A primal-dual method for approximating tree cover with two weights

Takashi Doi\textsuperscript{a,1}, Toshihiro Fujito\textsuperscript{b,*}

\textsuperscript{a} Department of Information Electronics, Nagoya University, Furo, Chikusa, Nagoya, 464-8603, Japan
\textsuperscript{b} Department of Information and Computer Sciences, Toyohashi University of Technology, Hibarigaoka, Tempaku, Toyohashi, 441-8580, Japan

Received 24 September 2004; received in revised form 10 June 2005; accepted 1 May 2006
Available online 12 July 2006

Abstract

The tree cover (TC) problem is to compute a minimum weight connected edge set, given a connected and edge-weighted graph \(G\), such that its vertex set forms a vertex cover for \(G\). Unlike related problems of vertex cover or edge dominating set, weighted TC is not yet known to be approximable in polynomial time as well as the unweighted version is. Moreover, the best approximation algorithm known so far for weighted TC is far from practical in its efficiency. In this paper we consider a restricted version of weighted TC, as a first step towards better approximation of general TC, where only two edge weights differing by at least a factor of 2 are available. It will be shown that a factor 2 approximation can be attained efficiently (in the complexity of max flow) in this case by a primal-dual method. Even under the limited weights as such, the primal-dual arguments used will be seen to be quite involved, having a nontrivial style of dual assignments as an essential part, unlike the case of uniform weights.

Keywords: Approximation algorithms; Tree cover; Primal-dual method

1. Introduction

In an undirected graph \(G = (V, E)\) a vertex is said to cover all the edges incident to it, and similarly, an edge dominates all the edges adjacent to it. A vertex cover is such a vertex set \(C \subseteq V\) that collectively covers all the edges in \(G\), whereas an edge dominating set is an edge set \(D \subseteq E\) collectively dominating all the other edges in \(G\). A tree cover (tc) for connected \(G\) is defined to be an edge set \(T \subseteq E\) forming a “connected” edge dominating set. Or equivalently, it is a connected edge set such that its vertex set \(V(T)\) forms a vertex cover for \(G\), where \(V(T)\) is the set of vertices induced by \(T\) (and in general, \(V(F) = \{u \in V \mid \exists \text{ an edge } \in F \text{ incident with } u\}\) for any edge set \(F \subseteq E\)). The vertex cover (VC), edge dominating set (EDS), and tree cover (TC) problems are to compute a vertex cover, an edge dominating set, and a tree cover, respectively, of minimum weight in a graph, where either vertices or edges are associated with nonnegative weights. The problems VC and EDS are classic \(NP\)-hard graph problems, and the TC problem is also \(NP\)-hard even in the unweighted case (i.e., all the edge weights are equal) since it then becomes equivalent to the connected vertex cover problem, which in fact is known to be as hard (to approximate) as

* Corresponding author.

E-mail address: fujito@ics.tut.ac.jp (T. Fujito).

\textsuperscript{1} Currently at NTT Data Corporation.
VC [4]. Given the apparent intractability in exact computation, efficient approximation algorithms for these problems have been studied extensively in the literature. A factor 2 approximation of VC was found early; it suffices to compute any maximal matching \( M \) and output the set \( V(M) \) of vertices matched by \( M \) in the unweighted case. The best approximation algorithm known today for VC, weighted or unweighted, achieves a ratio of \( 2 - (1 - o(1)) \ln \ln n \) [7]. Likewise, weighted EDS was recently found 2-approximable as in the unweighted case [3]. The unweighted version of TC is also known to be approximable within a factor of 2 by simple algorithms [9,1]. On the other hand, TC with general weights was first shown to be approximable within a factor of 3.55 [1], and currently the best bound known is \( 3 + \varepsilon [8,2] \). Thus, general TC is not yet known to be approximable as well as the unweighted version is. Even worse, the algorithms of [8] and [2] are far from practical in their efficiency; either one requires to solve optimally an LP of huge size, and to do so, it inevitably resorts to calling the ellipsoid method as their subroutine. In this paper we consider a restricted version of the weighted TC problem, as a first step towards better approximation of general TC, where edge weights are limited to either \( w_1 \) or \( w_2 \) satisfying \( w_2 \geq 2w_1 \). It will be shown that a factor 2 approximation can be attained efficiently (in the complexity of max flow) in this case by the primal-dual method. Even under the limited weights as such, the primal-dual arguments used will be seen to be quite involved, having a nontrivial style of dual assignments as an essential part, unlike the case of uniform weights.

2. Preliminaries

Given an undirected graph \( G = (V, E) \) with edge weights \( c : E \to \mathbb{Q}_+ \), let \( \tilde{G} = (V, \tilde{E}) \) denote its directed version obtained by replacing every edge \( [u, v] \) of \( G \) by two anti-parallel arcs, \((u, v)\) and \((v, u)\), each of weight \( c([u, v]) \). For a designated vertex \( r \) in \( \tilde{G} \) called root, suppose \( \tilde{T}' \subseteq \tilde{E} \) is a tree rooted at \( r \). It is assumed throughout that the arcs in a rooted tree \( \tilde{T} \) are always directed away from the root to a leaf, and let \( T \) denote the undirected counterpart obtained by ignoring the direction of each arc in \( \tilde{T} \). We say that \( \tilde{T}' \) is an \( r\text{-tc} \) in \( \tilde{G} \) if and only if \( T' \) is a tree covering touching \( r \) in \( G \), and \( r\text{-TC} \) is the problem of computing a minimum weight \( r\text{-tc} \) given \( \tilde{G} \) and \( r \in V \). Clearly, \( c(T') = c(\tilde{T}') \), and as it suffices to approximate \( r\text{-TC} \) well for our purpose, we focus on this variant instead of approximating TC itself.

A non-empty set \( S \subseteq V - \{r\} \) is called dependent if \( S \) induces at least one edge in \( G \), and let \( \mathcal{D} \) denote the family of such dependent sets (i.e., \( \mathcal{D} = \{S \subseteq V - \{r\} \mid S \) is dependent in \( G \}) \). Then, the characteristic vector \( x_{\tilde{T}} \in \{0, 1\}^\tilde{E} \) of any \( r\text{-tc} \) \( \tilde{T} \) satisfies the linear inequality \( x(\delta^-(S)) \geq 1 \) for all dependent sets \( S \subseteq V - \{r\} \), where \( \delta^-(S) = \{(u, v) \in \tilde{E} \mid u \notin S, v \in S\} \), and \( x(\tilde{F}) = \sum_{a \in \tilde{F}} x_{\tilde{\alpha}} \) for \( x \in \mathbb{Q}^\tilde{E} \) and \( \tilde{F} \subseteq \tilde{E} \), because at least one arc of \( \tilde{T} \) must enter \( S \) when an edge exists inside \( S \). Thus, the following LP is a relaxation of \( r\text{-TC} \), and its 0–1 solutions are the ones we will be actually seeking for:

\[
\min \sum_{a \in \tilde{E}} c(\tilde{\alpha}) x_{\tilde{\alpha}} \\
\text{(LP) subject to:} \quad x(\delta^-(S)) \geq 1, \quad \forall S \in \mathcal{D} \\
x_{\tilde{\alpha}} \geq 0, \quad \forall \tilde{\alpha} \in \tilde{E}.
\]

Unlike the algorithms in [2,8], our algorithm also requires the dual of (LP), which can be expressed as follows:

\[
\max \sum_{S \in \mathcal{D}} y_S \\
\text{(D) subject to:} \quad \sum_{S \in \mathcal{D} : \tilde{a} \in \delta^-(S)} y_S \leq c(\tilde{\alpha}), \quad \forall \tilde{\alpha} \in \tilde{E} \\
y_S \geq 0, \quad \forall S \in \mathcal{D}.
\]

For any \( y \in \mathbb{Q}^\mathcal{D} \) feasible in (D), the value \( \sum_{S \in \mathcal{D}} y_S \) of (D) is no larger than the value \( \sum_{a \in \tilde{E}} c(\tilde{\alpha}) x_{\tilde{\alpha}} \) of (LP) for any \( x \in \tilde{E} \) feasible in (LP), due to the (weak) duality theorem of LP, and hence, it lower bounds the cost of an optimal \( r\text{-tc} \) in particular. Suppose that dual feasible \( y \) is determined from any \( r\text{-tc} \) \( \tilde{T}' \) computed by an algorithm in such a way that the cost of \( \tilde{T}' \) is related to the value \( \sum_{S \in \mathcal{D}} y_S \) of (D) for this \( y \), and \( \sum_{\tilde{a} \in \tilde{T}'} c(\tilde{\alpha}) \leq \alpha \sum_{S \in \mathcal{D}} y_S \) for some \( \alpha \in \mathbb{R} \). It then follows that the cost of computed \( \tilde{T}' \) never exceeds the optimal \( r\text{-tc} \) cost by a factor of more than \( \alpha \). As in the primal-dual arguments used for many other approximation algorithms, this will be the way in what follows to ascertain the factor \( \alpha \) approximability of subjected algorithms.
3. Basic approximation techniques

In our algorithm design we use the following primal-dual techniques; two for approximating unweighted tree cover, and one for exactly computing minimum weight vertex cover in bipartite graphs. For the first two, let \( \hat{G} = (V, \hat{E}) \) and \( r \in V \) be an instance of unweighted \( r \)-TC (i.e., \( c(\hat{a}) = 1, \forall \hat{a} \in \hat{E} \)), and \( T \) be a spanning tree of \( \hat{G} \) rooted at \( r \). Also let \( e \) denote the undirected edge \( \{u, v\} \in E \) for arc \( \hat{e} = (u, v) \in \hat{E} \), for the rest of the paper.

**PATH.** Suppose \( \hat{T} \) is a depth-first search (dfs) tree. Remove all the leaves from \( \hat{T} \), and the resulting tree \( \hat{T}' \) is an \( r \)-tc for \( \hat{G} \) since no edge exists in \( \hat{G} \) between any two leaves of \( \hat{T} \). This algorithm of Savage was shown to be a factor 2 approximation using the counting argument [9]. It is also possible to assert the 2-approximability by the primal-dual argument as follows. For each non-leaf \( u \) of \( \hat{T} \) with \( u \neq r \), choose one child \( v \) of \( u \), and mark the arc \( (u, v) \) “chosen”. Let \( \hat{P} \) be the set of these chosen arcs. Then, \( \hat{P} \) consists of disjoint dipaths \( \hat{P}_i \)'s, each starting at some non-leaf and ending at some leaf of \( \hat{T} \). Assign 1/2 to each of \( y_e \) for \( \hat{e} \in \hat{P} \). Then, while \( y \in \mathbb{Q}^{\hat{E}} \) is dual feasible in \( (D) \), it can be seen that \( |\hat{T}'| = 2 \sum_{e \in \hat{E}} y_e \) since \( |\hat{T}'| = |\hat{P}| \). As \( \sum_{e \in \hat{E}} y_e \) lower bounds the cost of an optimal \( r \)-tc, the 2-approximability of \( |\hat{T}'| \) thus follows.

**MATCH.** For a not necessarily dfs tree \( \hat{T} \), construct a set \( \hat{P} \) of disjoint dipaths as above, and label all the arcs in each dipath \( \hat{P}_i \) with “1” and “0” alternatively, starting with “1” at the first arc, continuing with “0” at the second, and so on. Let \( M_T = \{ e \in E | \hat{e} \text{ is labeled “1”} \} \) be a matching in \( T \) consisting of all the edges such that the corresponding arcs are labeled “1”. For the set \( L' \) of leaves of \( \hat{T} \) left unmatched by \( M_T \), consider the subgraph \( G[L'] \) of \( G \) induced by \( L' \). Then, \( G[L'] \) may contain some edges of \( G \) since \( \hat{T} \) may not be a dfs tree. For any “maximal” matching \( M_{L'} \) in \( G[L'] \), remove all such leaves of \( \hat{T} \) that are unmatched by both \( M_T \) and \( M_{L'} \), and let \( \hat{T}' \) be the resulting tree. Then, \( \hat{T}' \) is an \( r \)-tc for \( \hat{G} \) since any edge joining leaves of \( \hat{T} \) is covered by some leaf of \( \hat{T} \) matched by either \( M_T \) or \( M_{L'} \).

Setting \( y_e = 1 \) for each \( e \in M_T \cup M_{L'} \), \( y \) can be seen dual feasible since \( M_T \cup M_{L'} \) is a matching in \( G \). It is also easy to observe that \( |\hat{T}'| = 2 \sum_{e \in \hat{E}} y_e \); with each \( e \in M_T \) associate the two arcs, \( \hat{e} \in \hat{T} \) and the one preceding \( \hat{e} \) in \( \hat{T} \), and with each \( e = \{u, v\} \in M_{L'} \) associate the two arcs of \( \hat{T} \) entering \( \{u, v\} \). The 2-approximability thus follows as in **PATH.**

Suppose that it is actually the case in the above that every arc not in \( \hat{T} \) has a weight of at least 1 while \( c(\hat{a}) = 1 \) for \( \hat{a} \in \hat{T} \). It will be useful to notice that \( y \) given above then remains dual feasible under either type of the dual assignments.

**B-VC.** Suppose that \( G = (A \cup B, E) \) is a bipartite graph and each vertex is associated with a nonnegative weight \( b : A \cup B \rightarrow \mathbb{Q}_+ \). A \( b \)-matching for \( G \) is a function \( z : E \rightarrow \mathbb{Q}_+ \) such that \( z(\delta(u)) \leq b(u) \) for each vertex \( u \) in \( G \), where \( \delta(u) \) is the set of edges incident with \( u \). Call the sum of the entries in a \( b \)-matching \( z \) (i.e., \( \sum_{e \in E} z(e) \)) as its size. Then it is a well-known fact (cf. Egerváry’s theorem; see, e.g., [6]) that the maximum size of a \( b \)-matching is equal to the minimum weight of a vertex cover in any bipartite graph. Moreover, such an optimal vertex cover and an optimal \( b \)-matching can be found by a max flow computation.

4. Algorithm

Assume without loss of generality in what follows that edge weights are either 1 or \( w \geq 2 \), and call an arc of weight 1 (an arc of weight \( w \), respectively) as \( 1 \)-arc (\( w \)-arc, respectively). Our algorithm works in two stages. First it computes a (minimum) spanning tree \( T \) of given \( \hat{G} \). Then, it prunes certain leaves of \( T \), and to determine which leaves are to be pruned, it computes a vertex cover of the graph induced by the leaves of \( T \). By removing all the leaves of \( T \) excluded from this vertex cover, the resulting tree \( T' \) is output as an \( r \)-tc for \( \hat{G} \). The first stage starts with:

1. Construct a maximal forest \( \mathcal{F}_1 \) in \( \hat{G} = (V, \hat{E}) \) consisting of 1-arcs only; call a tree in \( \mathcal{F}_1 \) a 1-tree.
2. Shrink each 1-tree into a vertex (naming anew the vertex as \( r \) if original \( r \) was shrunked into it), and compute a dfs spanning tree \( \hat{T}_w \) rooted at \( r \) in the resulting graph.
3. For each 1-tree \( \hat{T}_i \) let \( r_i \) be the (unique) vertex in \( \hat{T}_i \) having an incoming arc of \( \hat{T}_w \) (or let \( r_i = r \) if it contains \( r \)). Redirect all the arcs of each \( \hat{T}_i \) so that it becomes a directed tree rooted at \( r_i \).
4. Construct a tree \( \hat{T} \) spanning in \( \hat{G} \) by gluing together \( \hat{T}_w \) and all 1-trees \( \hat{T}_i \)'s in \( \mathcal{F}_1 \).
By applying MATCH of the previous section to each 1-tree $\tilde{T}_i$, obtain a matching $M_{T_i}$ within it. Let $L'_1 \subseteq L_1$ be the set of all such 1-leaves of $\tilde{T}$ that are not matched by any of $M_{T_i}$'s (i.e., $L'_1 = L_1 - V(\bigcup_i M_{T_i})$).

Construct a maximal matching $M_{L'}$ in $G[L'_1]$, the subgraph of $G$ induced by $L'_1$, and let $L''_1 \subseteq L'_1$ be the set of leaves in $L'_1$ left unmatched by $M_{L'}$ (i.e., $L''_1 = L'_1 - V(M_{L'}) = L_1 - V(\bigcup_i M_{T_i} \cup M_{L'})$).

Consider $G[L''_1 \cup L_w]$. As (1) $L''_1$ is the set of the vertices left unmatched by a maximal matching ($M_{L'}$) in an induced subgraph ($G[L']$) of $G$, and (2) $L_w$ consists of such vertices that are leaves in a dfs tree $\tilde{T}_w$, each of $L''_1$ and $L_w$ is an independent set in $G$, and hence, $G[L''_1 \cup L_w]$ is bipartite. Apply B-VC to $G[L''_1 \cup L_w]$ with $b(u) = 1$ ($\forall u \in L''_1$) and $b(u) = w$ ($\forall u \in L_w$), and compute a minimum weight vertex cover $L_{vc}$ and a maximum $b$-matching $z \in \mathbb{Q}[E[L''_1 \cup L_w]]$.

Prune all such leaves of $\tilde{T}$ that are excluded from $L_{vc}$ in Step (7), i.e., those in $(L''_1 \cup L_w) - L_{vc} = (L_1 \cup L_w) - V(\bigcup_i M_{T_i} \cup M_{L'}) - L_{vc}$, and output the resulting tree $\tilde{T}'$.

Because the set of leaves $(V(\bigcup_i M_{T_i}) \cap L_1) \cup V(M_{L'}) \cup L_{vc}$ retained by the end of the algorithm can be seen, from the way of its construction, to constitute a vertex cover for $G[L_1 \cup L_w]$, the subgraph of $G$ induced by all the leaves of $\tilde{T}$, the set of the vertices $V(T')$ is a vertex cover for $G$. It follows that $\tilde{T}'$ thus constructed is indeed an r-tc for $G$.

**Example.** Fig. 1 shows an example run of the algorithm. In (a) a minimum spanning tree $T$ (solid edges) is computed by Steps 1–4, and it contains two 1-trees, $T_1$ and $T_2$. Here, 1-leaves are colored gray, and $w$-leaves white. Next, Steps 5–7 compute matchings (thickened edges) as shown in (b). Here, $M_{T_1} = \emptyset$, and $M_{T_2}$ consists of those two solid edges within $T_2$. Only one edge is in $M_{L'}$, and it is the dashed one connecting those 1-leaves, one in $T_1$ and the other in $T_2$. The maximum $b$-matching corresponds here to the remaining two edges (thickened and dashed), each connecting a 1-leaf with a $w$-leaf, and each of them has a $z$-value of 1. The minimum vertex cover $L_{vc}$ is then formed by those two 1-leaves incident with these two edges of the $b$-matching. Lastly, Step 8 prunes all the $w$-leaves (there are two of them) in (c), as they were excluded from $L_{vc}$ in Step 7, and outputs $T'$ (solid edges).

**Running time.** In execution of the algorithm above, the most time consuming part is Step (7), where a maximum $b$-matching (or a minimum vertex cover) is computed in a bipartite graph, and it can be carried out by computing a max flow on the (essentially) same graph. Letting $M(n, m)$ denote time complexity of computing an $s - t$ max flow in a network with $n$ nodes and $m$ arcs, the algorithm for r-TC thus runs in time $O(M(n, m))$, and TC can be (approximately) computed in time $O(nM(n, m))$ (by choosing every vertex in turn as a root $r$), where $M(n, m) = O(nm \log(n^2/m))$ when, for instance, the Goldberg–Tarjan’s algorithm is used [5].
5. Analysis

It will be shown first, using the following rules (1) through (7), which dual variables \( y_e \) are assigned with what positive values. By applications of MATCH to \( \tilde{T}_i \)'s and B-VC to \( G[L'_1 \cup L_w] \), the dual variables of form \( y_e \) with \( e \in E \) are set as follows:

1. \( y_e = 1 \) if \( e \in M_{\tilde{T}_i} \) (within each \( \tilde{T}_i \)),
2. \( y_e = 1 \) if \( e \in M_{L'} \) (within \( G[L'_1] \)), and
3. \( y_e = z_e/2 \) for every edge \( e \) of \( G[L'_1 \cup L_w] \), where \( z \) is a maximum \( b \)-matching in \( G[L'_1 \cup L_w] \) with \( b(u) = 1, \forall u \in L'_1 \).

Distinguish the arcs \( \tilde{a} \) of a tree as either leaf or non-leaf according to whether \( \tilde{a} \) is incident with a leaf or not. We also call a vertex \( u \) of \( \tilde{T}_w \) an s-node if some 1-tree is shrunk into it, and thus, \( u \) corresponds to \( V(T_i) \) in \( \tilde{T} \) for some 1-tree \( T_i \).

To account for the weights of the non-leaf \( w \)-arcs of \( \tilde{T}_w \) as well as such leaf arcs of \( \tilde{T}_i \)'s that are glued directly with \( w \)-arcs of \( \tilde{T}_w \) (thus they are not leaf arcs in \( \tilde{T} \)), we use \( \tilde{T}_w \) and simulate the dual assignment of PATH on it up to a certain extent. Notice that no edge \( e \in T \) of cost \( w \) is yet given any dual value since \( y_{u,v} \) has been so far assigned with some positive value (according to rules (1)-(3)) only for the case of either \( \{u, v\} \subseteq V(T_i) \) for some 1-tree \( T_i \), or \( \{u, v\} \subseteq L'_1 \cup L_w \). To raise such \( y_e \)-values, we need to know to which arc of \( \tilde{T} \) an arc \( \tilde{e} \) of \( \tilde{T}_w \) is mapped, when \( \tilde{e} \) is incident with an s-node (as an s-node is actually a set of vertices in \( \tilde{T} \). We denote by \((u_T, v_T)\) the \( w \)-arc of \( \tilde{T} \) corresponding to an arc \((u, v)\) of \( \tilde{T}_w \). So, while \( u_T = u \) if \( u \) is not an s-node, if \( u \) is an s-node and a 1-tree \( T_i \) was shrunk into \( u \), \( u_T \) is the vertex of \( \tilde{T}_i \) (and \( \tilde{T} \)) to which \((u, v)\) of \( \tilde{T}_w \) was directly glued (and similarly for \( v \) and \( v_T \)).

We will set up additional rules for dual assignments based on the following observations:

- Let \((u, v)\) be an arc of \( \tilde{T}_w \) chosen into \( \tilde{P} \), and \( \tilde{e} \) be the arc \((u_T, v_T)\) of \( \tilde{T} \) corresponding to \((u, v)\). If neither \( u \) nor \( v \) is an s-node, there is no problem setting \( y_e = w/2 \) (as in PATH).

Notice that, if \( v \) is an s-node and \( v_T \) belongs to some 1-tree \( T_i \), \( v_T \) coincides with the root \( r_i \) of \( T_i \). Moreover, any root \( r_j \) of a 1-tree is left unmatched by any matchings, \( M_{\tilde{T}_i} \)'s and \( M_{L'} \), or by the \( b \)-matching \( z \). Therefore, even if \( v \) is an s-node, we may set \( y_e = 1 \) if \( u \) is not an s-node.

- Suppose \( \tilde{e}' \) is a leaf arc of some 1-tree \( \tilde{T}_i \), and it is directly glued with \( \tilde{T}_w \). Let \( D_{\tilde{e}'} \) denote the set of all those arcs in \( \tilde{T} \) immediately following \( \tilde{e}' \) (thus, they all correspond to some arcs of \( \tilde{T}_w \)). If \( e' \in M_{\tilde{T}_i} \), no positive value can be assigned in general to \( y_e \) for any \( \tilde{e} \in D_{\tilde{e}'} \), but if \( e' \notin M_{\tilde{T}_i} \), it is safe to pick one arc \( \tilde{e} \) from \( D_{\tilde{e}'} \) and to give 1 to \( y_e \). In doing so, moreover, we don’t have to care whether \((u, v)\) of \( \tilde{T}_w \) corresponding to \( \tilde{e} \) is a chosen arc or not, nor whether \( v \) is an s-node or not.

In a summary we use the following rules for dual assignments on the edges \( e \in T \) of cost \( w \):

- For each \((u, v)\) of \( \tilde{T}_w \) chosen into \( \tilde{P} \), let \( \tilde{e} = (u_T, v_T) \) of \( \tilde{T} \) and set
  (4) \( y_e = w/2 \) (as in PATH) if neither \( u \) nor \( v \) is an s-node, and
  (4) \( y_e = 1 \) if \( v \) is an s-node but \( u \) is not.
- For each leaf arc \( \tilde{e}' \) of any 1-tree \( \tilde{T}_i \), such that \( e' \notin M_{\tilde{T}_i} \) and \( \tilde{e}' \) is directly glued with \( \tilde{T}_w \),
  (6) pick one arc \( \tilde{e} \) from \( D_{\tilde{e}'} \), and set \( y_e = 1 \).

It is also important to note here that, due to the preconditions associated with rules (4) through (6), at most one of them applies to any \( w \)-arc \( \tilde{e} \) of \( \tilde{T} \) (or to any arc \((u, v)\) of \( \tilde{T}_w \)).

All the dependent sets given so far by the rules above ((1) through (6)) are of size 2 (i.e., they are edges), and the next lemma is concerned with the sum of such duals only, assigned on the edges around a vertex of \( G \):

**Lemma 1.** The dual variables \( y_e \)’s determined by rules (1)-(6) satisfy that

\[
\sum_{e \in \delta(v)} y_e \leq \begin{cases} 
1 & \text{if } v \in V(\tilde{T}_k) \text{ for some } 1 \text{-tree } \tilde{T}_k \\
0 & \text{otherwise}
\end{cases}
\]

for each \( v \in V \).
Proof. Case \( v \in V(\overline{T}_k) \) for some 1-tree \( \overline{T}_k \). Suppose \( v \in L''_1 \). Then \( v \) is free from \( M_{T_i}'s, M_{L_i}'s \), or any \( w \)-arc of \( \overline{T} \), and only rule (3) is relevant here. So, \( \sum_{e \in \delta(v)} y_e = \sum_{e \in \delta(v)} z_e/2 \leq 1/2 \) as \( b(v) = 1 \) for the maximum \( b \)-matching \( z \). If \( v \not\in L''_1 \) on the other hand, at most one \( e' \) of those edges in \( \delta(v) \) receives a nonzero value, by the way the dual values are set in \( y \), and such \( e' \) is either from \( M_{T_i} \) or \( M_{L_i} \), or otherwise, it is an edge of \( T_w \). According to rules (1) and (2) in the former case, and to rules (5) and (6) in the latter case, \( y_{e'} \leq 1 \) in every case, and it follows that \( \sum_{e \in \delta(v)} y_e \leq 1 \).

Case \( v \not\in V(\overline{T}_i) \) for any 1-tree \( \overline{T}_i \).

Case \( v \) is a non-leaf of \( \overline{T} \). Then, only those rules (4)–(6) are relevant to the edges in \( \delta(v) \), and hence, only the tree edges among them, i.e., those in \( \delta(v) \cap T \), can possibly carry some positive duals. Certainly, there can be at most one arc of \( \overline{T} \) entering \( v \). Since \( v \) does not belong to any 1-tree, rule (6) cannot assign a dual to any of those arcs leaving \( v \). Meanwhile, rules (4) and (5) can together assign only to a single arc among them, namely to the chosen one. Therefore, at most two among those in \( \delta(v) \) can carry positive duals, each of which is at most \( w/2 \). It thus follows that \( \sum_{e \in \delta(v)} y_e \leq w \).

Case \( v \) is a leaf of \( \overline{T} \). So \( v \) is a \( w \)-leaf. Only a single edge of \( T \) (or of \( T_w \)), say \( e' \), is incident with \( v \), and it can carry at most \( w/2 \) as its dual. All the other edges in \( \delta(v) \) with positive duals come from the \( b \)-matching \( z \). Since \( y_e = z_e/2 \) within \( G[L''_1 \cup L_w] \) and \( \sum_{e \in \delta(y)} z_e \leq b(v) = w \), the total contribution of these \( y_e \)'s from \( z \) is at most \( w/2 \). Therefore, together with \( y_{e'} \), we have \( \sum_{e \in \delta(v)} y_e \leq w \). \( \square \)

Recall now that the dual variables correspond in general to any dependent sets, not only to edges. In fact the use of non-edge dependent sets can be shown crucial in our analysis, and the dual variables \( y_S \) with such dependent sets \( S \) will be subjected in the last rule for dual assignments as follows:

(7) Set \( y_{V(\overline{T}_i)} = w - 1 \) for any 1-tree \( \overline{T}_i \) if \( r \not\in V(\overline{T}_i) \).

Clearly, rule (7) does not conflict with any of the previous ones if \( |V(\overline{T}_i)| \geq 3 \), and even if \( |V(\overline{T}_i)| = 2 \), since the unique edge of \( T_i \) cannot be matched by \( M_{T_i} \), the set \( V(\overline{T}_i) \), which coincides with the unique edge of \( T_i \), gets assigned only once by rule (7).

Lemma 2. The vector \( y \) of dual variables determined as above is feasible in (D).

Proof. We need to show that the inequality

\[
\sum_{S \in D; a \in \delta^-(S)} y_S = \sum_{S \in D; a \not\in S, v \in S} y_S \leq c(\overline{a})
\]

holds for any \( \overline{a} = (u, v) \in \overline{E} \).

Recall that rules (1) through (6) assign positive duals only to such dependent sets that coincide with edges in \( G \). Any dependent set assigned by (7), on the other hand, is of form \( V(\overline{T}_i) \) for some 1-tree \( \overline{T}_i \), and \( y_S \) with such \( S = V(\overline{T}_i) \) occurs in the summation \( \sum_{S \in D; a \in \delta^-(S)} y_S \) above if and only if \( \overline{a} \) enters into \( V(\overline{T}_i) \) for some 1-tree \( \overline{T}_k \):

Case \( \overline{a} = (u, v) \in \delta^-(V(\overline{T}_k)) \) for some 1-tree \( V(\overline{T}_k) \). Then, \( c(\overline{a}) = w \).

For any \( \overline{T}_k \) different from \( \overline{T}_k \), the value of \( y_{V(\overline{T}_k)} \) set by rule (7) is irrelevant here, and hence,

\[
\sum_{S \in D; a \not\in \delta^-(S)} y_S \leq y_{V(\overline{T}_k)} + \sum_{e \in \delta(v) - a} y_e \leq y_{V(\overline{T}_k)} + \sum_{e \in \delta(v)} y_e.
\]

Since \( y_{V(\overline{T}_k)} \leq w - 1 \) and \( \sum_{e \in \delta(v)} y_e \leq 1 \) when \( v \) belongs to a 1-tree, according to Lemma 1, we have that \( \sum_{S \in D; a \in \delta^-(S)} y_S \leq (w - 1) + 1 = c(\overline{a}) \).

Case \( \overline{a} = (u, v) \not\in \delta^-(V(\overline{T}_i)) \). Any dual \( y_{V(\overline{T}_i)} \) set by rule (7) is irrelevant here, and any positive \( y_S \) with \( \overline{a} \in \delta^-(S) \) is assigned by one of rules (1) through (6). So, such \( S \) is actually an edge in \( G \), and we may write \( \sum_{S \in D; a \in \delta^-(S)} y_S = \sum_{e \in \delta(v) - a} y_e \).

Case \( v \in V(\overline{T}_i) \) for some 1-tree \( \overline{T}_k \). Then, \( \sum_{e \in \delta(v)} y_e \leq 1 \) according to Lemma 1, and hence, \( \sum_{e \in \delta(v) - a} y_e \leq \sum_{e \in \delta(v)} y_e \leq c(\overline{a}) \), whether \( c(\overline{a}) = 1 \) or \( = w \).

Case \( v \not\in V(\overline{T}_i) \) for any 1-tree \( \overline{T}_i \). Then, \( c(\overline{a}) = w \) and, as \( \sum_{e \in \delta(v)} y_e \leq w \) by Lemma 1, \( \sum_{e \in \delta(v) - a} y_e \leq \sum_{e \in \delta(v)} y_e \leq c(\overline{a}) \). \( \square \)
This completes the proof of the dual feasibility of $y$, and it remains to show that the cost of an $r$-tc computed is no larger than twice the value of $y$.

**Lemma 3.** For $r$-tc $\tilde{T}'$ output by the algorithm and the dual variables $y$ determined as above, $\sum_{\tilde{a} \in \tilde{T}'} c(\tilde{a}) \leq 2 \sum_{s \in D} y_s$.

**Proof.** Recall MATCH, and consider the dual values assigned to the edges in the matchings $M_{T_1}$'s and $M_{L'}$. As was done in MATCH, these $y$-values can be used to account for the costs of all the non-leaf arcs of $\tilde{T}_1$'s plus such leaf arcs of $\tilde{T}_1$'s that are incident to the 1-leaves in $L_1 - L_1'$. It should be noted here, however, that some leaf arcs $(u, v)$ of $\tilde{T}_1$'s may not be leaf arcs in $\tilde{T}$ (because they are glued directly with $\tilde{T}_w$). If so, such $v$ is not taken into the set $L_1$ of 1-leaves of $\tilde{T}$, and the cost of $(u, v)$ is not accounted for by these $y$-values unless $(u, v)$ itself is in $M_{T_1}$.

Recall B-VC next, and consider the dual values assigned to the edges in $G[L_1' \cup L_w]$ as $y_e = z_e/2$ for the maximum $b$-matching $z$. Then, as was done in B-VC, these $y$-values can account for the costs of such leaf arcs of $\tilde{T}$ that are incident to the (1- or $w$-) leaves in $L_{vc}$.

By now, the costs of all the leaf arcs of $\tilde{T}'$ have been taken into account, using the duals on the edges of $M_{T_1}$'s, $M_{L'}$, and $b$-matching $z$, assigned by rules (1)–(3). What remains to be accounted for, therefore, are (i) the costs of those 1-arcs indicated above, and (ii) the costs of non-leaf $w$-arcs in $\tilde{T}'$. In what follows, we will spend all the duals assigned by rule (6) to account for the costs in (i), whereas all the duals assigned by rules (4), (5), and (7) will be used for those in (ii):

(i) For any 1-arc $\tilde{e}' \in \tilde{T}'$ its cost is not yet accounted for if and only if $\tilde{e}'$ is a leaf arc of some 1-tree $\tilde{T}_1$, $\tilde{e}' \notin M_{T_1}$, and $\tilde{e}'$ is directly glued to a $w$-arc of $\tilde{T}_w$. Then, $\tilde{e}'$ satisfies exactly the preconditions for application of rule (6). So, some $\tilde{e} \in D_{\tilde{e}'}$ will be picked, and $y_{\tilde{e}'}$ will be set $= 1$. Moreover, the value of this $y_{\tilde{e}'}$ can be used exclusively for (accounting for the cost of) $\tilde{e}'$, as $D_{\tilde{e}'}$ and $D_{\tilde{e}''}$ cannot share an arc of $\tilde{T}$ for different $\tilde{e}'$ and $\tilde{e}''$. Therefore, the total of dual values assigned by rule (6) is sufficient for the total cost of 1-arcs left unaccounted for.

(ii) Recall PATH and consider a path $\tilde{P}_1 \subseteq \tilde{P}$ in $\tilde{T}_w$. Along this path some of the dual values $y_{\tilde{e}'}$ on arc $\tilde{e} = (u, v) \in \tilde{P}_1$, might be already used up in (i) if $u$ is an $s$-node (rule (6)). Therefore, so as to no longer take into account such duals, let us suppose that the duals still available for us are those given in the following forms:

$$y_{\tilde{e}} = \begin{cases} w/2 & \text{if neither } u \text{ nor } v \text{ is an } s \text{-node (rule (4))} \\ 0 & \text{otherwise (i.e., if } u \text{ is an } s\text{-node) } \end{cases}$$

for each $\tilde{e} = (u, v)$ in $\tilde{P}_1$. Moreover, if a vertex $u$ ($\neq r$) on $\tilde{P}_1$ is an $s$-node and shrunk from some $\tilde{T}_1$, the value of $y_{v(u,1)} = w - 1$ (rule (7)) remains untouched yet. So we may use such $y$-values here, and represent them by $y_u = w - 1$ if $u$ is an $s$-node.

Let $v$ be an $s$-node on $\tilde{P}_1$, and $(u, v)$ and $(v, w)$ be the arcs of $\tilde{P}_1$ entering and leaving $v$, respectively. Then, while $y_{[v,u]} = 0$, the value of $y_{[u,v]}$ depends on whether $u$ is an $s$-node or not; it is $= 0$ if $u$ is an $s$-node, and otherwise, it is $= 1$. Let us redistribute the “dual” values of $y_u$'s for s-nodes $v$ as follows. If $y_{[u,v]} = 0$ (i.e., $u$ is an $s$-node), then allocate $w/2 (\leq w - 1)$ of $y_v$ to $y_{[u,v]}$. If $y_{[u,v]} = 1$ (i.e., $u$ is not an $s$-node), on the other hand, using this and $w - 1$ of $y_u$, reassign $w/2$ to each of $y_{[u,v]}$ and $y_{[v,u]}$. Then, $y_{[v,u]}$ always receives $w/2$ in general if $v$ is an $s$-node, and it can be verified that, after redistributing all the dual values $y_u$ at $s$-nodes $u$ on the paths in $\tilde{P}$ in this manner, every arc $\tilde{e} \in P$ receives $w/2$ as its dual value. (Note: If $(u, v)$ does not exist when redistributing the value of $y_v$, simply ignore the allocation to it. If $(u, v)$ does not exist, on the other hand, because $v$ is the last end of a path, save the allocation to it, and use this dual in accounting for the cost of $(u, v)$; we need this, unlike in PATH, as $(u, v)$ is not a leaf arc of $\tilde{T}$ if $v$ is an $s$-node, although $v$ is a leaf of $\tilde{T}_w$.) Therefore, we may use the same argument as used in PATH to account for the costs of all the internal arcs of $\tilde{T}_w$, i.e., the costs of non-leaf $w$-arcs of $\tilde{T}'$. \[ \square \]

**Theorem 4.** The algorithm of Section 4 computes an $r$-tc $\tilde{T}'$ such that its cost $c(\tilde{T}') = \sum_{\tilde{a} \in \tilde{T}'} c(\tilde{a})$ is no larger than twice the optimal cost. Therefore, our algorithm approximates $r$-TC (and hence, TC) within a factor of 2.

**Proof.** As explained in the paragraph following the algorithm description, the algorithm computes an $r$-tc $\tilde{T}'$ for any given $G$. 
By Lemma 2, y is dual feasible, and hence, its value $\sum_{S \in D} y_S$ in (D) lower bounds the cost of an optimal r-tc by virtue of the (weak) duality theorem of LP. The cost of $T'$ on the other hand is at most twice the value of y ($=2 \sum_{S \in D} y_S$) by Lemma 3; thus, it is no larger than twice the optimal cost. $\square$

6. Final remarks

We have considered the tree cover problem with two edge weights $w_1$ and $w_2$ satisfying that $w_2 \geq 2w_1$. Generalizing the result so far known only for the unweighted version, it was shown possible to efficiently approximate this version of tree cover within a factor of 2. While related problems of vertex cover or edge dominating set are known approximable within the same constant factor, whether weighted or not, this is the first case of tree cover with non-uniform weights found to be approximable as good as the case of uniform weights. For further generalization, whereas the techniques employed in the current approach might be possibly extendable to the case of more distinctive weights, it is not clear at this point how to deal even with the case of two edge weights being closer to each other, not differing by a factor of 2.

Acknowledgement

The second author was supported in part by a Grant in Aid for Scientific Research of the Ministry of Education, Science, Sports and Culture of Japan.

References