RAIRO Operations Research
RAIRO Oper. Res. **37** (2003) 179-194
DOI: 10.1051/ro:2003020

CONSTRAINED STEINER TREES IN HALIN GRAPHS *

Guangting Chen[1] and Rainer E. Burkard[2]

Abstract. In this paper, we study the problem of computing a minimum cost Steiner tree subject to a weight constraint in a Halin graph where each edge has a nonnegative integer cost and a nonnegative integer weight. We prove the NP-hardness of this problem and present a fully polynomial time approximation scheme for this NP-hard problem.

Keywords. Steiner trees, Halin graph, approximation scheme.

1. Introduction

A computer network is often modeled by an undirected graph $G = (V, E)$ where V is the set of vertices and E is the set of edges. The traditional Steiner tree problem associates an edge cost $c(e) \geq 0$ with each edge $e \in E$ and asks for a minimum cost subgraph of G spanning a given subset $S \subseteq V$ of target vertices. Such problems find important applications in computer networks and have been studied by many researchers [5–7, 9, 10].

In this traditional network model, there is only one cost for each edge. In real-life problems, each edge may have a weight besides a cost, and we want to find a minimum cost subgraph of G spanning the vertices in S with a total weight

Received February 28, 2003.

* *This research has been supported by the Spezialforschungsbereich F 003 "Optimierung und Kontrolle", Projektbereich Diskrete Optimierung and by Chinese National Science Foundation.*

[1] The School of Science, Hangzhou Institute of Electronics Engineering, Hangzhou 310037, P.R. China; e-mail: gtchen@sci.hziee.edu.cn

[2] Technische Universität Graz, Institut für Mathematik, Steyrergasse 30, A-8010 Graz, Austria; e-mail: burkard@tugraz.at

no more than a given weight constraint W. We call this problem the *Weight Constrained Minimum Cost Steiner Tree Problem (WCSTP).*

Formally, we generalize the traditional network model to allow two independent edge weighting functions: with each edge $e \in E$, there is an associated integer cost $c(e) \geq 0$ and an associated integer weight $w(e) \geq 0$. Let G' be a subgraph of G. The cost (or weight) of G', denoted by $c(G')$ (or $w(G')$), is the sum of the edge costs (or weights) of G'. Given a set of target vertices $S \subseteq V$ and an integer weight $W \geq 0$, we are interested in computing a minimum cost tree subgraph T of G which spans the vertices in S subject to the constraint that $w(T) \leq W$. We call such a tree a *weight constrained minimum cost Steiner tree.*

It is easy to see that the $WCSTP$ is NP-hard in the strong sense. In this paper, we are interested in a $WCSTP$ on an important class of networks known as *Halin graphs.* A Halin graph H_0 is obtained by a planar embedding of a tree T_0 having no vertices of degree 2, and then by adding new edges between the consecutive leaves of T_0 to form a cycle C_0 (the order of leaves is determined by the embedding of T_0) [9,10]. We write $H_0 = T_0 \cup C_0$.

Halin graphs are nontrivial generalizations of tree and ring networks, since a Halin graph is obtained by connecting the leaves of a tree by a cycle. Whereas Winter [9] showed that the Steiner tree problem on a Halin graph is solvable by a linear time algorithm, we show in this paper that the weight constrained minimum cost Steiner tree problem is NP-hard even when the graph is a Halin graph. We then present a fully polynomial time approximation scheme for computing a weight constrained minimum cost Steiner tree in a Halin graph.

2. $WCSTP$ in Halin graphs is intractable

When there are only two vertices in S, the $WCSTP$ becomes the well-known weight constrained shortest path problem ($WCSPP$) which is known to be NP-hard [3]. To the best of our knowledge, no one has ever addressed the complexity of $WCSTP$ or $WCSPP$ in a Halin graph specifically. We will present the NP-hardness of $WCSPP$ in a Halin graph at first. Then we get that $WCSTP$ in a Halin graph is also NP-hard, since $WCSTP$ is more general than $WCSPP$.

Theorem 2.1. *The $WCSPP$ in a Halin graph is NP-hard.*

Proof. We transform PARTITION to the $WCSPP$ in Halin graphs.

Suppose we have a set A with n elements and every element a has a corresponding size $s(a) \in \mathbf{Z}^+$, $\sum_{a \in A} s(a) = 2B$. We construct a graph as follows.

- $V = \{v_1, v_2, \ldots, v_{n+1}, u_2, u_3, \ldots, u_n, x_2, x_3, \ldots, x_n\}$.
- $E = \{(v_i, v_{i+1}), (u_i, u_{i+1}), (x_i, x_{i+1}) | i = 1, 2, \ldots, n\} \cup \{(v_i, u_i), (u_i, x_i) | i = 2, \ldots, n\}$, where we suppose that $x_1 = u_1 = v_1, x_{n+1} = u_{n+1} = v_{n+1}$.
- For $e = (v_i, v_{i+1}), c(e) = s(a_i), w(e) = 0$.
- For $e = (u_i, u_{i+1}), c(e) = 0, w(e) = s(a_i)$.
- For $e = (u_i, v_i), c(e) = 0, w(e) = 0$.
- For the other edges, $c(e) = w(e) = 2B$.

Obviously, graph H is a Halin graph. For this Halin graph, we want to find a shortest path from v_1 to v_{n+1} such that the total weight of this path is no more than B.

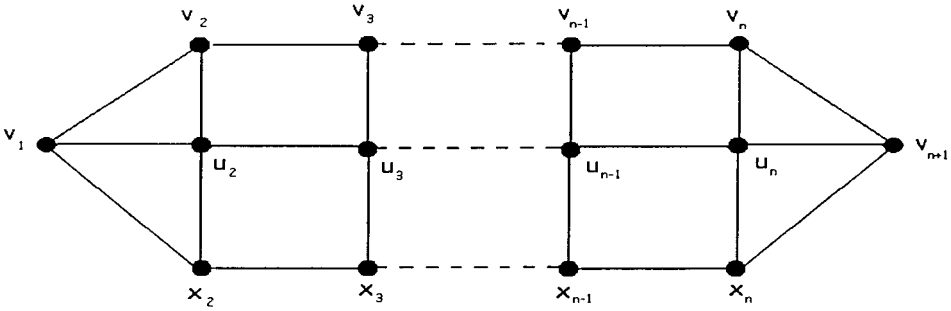

FIGURE 1. A graph H constructed from a PARTITION problem.

If we can find a partition of $A = A_1 \cup A_2$ such that $\sum_{a \in A_1} s(a) = \sum_{a \in A_2} s(a) = B$, then we have a path from v_1 to v_{n+1} consisting of the edges (v_i, v_{i+1}) for $a_i \in A_1, (u_i, u_{i+1})$ for $a_i \in A_2$ and some necessary edges (v_i, u_i). The total cost and total weight of this path are all equal to B. This path is obviously the shortest path under the restriction of a total weight no more than B.

Conversely, assume that we have found a shortest path from v_1 to v_{n+1} with the total weight no more than B. If the weight of this shortest path is just B, then from the construction of this Halin graph, we can get a partition of A by setting $A_1 = \{a_i | (v_i, v_{i+1})$ is in the shortest path$\}$. If the weight of the path is less than B, then the answer to problem PARTITION is negative.

This shows that the $WCSPP$ in Halin graphs is NP-hard. □

Since the weight constrained Steiner tree problem comprises the WCSPP, we get immediately

Theorem 2.2. *The $WCSTP$ in a Halin graph is NP-hard.*

Up to now we considered a Steiner tree problem under the additional constraint that the sum of weights in the Steiner tree is bounded. This problem is NP-hard. If we replace, however, this constraint by a bottleneck constraint of the form that the largest weight of an edge in the Steiner tree is bounded by a given constant M_0, then the problem becomes polynomially solvable. Let us call the latter problem $MAX - WCSTP$. To see this, let G' be the graph obtained by changing the cost of edges in G as follows: if the weight of edge e is more than M_0, then let the cost of this edge become BIG, a very large number. The graph G' is still a Halin graph, and we can compute a minimum cost Steiner tree of G' in $O(n)$ time. If the cost of the minimum cost Steiner tree in G' is smaller than BIG, then this tree is the optimal solution of $MAX - WCSTP$. This shows that the $MAX - WCSTP$ can be solved in $O(n)$ time. By using binary search it is possible to compute in

$O(n \log n)$ time a shortest Steiner tree in a Halin graph such that the maximum weight of an edge in the Steiner tree is as small as possible.

3. A pseudo polynomial time algorithm

In the definition of the $WCSTP$, the weight and cost of an edge are symmetric, we may also speak about the *cost constrained minimum weight Steiner tree problem(CCSTP)* which asks for a minimum weight tree subject to a cost constraint. We will present a pseudo polynomial time algorithm for computing a cost constrained minimum weight Steiner tree in this section, and then apply standard techniques of scaling and rounding to turn the pseudo polynomial time algorithm into a PTAS for $WCSTP$ in the next sections.

3.1. Fan contractions

We are given a Halin graph G with target set $S \subseteq V$ and a nonnegative integer M. We want to find a minimum weight Steiner tree among those whose cost is bounded by M.

Let $G = H_0 = T_0 \cup C_0$, and let i be a nonleaf of T_0 which is adjacent to at most one other nonleaf j of T_0. If no such j exists, then H_0 is a wheel. In this case j may be an arbitrarily chosen neighbour of i. Let Γ_i denote the set of neighbours of i. The subgraph F_i of H_0 induced by $\Gamma_i \cup \{i\} - \{j\}$ is called a fan, and vertex i is called the centre of the fan F_i. It is easy to verify that every Halin graph has at least two fans.

Our algorithm is based on fan contractions. A *fan contraction* of $H_0 = T_0 \cup C_0$ along fan F_i is obtained as follows (see Winter [9]). (i) Attach the edges incident to F_i to the vertex i, (ii) delete the vertices $F_i \cap C_0$ from H_0. The resultant graph, denoted by H_1, is clearly a Halin graph unless H_0 is a wheel. If H_0 is a wheel, H_1 consists of two vertices joined by three parallel edges.

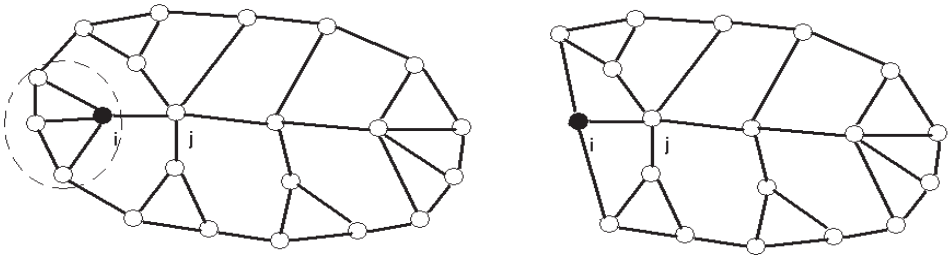

Figure 2. A Halin graph H_0 and its fan contraction H_1.

Let $H_k = T_k \cup C_k$ denote a Halin graph obtained from the original graph $H_0 = T_0 \cup C_0$ by k fan contractions. Let F_i denote a fan of H_k and assume that $1, 2, \ldots, r$ denote the vertices in $F_i \cap C_k$ (in clockwise order). Note that some of these vertices may represent contracted fans. For every p, $1 \leq p \leq r$, let p_f and p_l

denote the first and the last vertex (in clockwise order) of C_0, respectively, which has been contracted into p in H_k. If $p \in C_0$, then $p_f = p = p_l$. Let $G(p)$ denote the subgraph of H_0 induced by all the vertices that have been contracted into p.

For every p, $1 \leq p \leq r$, and every K, $K = 0, 1, \ldots, M$, consider the following subgraphs of $G(p)$, each containing all target vertices of $G(p)$. We are in particular interested in the weights of these subgraphs. If some of the subgraphs do not exist, the corresponding weights are a big number *BIG*.

- $st(p, K)$ is the minimum weight of a Steiner tree with a cost at most K.
- $tf(p, K)$ is the minimum weight of a Steiner tree containing p_f, with a cost at most K.
- $tc(p, K)$ is the minimum weight of a Steiner tree containing p, with a cost at most K.
- $tl(p, K)$ is the minimum weight of a Steiner tree containing p_l, with a cost at most K.
- $fc(p, K)$ is the minimum weight of a Steiner tree containing p_f and p, with a cost at most K.
- $fl(p, K)$ is the minimum weight of a Steiner tree containing p_f and p_l, with a cost at most K.
- $cl(p, K)$ is the minimum weight of a Steiner tree containing p and p_l, with a cost at most K.
- $fcl(p, K)$ is the minimum weight of a Steiner tree containing p_f, p and p_l, with a cost at most K.
- $dfc(p, K)$ is the minimum weight of two disjoint Steiner trees, one containing p_f and the other containing p, with the total cost at most K.
- $dfl(p, K)$ is the minimum weight of two disjoint Steiner trees, one containing p_f and the other containing p_l, with the total cost at most K.
- $dcl(p, K)$ is the minimum weight of two disjoint Steiner trees, one containing p and the other containing p_l, with the total cost at most K.
- $ddfc(p, K)$ is the minimum weight of two disjoint Steiner trees, one containing p_f and p, the other containing p_l, with the total cost at most K.
- $ddfl(p, K)$ is the minimum weight of two disjoint Steiner trees, one containing p_f and p_l, the other containing p, with the total cost at most K.
- $ddcl(p, K)$ is the minimum weight of two disjoint Steiner trees, one containing p and p_l, the other containing p_f, with the total cost at most K.
- $ddd(p, K)$ is the minimum weight of three disjoint Steiner trees containing p_f, p and p_l, respectively, with the total cost at most K.

The corresponding subgraphs will be denoted by $ST(p, K)$, $TF(p, K)$, ..., $DDD(p, K)$. In addition, we define $none(p, K) := 0$, if $G(p)$ does not contain any target vertices, and $none(p, K) :=$*BIG*, otherwise. That means there is a big penalty if $G(p)$ contains any target vertices.

3.2. Recurrence rules

Now we will show how to determine $st(i, K), tf(i, K), \ldots, ddd(i, K), none(i, K)$ in a Halin graph $H_k = T_k \cup C_k$, provided $st(1, K), tf(1, K), \ldots, ddd(1, K)$,

$none(1,K),\ldots,st(r,K),tf(r,K),\ldots,\ ddd(r,K),\ none(r,K)$ are available by initialization or by previous computations, where i is the centre of a fan F_i in H_k, and $1,2,\ldots,r$ are the vertices in $F_i \cap C_k$. First we initialize the variables for each vertex p on the cycle C_0 of the original Halin graph $H_0 = T_0 \cup C_0$, and for every K, $K = 0,1,\ldots,M$, as follows.

$$st(p,K) := tf(p,K) := tc(p,K) := tl(p,K) := fc(p,K) := fl(p,K) := cl(p,K) := fcl(p,K) := 0,$$

$$dfc(p,K) := dfl(p,K) := dcl(p,K) := ddfc(p,K) := ddfl(p,K) := ddcl(p,K) := ddd(p,K) := BIG,$$

$$none(p,K) := \begin{cases} 0 & \text{if } p \notin S, \\ BIG & \text{if } p \in S. \end{cases}$$

Let $G_i(p)$, $1 \le p \le r$, denote a subgraph of H_0 induced by the vertex i together with the vertices of $G(1)$, $G(2)$, ..., $G(p)$. Define $ST_i(p,K)$, $TF_i(p,K)$, ..., $DDD_i(p,K)$ in $G_i(p)$ (and the corresponding weights as well as $none_i(p,K)$) in the same manner as their counterparts in $G(p)$, all occurrences of p_f are replaced by 1_f, and all occurrences of p are replaced by i.

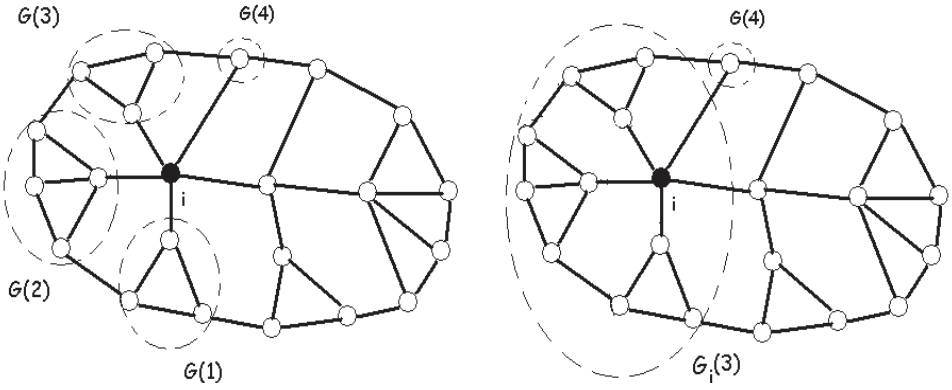

FIGURE 3. $G(p)$ and $G_i(p)$ in a Halin graph.

Let $w(i,1)$ and $c(i,1)$ be the weight and cost of edge $(i,1)$, respectively. The recurrence rules for $st_i(1,K), tf_i(1,K),\ldots,none_i(1,K)$ are as follows.

- $st_i(1,K) := \begin{cases} st(1,K) & \text{if } i \notin S, \\ 0 & \text{if } i \in S \text{ and } none(1,K) = 0, \\ w(i,1)+tc(1,K-c(i,1)) & \text{if } i \in S \text{ and } none(1,K) = BIG. \end{cases}$

- $tf_i(1,K) := \begin{cases} tf(1,K) & \text{if } i \notin S, \\ w(i,1) + fc(1,K-c(i,1)) & \text{if } i \in S. \end{cases}$

- $tc_i(1,K) := \begin{cases} 0 & \text{if } none(1,K) = 0, \\ w(i,1) + tc(1, K - c(i,1)) & \text{if } none(1,K) = BIG. \end{cases}$
- $tl_i(1,K) := \begin{cases} tl(1,K) & \text{if } i \notin S, \\ w(i,1) + cl(1, K - c(i,1)) & \text{if } i \in S. \end{cases}$
- $fc_i(1,K) := w(i,1) + fc(1, K - c(i,1))$.
- $fl_i(1,K) := \begin{cases} fl(1,K) & \text{if } i \notin S, \\ w(i,1) + fcl(1, K - c(i,1)) & \text{if } i \in S. \end{cases}$
- $cl_i(1,K) := w(i,1) + cl(1, K - c(i,1))$.
- $fcl_i(1,K) := w(i,1) + fcl(1, K - c(i,1))$.
- $dfc_i(1,K) := \min\{tf(1,K), w(i,1) + dfc(1, K - c(i,1))\}$.
- $dfl_i(1,K) := \begin{cases} dfl(1,K) & \text{if } i \notin S, \\ w(i,1) + \min\{ddfc(1, K - c(i,1)), ddcl(1, K - c(i,1)\} & \text{if } i \in S. \end{cases}$
- $dcl_i(1,K) := \min\{tl(1,K), w(i,1) + dcl(1, K - c(i,1))\}\cdot$
- $ddfc_i(1,K) := w(i,1) + ddfc(1, K - c(i,1))$.
- $ddfl_i(1,K) := \min\{tfl(1,K), w(i,1) + ddfl(1, K - c(i,1))\}\cdot$
- $ddcl_i(1,K) := w(i,1) + ddcl(1, K - c(i,1))$.
- $ddd_i(1,K) := \min\{dfl(1,K), w(i,1) + ddd(1, K - c(i,1))\}\cdot$
- $none_i(1,K) := \begin{cases} none(1,K) & \text{if } i \notin S, \\ BIG & \text{if } i \in S. \end{cases}$

For some p, $2 \le p \le r$, let $q = p - 1$. Suppose that $st_i(q,K)$, $tf_i(q,K)$, ..., $none_i(q,K)$ and $st(p,K)$, $tf(p,K)$, ..., $none(p,K)$ of every K, $K = 0, 1, \ldots, M$ are given. We want to determine $st_i(p,K)$, $tf_i(p,K)$, ..., $none_i(p,K)$. $G_i(p)$ consists of $G_i(q)$, $G(p)$ and the edges (q_l, p_f) and (i,p).

Let w_1, w_2 be the weights of (q_l, p_f) and (i,p), respectively. Let c_1, c_2 be the costs of (q_l, p_f) and (i,p), respectively. $ST_i(p,K)$ is a subgraph of $G_i(p)$ which is a minimum weight Steiner tree with a cost at most K and containing all target vertices of $G_i(p)$. Thus one of the following four exhaustive cases applies to the

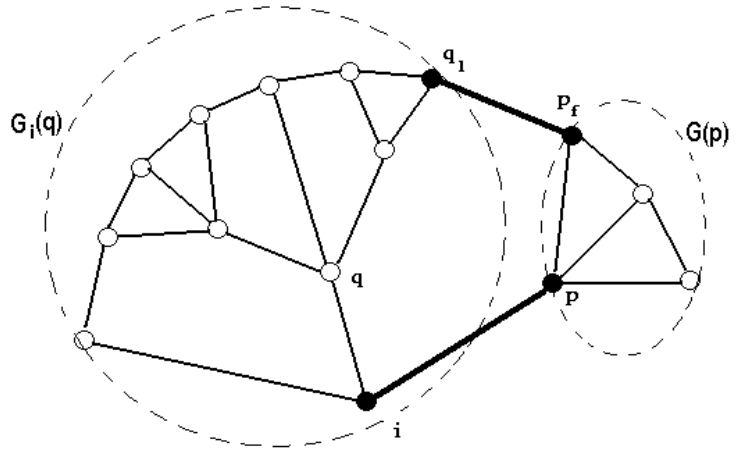

FIGURE 4. The configuration of $G_i(p)$.

subgraph $ST_i(p,K)$:

(i) $ST_i(p,K)$ contains neither (q_l,p_f) nor (i,p);
(ii) $ST_i(p,K)$ contains (q_l,p_f) but not (i,p);
(iii) $ST_i(p,K)$ contains (i,p) but not (q_l,p_f);
(iv) $ST_i(p,K)$ contains both (q_l,p_f) and (i,p).

Case (i) means that $ST_i(p,K)$ is just $ST_i(q,K_1)$ or $ST(p,K_1)$ for some K_1, $0 \le K_1 \le K$. In this case $st_i(p,K)$ is the minimum of the numbers in the following set:

$$\{st_i(q,K_1)+none(p,K_2); st(p,K_1)+none_i(q,K_2) | K_1+K_2 = K, K_1 \ge 0, K_2 \ge 0\}\cdot$$

Case (ii) means that $ST_i(p,K)$ must contain vertices q_l, p_f and edge (q_l,p_f) but not (i,p). Edge (q_l,p_f) connects two parts of $ST_i(p,K)$: the first part is a subgraph of $G_i(q)$, which contains the vertex q_l. It must be a $TL_i(q,K_1)$ for some $K_1 \ge 0$; the second part is a subgraph of $G(p)$, which contains the vertex p_f. It must be a $TF(p,K_2)$ for some $K_2 \ge 0$, where $K_1+K_2 = K-c_1 \ge 0$. Then $st_i(p,K)$ is the minimum of the numbers in the following set:

$$\{w_1+tl_i(q,K_1)+tf(p,K_2) | K_1+K_2 = K-c_1 \ge 0, K_1 \ge 0, K_2 \ge 0\}\cdot$$

Case (iii) means that $ST_i(p,K)$ must contain vertices i, p and edge (i,p) but not (q_l,p_f). Edge (i,p) connects two parts of $ST_i(p,K)$: $TC_i(q,K_1)$ and $TC(p,K_2)$, where $K_1+K_2 = K-c_2 \ge 0$, $K_1 \ge 0$, $K_2 \ge 0$. Then $st_i(p,K)$ is the minimum of the numbers in the set:

$$\{w_2+tc_i(q,K_1)+tc(p,K_2) | K_1+K_2 = K-c_2 \ge 0,\ K_1 \ge 0,\ K_2 \ge 0\}\cdot$$

Case (iv) means that $ST_i(p,K)$ must contain vertices i, p, q_l and p_f. $ST_i(p,K)$ consists of two edges, (i,p) and (q_l,p_f), and the other two subgraphs in $G_i(q)$ and $G(p)$, respectively. The subgraph in $G_i(q)$ is of the form $CL_i(q,K_1)$ or $DCL_i(q,K_1)$ for some $K_1 \geq 0$, the subgraph in $G(p)$ is of the form $DFC(p,K_2)$ or $FC(p,K_2)$ for some $K_2 \geq 0$, where $K_1+K_2 = K-c_1-c_2 \geq 0$. Since $ST_i(p,K)$ is a tree, $st_i(p,K)$ is the minimum of the numbers in the set:

$$\{w_1+w_2+\min\{cl_i(q,K_1)+dfc(p,K_2); dcl_i(q,K_1)+fc(p,K_2)\} | K_1+K_2 = K-c_1-c_2 \geq 0, K_1 \geq 0, K_2 \geq 0\}.$$

Therefore, the recurrence rule of $st_i(p,K)$ is as follows.

- For $K = 0,1,\ldots,M$, $st_i(p,K)$ is the minimum of the numbers in the following sets:
  $\{st_i(q,K_1)+none(p,K_2); st(p,K_1)+none_i(q,K_2) | K_1+K_2 = K, K_1 \geq 0, K_2 \geq 0\}$;
  $\{w_1+tl_i(q,K_1)+tf(p,K_2) | K_1+K_2 = K-c_1 \geq 0, K_1 \geq 0, K_2 \geq 0\}$;
  $\{w_2+tc_i(q,K_1)+tc(p,K_2) | K_1+K_2 = K-c_2 \geq 0, K_1 \geq 0, K_2 \geq 0\}$;
  $\{w_1+w_2+\min\{cl_i(q,K_1)+dfc(p,K_2);\ dcl_i(q,K_1)+fc(p,K_2)\} | K_1+K_2 = K-c_1-c_2 \geq 0,$
  $K_1 \geq 0, K_2 \geq 0\}$.

The recurrence rules of $tf_i(p,K), tc_i(p,K), \ldots, none_i(p,K)$ can be obtained by a similar way. We refrain from presenting these rules here in detail.

3.3. Determination of the Steiner tree

After contraction of F_i, i becomes a vertex on the cycle C_{k+1} of H_{k+1}, and $G_i(r)$ is equal to $G(i)$, where r is the last vertex in $F_i \cap C_k$ (in clockwise order). Consequently, for every K, $K = 0,1,\ldots,M$, we get $st(i,K) = st_i(r,K), tf(i,K) = tf_i(r,K), \ldots, ddd(i,K) = ddd_i(r,K), none(i,K) = none_i(r,K)$, and this process can be repeated until the graph H_0 is reduced to a graph H_t consisting of two vertices i and j joined by three parallel edges. Let $G(i)$ and $G(j)$ be subgraphs of H_0 corresponding to i and j in H_t, respectively. In the same way as in Winter [9], we can employ a contraction sequence $H_0, H_1, \ldots, H_t$, such that $G(j)$ consists of a single vertex j. Let w_1, w_2, w_3 and c_1, c_2, c_3 be weights and costs of the three edges $(i_f,j), (i,j), (i_l,j)$, respectively. W.l.o.g. assume that there at least one target vertex in $G(i)$. For every cost constraint K, $K = 0,1,\ldots,M$, let $w(T_S,K)$ be the minimum weight of a Steiner tree for the target set S in H_0 with the cost no more than K, the corresponding tree is denoted by $T_S(K)$. The configuration of $T_S(K)$ must be one of the following cases:

(i) $T_S(K)$ contains none of (i_f,j), (i,j) and (i_l,j);
(ii) $T_S(K)$ contains one of (i_f,j), (i,j) and (i_l,j);
(iii) $T_S(K)$ contains two of (i_f,j), (i,j) and (i_l,j);
(iv) $T_S(K)$ contains all of (i_f,j), (i,j) and (i_l,j).

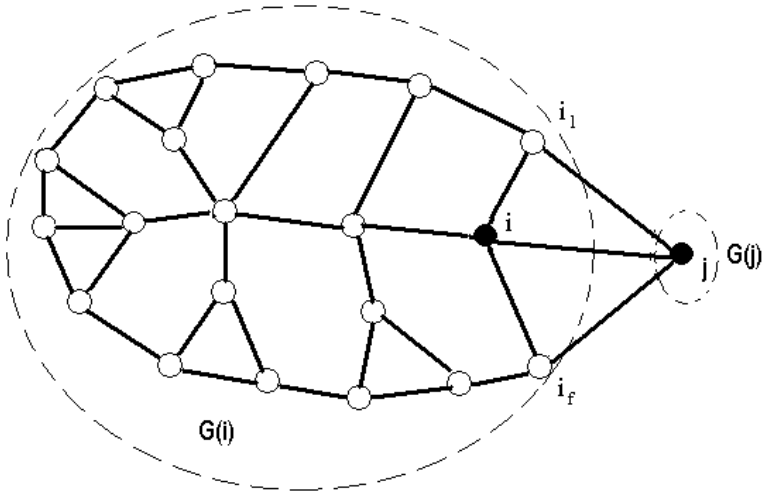

FIGURE 5. The configuration of $G(i)$ and $G(j)$.

In case (i), $T_S(K)$ is the subtree $ST(i,K)$ of $G(i)$. In addition, we should consider whether j is a target vertex. Thus $w(T_S,K)$ is $st(i,K)+none(j,0)$.

Case (ii): If $T_S(K)$ contains (i_f,j), then $T_S(K)$ consists of (i_f,j) and a subtree of $G(i)$. This subtree is a Steiner minimum tree containing i_f with a cost at most $K-w_1$. That means $w(T_S,K)$ is $w_1+tf(i,K-c_1)$. In the other two subcases, we get that $w(T_S,K)$ is $w_2+tc(i,K-c_2)$, or $w_3+tl(i,K-c_2)$.

In case (iii) a similar analysis shows that $w(T_S,K)$ is $w_1+w_2+dfc(i,K-c_1-c_2)$, or $w_1+w_3+dfl(i,K-c_1-c_3)$, or $w_2+w_3+dcl(i,K-c_2-c_3)$.

In case (iv), $T_S(K)$ contains all the three edges. Then $T_S(K)$ consists of these three edges and a subgraph of $G(i)$. This subgraph consists of three disjoint Steiner trees containing i_f, i and i_l, respectively, with the total cost at most $K-c_1-c_2-c_3$. That means $w(T_S,K)$ is $w_1+w_2+w_3+ddd(i,K-c_1-c_2-c_3)$.

Based on above discussion, we know that $w(T_S,K)$ is the minimum of the following numbers, and a weight BIG which indicates the nonexistence of a cost constrained Steiner tree.

$st(i,K)+none(j,0)$;
$w_1+tf(i,K-c_1)$, for $K\geq c_1$;
$w_2+tc(i,K-c_2)$, for $K\geq c_2$;
$w_3+tl(i,K-c_2)$, for $K\geq c_3$;
$w_1+w_2+dfc(i,K-c_1-c_2)$, for $K\geq c_1+c_2$;
$w_1+w_3+dfl(i,K-c_1-c_3)$, for $K\geq c_1+c_3$;
$w_2+w_3+dcl(i,K-c_2-c_3)$, for $K\geq c_2+c_3$;
$w_1+w_2+w_3+ddd(i,K-c_1-c_2-c_3)$, for $K\geq c_1+c_2+c_3$.

3.4. Pseudo polynomial time algorithm

We summarize the method discussed before in the following Algorithm 1.

Algorithm 1. *Pseudo polynomial time algorithm for CCSTP*

Step_1. *Let $H_0 = T_0 \cup C_0$, M be the bound for the cost. Set $k := 0$, F_i be a fan of H_k with centre i, $\{1, 2, \ldots, r\}$ be the vertices of $F_i \cap C_k$ in clockwise order.*
Define the quantities $st(p, K), \ldots, none(p, K)$ as in the initialization step (Subsection 3.2), for $p := 1, 2, \ldots, r$ and $K := 0, 1, \ldots, M$;

Step_2. *Use the recurrence rules of Subsection 3.2 to get $st_i(p, K), \ldots, none_i(p, K)$, for $p := 1, 2, \ldots, r$ and $K := 0, 1, \ldots, M$;*

Step_3. *Let $st(i, K) := st_i(r, K), \ldots, none(i, K) := none_i(r, K)$ for every $K = 0, 1, \ldots, M$ when F_i is contracted into a vertex and H_k is transformed to H_{k+1}. Set $k := k + 1$.*
if H_k consists of only two vertices with three parallel edges, goto Step_*4;*
Otherwise, take a fan F_i of H_k and the vertices $\{1, 2, \ldots, r\}$ of $F_i \cap C_k$ in clockwise order and goto Step_*2;*

Step_4. *Get the minimum weight Steiner tree for the target set S with a cost bound K, for every $K = 0, 1, \ldots, M$.*

Theorem 3.1. *For the target set S, Algorithm 1 correctly determines the minimum weight of a Steiner tree interconnecting all the vertices in S with a cost no more than a given bound M, and a weight* BIG *indicates the nonexistence of a cost constrained Steiner tree. Furthermore, the time complexity of Algorithm 1 is $O(n(M + 1)^2)$, where n is the number of vertices in the graph.*

Proof. We note that the variables are initialized correctly. In order to prove the recurrence rules, we simply need to list all relevant cases. We prove the rule for $st(i, M)$, the others can be verified in a similar way.

$G_i(1)$ consists of $G(1)$, vertex i and edge $(i, 1)$. If $i \notin S$, then $ST_i(1, K)$ contains neither i nor the edge $(i, 1)$, hence $st_i(1, K) = st(1, K)$. If $i \in S$, then i must belong to $G_i(1)$. If $G(1)$ contains no target vertex, *i.e.*, $none(1, K) = 0$, then $ST_i(1, K)$ consists of the vertex i alone. Thus $st_i(1, K) = 0$. If $G(1)$ contains some target vertices, *i.e.*, $none(1, K) = BIG$, then $ST_i(1, K)$ must contain the edge $(i, 1)$. $ST_i(1, K)$ is a union of $(i, 1)$ and $TC(1, K - c(i, 1))$, thus $st_i(1, K) = w(i, 1) + tc(1, K - c(i, 1))$. This proves the correct definition of $st_i(1, K)$.

For $p \geq 2$, $G_i(p)$ consists of $G_i(q)$, $G(p)$ and of the edges $e_1 = (q_l, p_f)$ and $e_2 = (i, p)$. We have to consider the following four cases for $ST_i(p, K)$:

(i) $ST_i(p, K)$ contains neither e_1 nor e_2;
(ii) $ST_i(p, K)$ contains e_1 but not e_2;
(iii) $ST_i(p, K)$ contains e_2 but not e_1;
(iv) $ST_i(p, K)$ contains both e_1 and e_2.

These four cases exhaust all possible configurations of $ST_i(p, K)$ and correspond to the four sets used for computing $st_i(p, K)$. This proves the correctness of the recurrence rule for $st_i(p, K)$.

For computing the minimum weight $w(T_S, K)$, we simply need to exhaust all possibilities of the three edges $(i_f, j), (i, j)$ and (i_l, j). The 8 cases just correspond to the 8 numbers used for computing $w(T_S, K)$.

Since there are $O(M+1)$ choices for each of the numbers K, K_1, K_2, the update of the quantities can be accomplished in $O((M+1)^2)$ time for each reduction. There are at most n reductions. This completes the proof of the theorem. □

We point out that the cost constrained Steiner tree can be constructed in $O(n(M+1))$ extra time by some bookkeeping operations during the reduction. If we fix M at 0 and assume the cost of all edges in the graph are zero, the above algorithm finds a minimum weight Steiner tree in $O(n)$ time.

4. The polynomial time approximation scheme

We use scaling and rounding techniques [1,2,4,8] to turn the pseudo polynomial time algorithm for $CCSTP$ into a fully polynomial time approximation scheme for $WCSTP$ in Halin graphs.

Let $c(S, W)$ be the minimum cost of a Steiner tree spanning the targets in S with a weight of no more than W. Given a positive real number C and a constant $\epsilon > 0$, we can decide in polynomial time whether $c(S, W) > C$ or $c(S, W) < (1+\epsilon)C$. This technique is described by the following Algorithm 2.

Algorithm 2. $TEST(C, \epsilon)$

Step_1. *Set* $\theta := \frac{n-1}{C \cdot \epsilon}$*;*
Let $c_\theta(e) := \lfloor c(e) \cdot \theta \rfloor$ *for every* $e \in E$*;*
Set $M := C \cdot \theta$*;*

Step_2. *Apply Algorithm 1 using the scaled edge cost* c_θ *instead of the original edge cost* c*;*
if *the weight of the cost constrained Steiner tree is no more than* W then
output **YES**
else
output **NO**
endif

Theorem 4.1. *For the given target set* S*, the weight constraint* W*, the positive real numbers* C *and* ϵ*, if* $TEST(C, \epsilon) =$ **NO***, then* $c(S, W) > C$*, if* $TEST(C, \epsilon) =$ **YES***, then* $c(S, W) < (1+\epsilon) \cdot C$*. In addition, the worst case time complexity of* $TEST(C, \epsilon)$ *is* $O\left(\frac{n^3}{\epsilon^2}\right)$.

The proof of the above theorem is standard and almost the same as in [1], so we omit it here. Now we use L and U to denote a lower and upper bound on $c(S, W)$, respectively. Our FPTAS starts with efficiently computable values of L and U and uses bisection to drive the ratio $\frac{U}{L}$ to below 4.

The initial values of L and U can be computed as follows. Let $c_1 < c_2 < \ldots < c_k$ be the different edge cost values. For a Halin graph with n vertices we have clearly $k \leq 2n-2$. Denote by G_j the graph obtained by changing the weight of the edges

in G as following: if the cost of edge e is more than c_j, then let the weight of this edge become *BIG*. We can compute a minimum weight Steiner tree in G_j in $O(n)$ time for every $j = 1, 2, \ldots, k$. Let J be the smallest index j such that the weight of the minimum weight Steiner tree in G_j is no more than W. From the definition of G_j, no edge with weight *BIG* can be included in the minimum weight Steiner tree of G_j, unless the weight of the tree is more than *BIG*. Therefore $c(S, W)$, the cost of an optimal solution to the weight constrained Steiner tree must be in the interval $[c_J, c_J \cdot (n-1)]$. We may use c_J and $c_J \cdot (n-1)$ as the initial values for L and U, respectively. The index J can be found in $O(n \log n)$ time by bisection. Then we can use scaling and rounding to compute a $(1+\epsilon) - approximation$ to the weight constrained Steiner tree. Algorithm 3 summarizes the FPTAS for a weight constrained minimum cost Steiner tree problem in Halin graphs.

Algorithm 3. *FPTAS for finding a weight constrained minimum cost Steiner tree in a Halin graph*

Step_1. *Set L and U to their initial values such that $U \leq L \cdot (n-1)$;*
Step_2. if $U \leq 4L$ then
goto Step_*3;*
else
let $C := \sqrt{\frac{U \cdot L}{2}}$;
if $TEST(C, 1) =$**NO**, *set* $L = C$;
if $TEST(C, 1) =$**YES**, *set* $U = 2C$;
goto Step_*2;*
endif
Step_3. *Set* $\theta := \frac{n-1}{L \cdot \epsilon}$;
$M := U \cdot \theta$;
Let $c_\theta(e) := \lfloor c(e) \cdot \theta \rfloor$ *for every* $e \in E$;
Apply Algorithm 1 using the scaled edge cost c_θ instead of the original edge cost c, find the smallest integer $C \leq M$ such that there is a weight constrained Steiner tree whose scaled cost is C and output the corresponding Steiner tree.

Theorem 4.2. *If there is a weight constrained Steiner tree for target set S, then Algorithm 3 finds a $(1+\epsilon)$–approximation T of the weight constrained Steiner tree. If there is no weight constrained Steiner tree of set S, we can find this out during the computation of the initial values of U and L: one can not find the smallest index J. Furthermore, the worst case time complexity of Algorithm 3 is $O(n^3 \cdot (\log \log n + \frac{1}{\epsilon^2}))$.*

Proof. The correctness of the algorithm is due to Theorems 3.1 and 4.1.
Now let us proof the time complexity. As discussed before Theorem 4.2, Step_1 requires $O(n \log n)$ time. Let $L^{[0]}$ and $U^{[0]}$ be the initial lower bound and upper bound for $c(S, W)$, and $L^{[k]}$ and $U^{[k]}$ be the lower and upper bound for $c(S, W)$ after k iterations of Step_2 for $k \geq 1$.

According to the rules of Step_2,

$$U^{[k]} = 2 \cdot \sqrt{\frac{U^{[k-1]} \cdot L^{[k-1]}}{2}},\ L^{[k]} = L^{[k-1]},$$

or

$$L^{[k]} = \sqrt{\frac{U^{[k-1]} \cdot L^{[k-1]}}{2}},\ U^{[k]} = U^{[k-1]}.$$

Therefore

$$\frac{U^{[k]}}{L^{[k]}} = \sqrt{\frac{2 \cdot U^{[k-1]}}{L^{[k-1]}}}.$$

Then

$$\log \frac{U^{[k]}}{L^{[k]}} = \frac{1}{2} \log \frac{2 \cdot U^{[k-1]}}{L^{[k-1]}} = \left(\sum_{i=1}^{i=k} \frac{1}{2^i} \right) \log 2 + \frac{1}{2^k} \log \frac{U^{[0]}}{L^{[0]}} \leq 1 + \frac{1}{2^k} \log(n-1).$$

As a result, we would have $\frac{U^{[k]}}{L^{[k]}} \leq 4$ after k iterations where k is no more than $\log \log n$. Every iteration of Step_2 requires $O(n^3)$ time according Theorem 4.1. Then the total time of Step_2 is $O(n^3 \log \log n)$. In Step_3, we apply Algorithm 1 to compute all quantities for $K = 0, 1, \ldots, M$, in $O(\frac{n^3}{\epsilon^2})$ time. Then we find the smallest integer C in $O(\log \frac{n}{\epsilon})$ extra time by bisection. Therefore the total time complexity of Algorithm 3 is $O\left(n^3 \cdot \left(\log \log n + \frac{1}{\epsilon^2}\right)\right)$. □

5. Conclusions

In this paper we have studied the weight constrained minimum cost Steiner tree problem on a class of Halin graphs which generalize in a nontrivial way tree and ring networks. Although the traditional Steiner tree problem is polynomial time solvable on Halin graphs, it is shown that the weight constrained minimum cost Steiner tree problem on Halin graphs is NP-hard. We presented a fully polynomial time approximation scheme for this problem, which has many applications to communication networks. Our result is also of interest from the network design point of view, as it can be used to obtain suboptimal solutions to the weight constrained minimum cost Steiner tree problem on a general graph by appropriately removing some edges such that the graph becomes a Halin graph.

References

[1] G. Chen and G. Xue, A PTAS for weight constrained Steiner trees in series-parallel graphs. Springer-verlag, *Lecture Notes in Comput. Sci.* **2108** (2001) 519-528.

[2] G. Chen and G. Xue, K-pair delay constrained minimum cost routing in undirected networks. *Proc. of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms* (2001) 230-231.

[3] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of P-Completeness*. San Francisco, W.H. Freeman and Company (1979).

[4] R. Hassin, Approximation schemes for the restricted shortest path problem. *Math. Oper. Res.* **17** (1992) 36-42.
[5] F.K. Hwang and D.S. Richards, Steiner tree problems. *Networks* **22** (1992) 55-89.
[6] F.K. Hwang, D.S. Richards and P. Winter, The Steiner tree problem. *Ann. Discrete Math.* **53** (1992) 68-71.
[7] T. Jiang and L. Wang, Computing shortest networks under a fixed topology, in *Advances in Steiner Trees*, edited by D.-Z. Du, J.M. Smith and J. H. Rubinstein. Kluwer Academic Publishers (2000) 39-62.
[8] D.H. Lorenz and D. Raz, A simple efficient approximation scheme for the restricted shortest path problem. *Oper. Res. Lett.* **28** (2001) 213-219.
[9] P. Winter, Steiner problem in Halin networks. *Discrete Appl. Math.* **17** (1987) 281-294.
[10] P. Winter, Steiner problem in networks – a survey. *Networks* **17** (1987) 129-167.