(\mu_{u,x,j})\}$;
  if $u=root(T)$ **then** $W(u,1,deg(u))$ is the size of the minimum edge-cover;
end

The algorithm works in $O(\sum_{f\in F}|V(f)|^{1.5}\ |V(T)|\ log|V(T)|)$ time by the matching algorithms in [1,2].

The maximum packing problems of edge disjoint copies of trees of F into T, where the children of u covered by the children of x, are consecutive in the order of T, can also be solved in polynomial time; here we assume that F contains no single vertex and no single edge tree otherwise the problem is trivial. This is done by an algorithm similar to the above, by changing the equations (12), (13) to:

$$W(u,1,j) = min\ \{W(u,1,j-1),\ min_{f\in F}\{W(u,1,\ j-deg(root(f)) + cost(\mu_{u,root(f),j})\}\}. \quad (15)$$

$$W(u,j,deg(u)) = min\ \{W(u,1,j+1),\ min_{f\in F}\{W(u,j+deg(root(f),deg(u)) + cost(\mu_{u,root(f),j+deg(root(f)1)})\}\}. \quad (16)$$

References

1. Feder, T., Botwani, R.: Clique Partitions, Graph Compression and Speeding-up Algorithms, J. Comput. Syst. Sci. 51, 261–272 (1995)
2. Gabow, H.N., Tarjan, R.E.: Faster Scaling Algorithms for Network Problems. SIAM J. Comput. 18, 1013–1036 (1989)
3. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory NP-Completeness. W. H. Freeman and Co., San Francisco (1979)
4. Golumbic, M.C., Mintz, A., Rotics, U.: Factoring and Recognition of Read-once Functions using Cographs and Normality. In: DAC 2001, Las Vegas, pp. 109–114 (2001)
5. Levin, I., Pinter, R.Y.: Realizing Expression Graphs using Table-Lookup FPGAs. In: Proceedings of EuroDAC 1993, pp. 306–311 (1993)
6. Lingas, A.: An Application of Maximum Bipartite c-Matching to Subtree Isomorphism. In: CAAP 1983. LNCS, vol. 159, pp. 284–299. Springer, Heidelberg (1983)
7. Pinter, R.Y., Rokhlenko, O., Tsur, D., Ziv-Ukelson, M.: Approximate Labelled Subtree Homeomorphism. In: Sahinalp, S.C., Muthukrishnan, S.M., Dogrusoz, U. (eds.) CPM 2004. LNCS, vol. 3109, pp. 59–73. Springer, Heidelberg (2004)
8. Shamir, R., Tsur, D.: Faster Subtree Isomorphism. J. Algorithms 33, 267–280 (1999)