Available online at www.sciencedirect.com
ScienceDirect
Information Sciences 177 (2007) 2418–2435
INFORMATION SCIENCES
AN INTERNATIONAL JOURNAL
www.elsevier.com/locate/ins

A factoring approach for the Steiner tree problem in undirected networks

Chu-Fu Wang [a,*], Rong-Hong Jan [b]

[a] Department of Computer Science, National Pingtung University of Education, No. 4-18, Ming Shen Road, Pingtung 90044, Taiwan, ROC
[b] Department of Computer Science, National Chiao Tung University, 1001 Ta Hsueh Road, Hsinchu 30050, Taiwan, ROC

Received 22 January 2007; accepted 25 January 2007

Abstract

In communication networks, many applications, such as video on demand and video conferencing, must establish a communications tree that spans a subset K in a vertex set. The source node can then send identical data to all nodes in set K along this tree. This kind of communication is known as multicast communication. A network optimization problem, called the Steiner tree problem (STP), is presented to find a least cost multicasting tree. In this paper, an $O(|E|)$ algorithm is presented to find a minimum Steiner tree for series–parallel graphs where $|E|$ is the number of edges. Based on this algorithm, we proposed an $O(2^{2c} \cdot |E|)$ algorithm to solve the Steiner tree problem for general graphs where c is the number of applied factoring procedures. The c value is strongly related to the topology of a given graph. This is quite different from other algorithms with exponential time complexities in $|K|$.

Keywords: Steiner tree problem; Multicast routing; Network optimization; Two-terminal series–parallel graph

1. Introduction

The demand for multimedia transmission in computer networks is rapidly increasing, and these multimedia transmissions are time sensitive. To maintain Quality of Service (QoS), networks must allocate enough bandwidth for transmitting multimedia data. Two types of multimedia transmissions are usually used. They are point-to-point transmission and point-to-multipoint transmission. The point-to-multipoint transmission, known as multicast, occurs when the multimedia data is delivered to a subset of nodes in the network. Notable examples of multicast applications include video conferencing, distributed games, distributed simulations, and file replication on mirrored sites. In a multicast communication, the source sends identical data to all destinations. Since the data can be duplicated at switching nodes, it is not necessary for the source node to send separate copies to all destinations. Thus a good multicasting path can help reduce the number of redundant streams flowing on the network.

* Corresponding author. Fax: +886 8 7215034.
E-mail address: cfwang@mail.npue.edu.tw (C.-F. Wang).

0020-0255/$ - see front matter
doi:10.1016/j.ins.2007.01.014

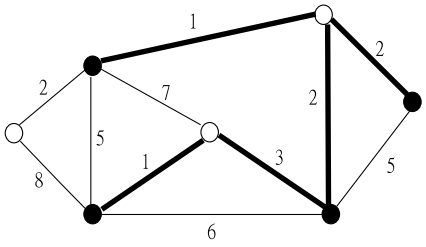

Fig. 1. A minimum Steiner tree in a graph.

A communication network can be modeled as a weighted undirected graph $G = (V, E, C)$, where V is the set of vertices that represents communication nodes, E is the set of edges that represents the communication links and a cost function $C : E \rightarrow \Re^+$ maps each edge to a positive real number which represents the communication cost. Let $w(H)$ denote the weight of a graph H which is equal to $\sum_{e_i \in E(H)} c(e_i)$ where $c(e_i)$ is the edge cost of e_i. For a given vertex set $K \subseteq V$, the STP is to find a tree T, such that $K \subseteq V(T)$ and $w(T)$ are minimized.

Throughout this paper, we refer $T_K^*(H)$ represents the *minimum Steiner tree* for graph H with respect to the *Steiner vertex set* K. A vertex v in H is also called a non-Steiner vertex if $v \in K$; otherwise, vertex v is called a Steiner vertex. In Fig. 1, the black nodes indicate the non-Steiner vertices. The heavy line segments indicate the minimum Steiner tree for this graph.

The STP is known to be *NP*-complete [10]. Several solution approaches, such as, dynamic programming [6,17,20], branch-and-bound [9,27,33], enumeration approaches [2,12], linear relaxations [18,32], and Lagrangean relaxations [4,7], have been proposed for finding the exact solution for STP. Among these works, Hakimi [12] has proposed a spanning tree enumeration algorithm that runs in $O(k^2 2^{n-k} + n^3)$ time, where $n = |V|$ and $k = |K|$. Levin [17] has proposed a dynamic programming approach algorithm that runs in $O(3^k n + 2^k n^2 + k^2 n)$ time. Beasley [4] has presented a Lagrangean relaxation algorithm. In addition, there are many centralized heuristic algorithms [1,8,13,15,19,22–26,28,31,34,35] and distributed heuristic algorithms [3,11,16] presented for solving STP. For detailed descriptions and more lectures on both exact and heuristic methods, one can refer to two survey papers, Hwang and Richards [14] and Winter [30].

Note that the computation time for Hakimi's (Levin's) algorithm increases exponentially with the number of set $V - K$ (K). That is, Levin's algorithm is favored when the number of non-Steiner vertices is small. Conversely, when the number of non-Steiner vertices is large, Hakimi's algorithm is favored. However, the performance of these algorithms are all limited by size of K or $V - K$. This paper proposes a factoring approach for solving undirected STP. The complexity of this algorithm increases exponentially not in the number of k or $n - k$ but in the number of factoring procedures applied (c). The c value strongly relates to the graph topology. This is quite different from other algorithms.

This paper is organized as follows. In Section 2, a polynomial time algorithm for finding the minimum Steiner tree on Series–Parallel (SP) graph is described. In Section 3, we extend the solution method in Section 2 to general graphs. Concluding remarks are given in Section 4.

2. Solution method for series–parallel graphs

Although the STP is known to be *NP*-complete, there are many special classes of graphs in which STP are solvable in polynomial time. Prodon et al. [21], Wald and Colbourn [29] presented algorithms that solve STP for SP graphs in $O(|E|)$, where E is the set of edges. An SP graph (with two end vertices), also known as Two-Terminal Series–Parallel graph (TTSP graph), has a well-defined structure. It can be defined recursively as follows.

1. A single edge is a TTSP graph.
2. If H_1 and H_2 are TTSP graphs, so is the graph obtained by collapsing one end vertex of H_1 with one end vertex of H_2 (series composition. The resulting graph is denoted by $H_1 * H_2$).
3. If H_1 and H_2 are TTSP graphs, so is the graph obtained by collapsing two end vertices of H_1 and two end vertices of H_2, respectively (parallel composition. The resulting graph is denoted by $H_1 \vee H_2$).

Following is a description of the algorithm proposed by Wald and Colbourn [29] which summarize solving STP on TTSP graphs in detail. The solution strategy here is to reverse the recursive composition; that is, starting from each edge repeatedly applying a series or parallel composition until the whole TTSP graph is completely constructed. During each composition, we maintain five types of subgraphs, which summarizes useful Steiner tree information for each TTSP subgraph.

We use $T_K(H)$ to denote a subtree T of TTSP graph H where vertex set $K \subseteq V(T)$. In particular, we use $T_{K+u}(H)$ ($T_{K-v}(H)$) to denote a subtree T of H where $K \subseteq V(T)$ and $u \in V(T)$ ($v \notin V(T)$). Note that, $T^*_K(H)$ is the minimum Steiner tree for H; i.e., $w(T^*_K(H)) = \min_{T_K(H)\subseteq H} w(T_K(H))$. With respect to SP graph H which has two end vertices u, v, and a Steiner vertex set K, we define five subgraphs $T^*_{K+u+v}(H)$, $T^*_{K+u-v}(H)$, $T^*_{K+v-u}(H)$, $T^*_{K-u-v}(H)$, and $ST^*_{K+u+v}(H)$ for H as follows:

- Tree $T^*_{K+u+v}(H)$ is a subtree of H such that
$$w(T^*_{K+u+v}(H)) = \min_{T_{K+u+v}(H)\subseteq H} w(T_{K+u+v}(H)) \tag{1}$$
- Tree $T^*_{K+u-v}(H)$ is a subtree of H such that
$$w(T^*_{K+u-v}(H)) = \min_{T_{K+u-v}(H)\subseteq H} w(T_{K+u-v}(H)) \tag{2}$$
- Tree $T^*_{K+v-u}(H)$ is a subtree of H such that
$$w(T^*_{K+v-u}(H)) = \min_{T_{K+v-u}(H)\subseteq H} w(T_{K+v-u}(H)) \tag{3}$$
- Tree $T^*_{K-u-v}(H)$ is a subtree of H such that
$$w(T^*_{K-u-v}(H)) = \min_{T_{K-u-v}(H)\subseteq H} w(T_{K-u-v}(H)) \tag{4}$$
- Subgraph $ST^*_{K+u+v}(H)$ is a union of trees $T_{U+u-v}(H)$ and $T_{(K\setminus U)+v-u}(H)$, where $U \subseteq K$, such that
$$w(ST^*_{K+u+v}(H)) = \min_{U\subseteq K} w(T_{U+u-v}(H) \cup T_{(K\setminus U)+v-u}(H)) \tag{5}$$

  We refer to these five subgraphs $T^*_{K+u+v}(H)$, $T^*_{K+u-v}(H)$, $T^*_{K+v-u}(H)$, $T^*_{K-u-v}(H)$, and $ST^*_{K+u+v}(H)$ as the elementary Steiner subgraphs of a TTSP graph H. Fig. 2 gives an example of a TTSP graph and its corresponding elementary Steiner subgraphs.

Let H_1 with end points (u_1, v_1) and H_2 with end points (u_2, v_2) be two TTSP subgraphs, and K_1 and K_2 be the Steiner vertex sets for H_1 and H_2, respectively. Assume that $(T^*_{K_1+u_1+v_1}(H_1), T^*_{K_1+u_1-v_1}(H_1), T^*_{K_1+v_1-u_1}(H_1), T^*_{K_1-u_1-v_1}(H_1), ST^*_{K_1+u_1+v_1}(H_1))$ and $(T^*_{K_2+u_2+v_2}(H_2), T^*_{K_2+u_2-v_2}(H_2), T^*_{K_2+v_2-u_2}(H_2), T^*_{K_2-u_2-v_2}(H_2), ST^*_{K_2+u_2+v_2}(H_2))$ denote the elementary Steiner subgraphs for H_1 and H_2, respectively. Thus, the elementary Steiner subgraphs for $H_1 * H_2$ and $H_1 \vee H_2$ can be numerically computed and summarized as follows.

(1) The elementary Steiner subgraphs for $H_1 \vee H_2$ (Parallel composition).

Note that the end point u (v) of graph $H_1 \vee H_2$ is collapsed by nodes u_1 and u_2 (v_1 and v_2). We will consider the computation for $T^*_{K+u+v}(H_1 \vee H_2)$ first. The definition of $T^*_{K+u+v}(H_1 \vee H_2)$ is equal to the minimum subtree that contains both end vertices u, v and set $K = K_1 \cup K_2$. Obviously, there are only six classes from the combinations of $T_{K_1+u_1+v_1}(H_1)$, $T_{K_1+u_1-v_1}(H_1)$, $T_{K_1+v_1-u_1}(H_1)$, $ST_{K_1+u_1+v_1}(H_1)$, $T_{K_2+u_2+v_2}(H_2)$, $T_{K_2+u_2-v_2}(H_2)$, $T_{K_2+v_2-u_2}(H_2)$, and $ST_{K_2+u_2+v_2}(H_2)$ for tree $T_{K+u+v}(H_1 \vee H_2)$, which are shown in Fig. 3.

Note that $T_{K_1+u_1+v_1}(H_1) \cup T_{K_2+u_2+v_2}(H_2)$ is not a tree. This is because $T_{K_1+u_1+v_1}(H_1)$ contains a path from u to v in subgraph H_1, and $T_{K_2+u_2+v_2}(H_2)$ also contains a path from v to u in subgraph H_2. Thus, these two paths form a cycle in $T_{K_1+u_1+v_1}(H_1) \cup T_{K_2+u_2+v_2}(H_2)$.

For any graph H with Steiner vertex set K, using Eq. (5), we have

$$w(ST^*_{K+u+v}(H)) = \min_{U\subseteq K} w(T_{U+u-v}(H) \cup T_{(K\setminus U)+v-u}(H))$$

If $U = K$, then

$$w(ST^*_{K+u+v}(H)) \leqslant w(T^*_{K+u-v}(H) \cup T^*_{\emptyset+v-u}(H)) = w(T^*_{K+u-v}(H)) \tag{6}$$

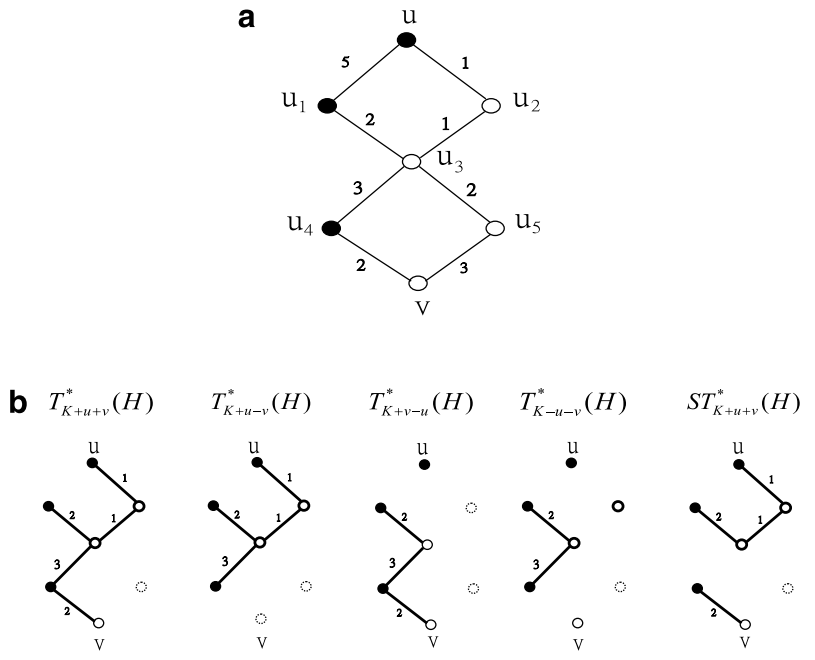

Fig. 2. (a) A TTSP graph H, (b) the elementary Steiner subgraphs for H.

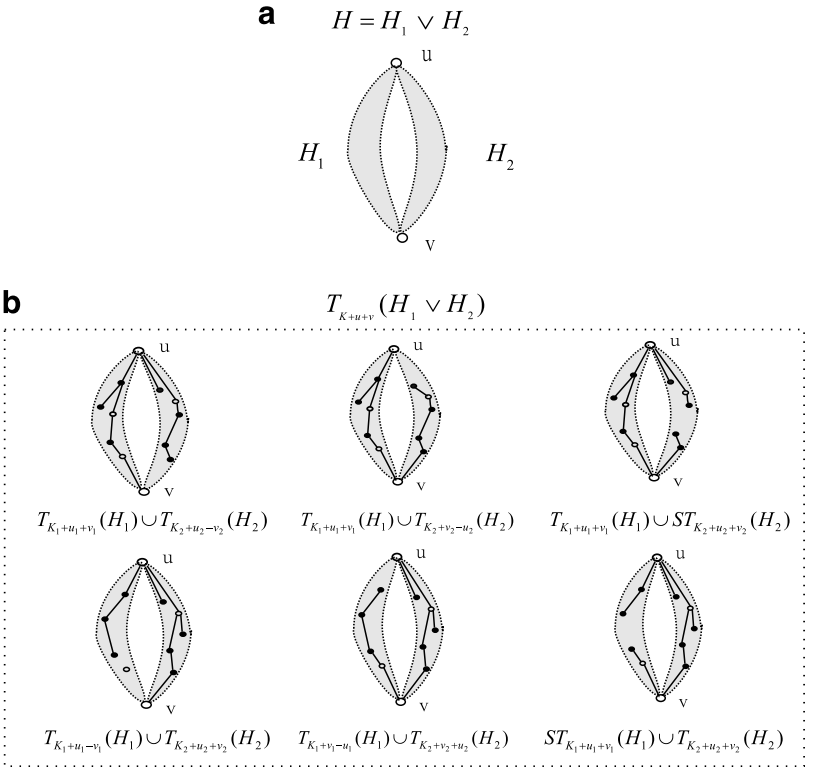

Fig. 3. Possible combinations for $T_{K+u+v}(H_1 \vee H_2)$.

Similarly, if $U = \emptyset$, we have

$$w(ST^*_{K+u+v}(H)) \leqslant w(T^*_{K+v-u}(H)) \tag{7}$$

It is clear that H_1 and H_2 are edge disjoint. From Inequation (6), we have

$$\begin{aligned} w(T^*_{K_1+u_1+v_1}(H_1) \cup ST^*_{K_2+u_2+v_2}(H_2)) &= w(T^*_{K_1+u_1+v_1}(H_1)) + w(ST^*_{K_2+u_2+v_2}(H_2)) \\ &\leqslant w(T^*_{K_1+u_1+v_1}(H_1)) + w(T^*_{K_2+u_2-v_2}(H_2)) \\ &= w(T^*_{K_1+u_1+v_1}(H_1) \cup T^*_{K_2+u_2-v_2}(H_2)) \end{aligned}$$

That is,

$$w(T^*_{K_1+u_1+v_1}(H_1) \cup ST^*_{K_2+u_2+v_2}(H_2)) \leqslant w(T^*_{K_1+u_1+v_1}(H_1) \cup T^*_{K_2+u_2-v_2}(H_2)) \tag{8}$$

Similarly, from (7) we have

$$w(T^*_{K_1+u_1+v_1}(H_1) \cup ST^*_{K_2+u_2+v_2}(H_2)) \leqslant w(T^*_{K_1+u_1+v_1}(H_1) \cup T^*_{K_2+v_2-u_2}(H_2)) \tag{9}$$

From (8) and (9), we conclude that if the tree $T^*_{K+u+v}(H_1 \vee H_2)$ occurs in classes of combinations $T_{K_1+u_1+v_1}(H_1)$ and $T_{K_2+u_2-v_2}(H_2)$, $T_{K_2+v_2-u_2}(H_2)$, $ST_{K_2+u_2+v_2}(H_2)$, then tree $T^*_{K+u+v}(H_1 \vee H_2)$ is equal to the combination of $T^*_{K_1+u_1+v_1}(H_1)$ and $ST^*_{K_2+u_2+v_2}(H_2)$. (i.e., $T^*_{K+u+v}(H_1 \vee H_2) = T^*_{K_1+u_1+v_1}(H_1) \cup ST^*_{K_2+u_2+v_2}(H_2)$). Similarly, we have the following inequalities:

$$\begin{aligned} &w(T^*_{K_2+u_2+v_2}(H_2) \cup ST^*_{K_1+u_1+v_1}(H_1)) \leqslant w(T^*_{K_2+u_2+v_2}(H_2) \cup T^*_{K_1+u_1-v_1}(H_1)) \quad \text{and} \\ &w(T^*_{K_2+u_2+v_2}(H_2) \cup ST^*_{K_1+u_1+v_1}(H_1)) \leqslant w(T^*_{K_2+u_2+v_2}(H_2) \cup T^*_{K_1+v_1-u_1}(H_1)) \end{aligned}$$

Conversely, if the tree $T^*_{K+u+v}(H_1 \vee H_2)$ occurs in classes of combinations $T_{K_2+u_2+v_2}(H_2)$ and $T_{K_1+u_1-v_1}(H_1)$, $T_{K_1+v_1-u_1}(H_1)$, $ST_{K_1+u_1+v_1}(H_1)$, then $T^*_{K+u+v}(H_1 \vee H_2) = ST^*_{K_1+u_1+v_1}(H_1) \cup T^*_{K_2+u_2+v_2}(H_2)$. Therefore, $T^*_{K+u+v}(H_1 \vee H_2)$ is the tree such that

$$w(T^*_{K+u+v}(H_1 \vee H_2)) = \min\{w(T^*_{K+u+v}(H_1) \cup ST^*_{K_2+u_2+v_2}(H_2)), w(ST^*_{K_1+u_1+v_1}(H_1) \cup T^*_{K+u+v}(H_2))\} \tag{10}$$

Next, consider $T^*_{K+u-v}(H_1 \vee H_2)$. Note that the subgraph $H_1 \vee H_2$ is composed of two edge-disjoint subgraphs H_1 and H_2. Thus,

$$\begin{aligned} w(T^*_{K+u-v}(H_1 \vee H_2)) &= \min\{w(T_{K_1+u_1-v_1}(H_1) \cup T_{K_2+u_2-v_2}(H_2))\} \\ &= \min\{w(T_{K_1+u_1-v_1}(H_1)) + w(T_{K_2+u_2-v_2}(H_2))\} \\ &= \min\{w(T_{K_1+u_1-v_1}(H_1))\} + \min\{w(T_{K_2+u_2-v_2}(H_2))\} \\ &= w(T^*_{K_1+u_1-v_1}(H_1)) + w(T^*_{K_2+u_2-v_2}(H_2)) \end{aligned}$$

Therefore,

$$T^*_{K+u-v}(H_1 \vee H_2) = T^*_{K_1+u_1-v_1}(H_1) \cup T^*_{K_2+u_2-v_2}(H_2) \tag{11}$$

Similarly,

$$T^*_{K+v-u}(H_1 \vee H_2) = T^*_{K_1+v_1-u_1}(H_1) \cup T^*_{K_2+v_2-u_2}(H_2) \tag{12}$$

$$ST^*_{K+u+v}(H_1 \vee H_2) = ST^*_{K_1+u_1+v_1}(H_1) \cup ST^*_{K_2+u_2+v_2}(H_2) \tag{13}$$

Now, consider the computation for $T^*_{K-u-v}(H_1 \vee H_2)$. If $K_1 \neq \emptyset$ and $K_2 \neq \emptyset$ then $T^*_{K-u-v}(H_1 \vee H_2) = \emptyset$. This is because a tree contains a vertex $a \in K_1$ and a vertex $b \in K_2$ should at least contain a vertex u or v. If $K_1 = \emptyset$ and $K_2 \neq \emptyset$ then $T^*_{K-u-v}(H_1 \vee H_2) = T^*_{K_2-u_2-v_2}(H_2)$. Similarly, if $K_2 = \emptyset$ and $K_1 \neq \emptyset$ then $T^*_{K-u-v}(H_1 \vee H_2) = T^*_{K_1-u_1-v_1}(H_1)$. Thus we have

$$T^*_{K-u-v}(H_1 \vee H_2) = \begin{cases} T^*_{K_2-u_2-v_2}(H_2) & \text{if } K_1 = \emptyset \text{ and } K_2 \neq \emptyset \\ T^*_{K_1-u_1-v_1}(H_1) & \text{if } K_1 \neq \emptyset \text{ and } K_2 = \emptyset \\ \emptyset, & \text{otherwise} \end{cases} \tag{14}$$

Therefore, we have the following lemma.

Lemma 1. *Given two TTSP graphs H_1 and H_2, the elementary Steiner subgraphs with respect to $H_1 \vee H_2$ can be correctly obtained using Eqs. (10)–(14).*

(2) The elementary Steiner subgraphs for $H_1 * H_2$ (Series composition).

Assume that the Steiner vertex set K of $H_1 * H_2$ is equal to $K_1 \cup K_2$, nodes u_1, v_2 are two end points of $H_1 * H_2$, and node s is collapsed by end points v_1 and u_2. It is clear that

$$T^*_{K+u_1+v_2}(H_1 * H_2) = T^*_{K_1+u_1+v_1}(H_1) \cup T^*_{K_2+u_2+v_2}(H_2) \tag{15}$$

Consider the computation for $T^*_{K+u_1-v_2}(H_1 * H_2)$. If $K_2 \neq \emptyset$ (see Fig. 4a), the tree $T^*_{K+u_1-v_2}(H_1 * H_2)$ is equal to $T^*_{K_1+u_1+v_1}(H_1) \cup T^*_{K_2+u_2-v_2}(H_2)$. For the other case $K_2 = \emptyset$, it is clear that $T^*_{K+u_1-v_2}(H_1 * H_2) = T^*_{K_1+u_1-v_1}(H_1)$. That is

$$T^*_{K+u_1-v_2}(H_1 * H_2) = \begin{cases} T^*_{K_1+u_1+v_1}(H_1) \cup T^*_{K_2+u_2-v_2}(H_2), & K_2 \neq \emptyset \\ T^*_{K_1+u_1-v_1}(H_1), & \text{otherwise} \end{cases} \tag{16}$$

Similarly

$$T^*_{K+v_2-u_1}(H_1 * H_2) = \begin{cases} T^*_{K_1+v_1-u_1}(H_1) \cup T^*_{K_2+u_2+v_2}(H_2), & K_1 \neq \emptyset \\ T^*_{K_2+v_2-u_2}(H_2), & \text{otherwise} \end{cases} \tag{17}$$

The computation for $T^*_{K-u_1-v_2}(H_1 * H_2)$ can be divided into three cases, i.e., case 1: $K_1 \neq \emptyset$ and $K_2 \neq \emptyset$; case 2: $K_1 \neq \emptyset$ and $K_2 = \emptyset$; case 3: $K_1 = \emptyset$ and $K_2 \neq \emptyset$. These three cases are shown in Fig. 4b. Applying the same arguments for Eq. (16), we have

$$T^*_{K-u_1-v_2}(H_1 * H_2) = \begin{cases} T^*_{K_1-u_1-v_1}(H_1) & \text{if } K_2 = \phi \\ T^*_{K_2-u_2-v_2}(H_2) & \text{if } K_1 = \phi \\ T^*_{K_1+v_1-u_1}(H_1) \cup T^*_{K_2+u_2-v_2}(H_2), & \text{otherwise} \end{cases} \tag{18}$$

Finally, we will consider $ST^*_{K+u_1+v_2}(H_1 * H_2)$. Let $T_{U_1^*+u_1-v_1}(H_1)$ and $T_{(K_1\setminus U_1^*)+v_1-u_1}(H_1)$ ($T_{U_2^*+u_2-v_2}(H_2)$ and $T_{(K_2\setminus U_2^*)+v_2-u_2}(H_2)$) denote two subtrees composed of $ST^*_{K_1+u_1+v_1}(H_1)$ ($ST^*_{K_2+u_2+v_2}(H_2)$). Using Eq. (5):

$$w(ST^*_{K+u+v}(H_1 * H_2)) = \min_{U \subseteq K} w(T_{U+u-v}(H_1 * H_2) \cup T_{(K\setminus U)+v-u}(H_1 * H_2))$$

Note that there are three cases of $U \subseteq K$, i.e., $U = U_1^*$, $U = K_1 \cup U_2^*$ and $U = K_1$ (see Fig. 5).

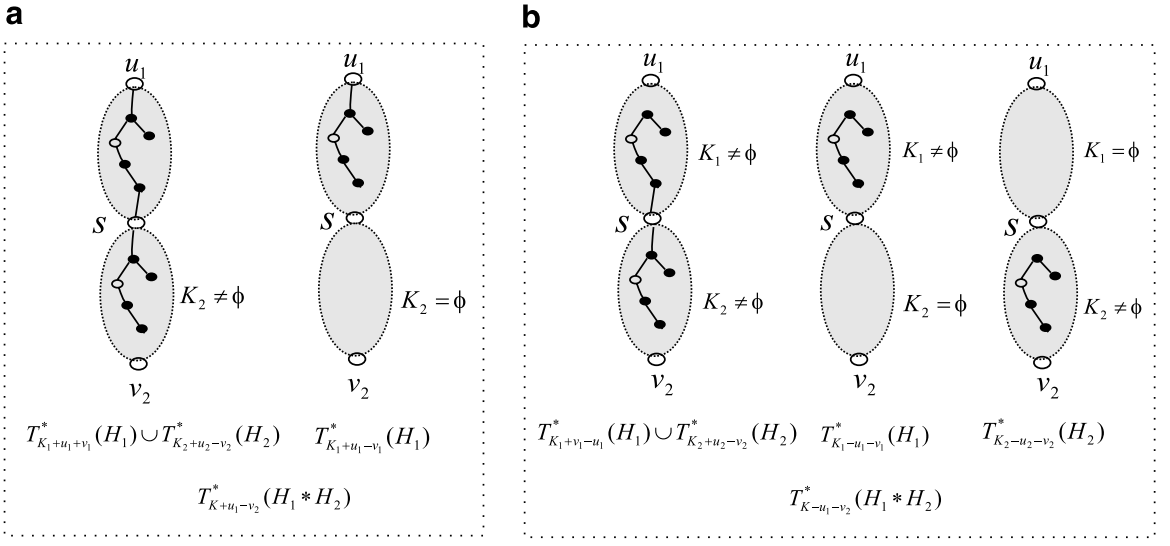

Fig. 4. Possible cases for $T^*_{K+u_1-v_2}(H_1 * H_2)$ and $T^*_{K-u_1-v_2}(H_1 * H_2)$.

Therefore, the tree $ST^*_{K+u_1+v_2}(H_1 * H_2)$ is the tree such that

$$w(ST^*_{K+u_1+v_2}(H_1 * H_2)) = \min\{w(ST^*_{K_1+u_1+v_1}(H_1) \cup T^*_{K_2+u_2+v_2}(H_2)), w(T^*_{K_1+u_1+v_1}(H_1) \cup ST^*_{K_2+u_2+v_2}(H_2)), w(T^*_{K_1+u_1-v_1}(H_1) \cup T^*_{K_2+v_2-u_2}(H_2))\} \quad (19)$$

Therefore we have the following lemma:

Lemma 2. *Given two TTSP graphs H_1 and H_2, the elementary Steiner subgraphs with respect to $H_1 * H_2$ can be correctly obtained using Eqs. (15)–(19).*

The complete algorithm for solving STP on TTSP graphs is shown in Fig. 6. Fig. 7 shows the solution steps for a TTSP graph H with Steiner vertex set $K = \{u, u_1, u_4\}$ (see Fig. 7a). Initially, each edge (u, v) is associated with elementary Steiner subgraphs $(\{(u, v)\}, \{u\}, \{v\}, \emptyset, \{u, v\})$. We can apply series composition to edges (u, u_1) and (u_1, u_3) and find the elementary Steiner subgraphs for $H_1 = (u, u_1) * (u_1, u_3)$ by Eqs. (15)–(19) (see Fig. 7b). Similarly, the elementary Steiner subgraphs for H_2,H_3, and H_4 can also be obtained.

Next, we apply parallel composition to TTSP subgraphs H_1 and H_2 (H_3 and H_4) and obtain elementary Steiner subgraphs for $H_5 = H_1 \vee H_2$ ($H_6 = H_3 \vee H_4$) (see Fig. 7c). Apply a series composition for H_5 and

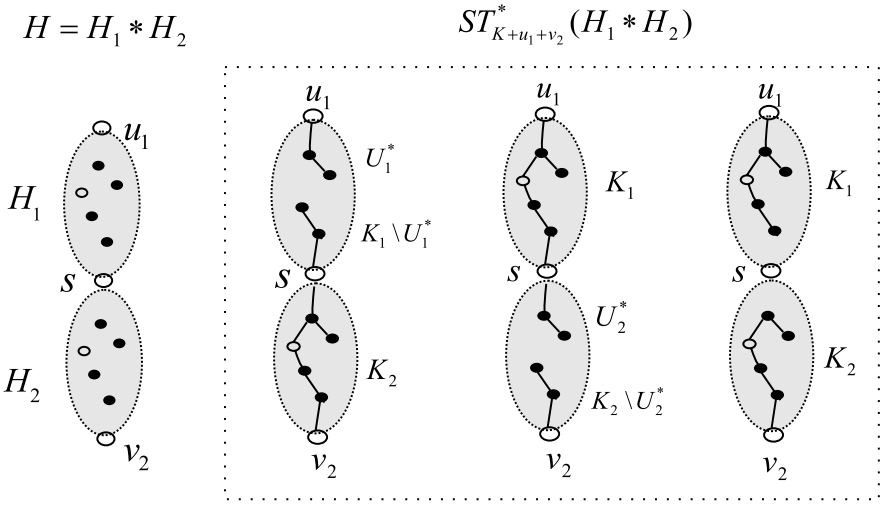

Fig. 5. The possible cases for $ST^*_{K+u_1+v_2}(H_1 * H_2)$.

Algorithm 1;
input: a weighted TTSP graph H and a Steiner vertex set K;
output: a minimum cost tree which spans every vertex in K.
begin
Step 1. Associate each edge (u, v) with elementary Steiner subgraphs $(\{(u, v)\}, \{u\}, \{v\}, \emptyset, \{u, v\})$;
Step 2. Apply any possible series or parallel composition until no composition can be applied. Let the resulting elementary Steiner subgraphs be $(T^*_{K+u+v}(H), T^*_{K+u-v}(H), T^*_{K+v-u}(H), T^*_{K-u-v}(H), ST^*_{K+u+v}(H))$;
Step 3. Output the tree T^* such that:

$$w(T^*) = \begin{cases} w(T^*_{K+u+v}(H)) & if\ u, v \in K \\ min\{w(T^*_{K+u-v}(H)), w(T^*_{K+u+v}(H))\} & if\ u \in K\ and\ v \notin K \\ min\{w(T^*_{K+v-u}(H)), w(T^*_{K+u+v}(H))\} & if\ v \in K\ and\ u \notin K \\ min\{w(T^*_{K-u-v}(H)), w(T^*_{K+u-v}(H)), & \\ \quad w(T^*_{K+v-u}(H)), w(T^*_{K+u+v}(H))\} & if\ u, v \notin K\ and\ T^*_{K-u-v}(H) \neq \emptyset \\ min\{w(T^*_{K+u-v}(H)), w(T^*_{K+v-u}(H)), & \\ \quad w(T^*_{K+u+v}(H))\} & if\ u, v \notin K\ and\ T^*_{K-u-v}(H) = \emptyset \end{cases}$$

end

Fig. 6. Algorithm for finding minimum Steiner tree in a TTSP graph.

H_6 and obtain the elementary Steiner subgraphs $(T^*_{K+u+v}(H), T^*_{K+u-v}(H), T^*_{K+v-u}(H), T^*_{K-u-v}(H), ST^*_{K+u+v}(H))$ for graph $H = H_5 * H_6$ (see Fig. 7d). Since $u \in K$ and $v \notin K$, then $w(T^*_K(H)) = \min\{w(T^*_{K+u-v}(H)), w(T^*_{K+u+v}(H))\} = \min\{7, 9\} = 7$, the minimum Steiner tree is $T^*_K(H) = T^*_{K+u-v}(H)$.

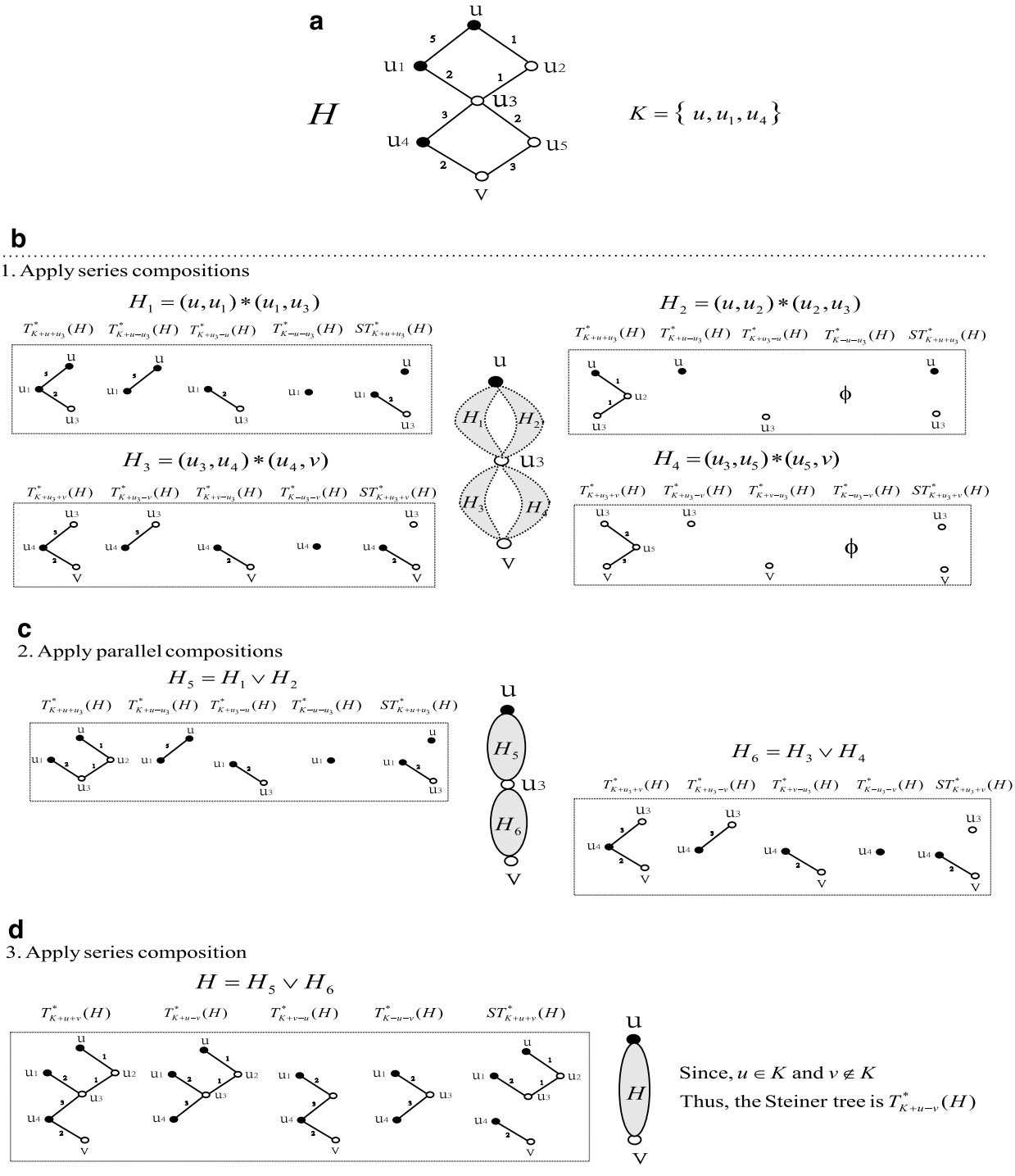

Fig. 7. A numerical example for solving STP in TTSP graph H.

Let us consider the complexity of this algorithm. Note that the computation of either series or parallel composition takes only constant time. Moreover, the number of series or parallel composition processes applied to a TTSP graph is equal to the number of edges $|E|$. Thus, we have the following theorem:

Theorem 3. *Algorithm* 1 *takes* $O(|E|)$ *to determine the minimum Steiner tree for weighted TTSP graph* $H = (V, E, C)$.

3. Solution method for general graphs

Given a weighted undirected graph $G = (V, E, C)$ and a Steiner vertex set K, if G is a TTSP graph, then algorithm 1 can solve STP in G to optimal using $O(|E|)$ time. However, algorithm 1 will fail if the input graph is not a TTSP graph. In this section, we will extend the solution method for algorithm 1 to solve STP in general graph cases. The proposed solution method consists of two major operations called the *TTSP subgraphs reduction* and the *factoring reduction*. For the TTSP subgraphs reduction, a recognizing procedure is first applied to identify every TTSP subgraph in G. Algorithm 1 can then compute the five elementary Steiner subgraphs with respect to each TTSP subgraph. Each TTSP subgraph is replaced using a single edge and associating this edge with the respective elementary Steiner subgraphs. We call the above operations the TTSP subgraph reduction (a detailed description will be given in Section 3.1) and the resulting graph is denoted by $[G]$. For the factoring reduction, an edge in graph $[G]$ is chosen arbitrarily for application to the reduction operation, which can decompose the STP into several subproblems. Note that the graph size for each subproblem is reduced compared to the original graph size of $[G]$. Moreover, the optimal solution for the original problem can easily be computed using the solutions from those subproblems (a detailed description will be shown in Section 3.2).

The complete solution method for solving STP in a general graph is illustrated by constructing a solution problem tree and summarized as follows. In the first step, we apply the TTSP subgraph reduction to the original problem and regard the resulting reduced problem as the root node of the solution problem tree. An edge is arbitrarily chosen in the resulting graph for applying the factoring reduction and then several subproblems will be obtained. Meanwhile, the TTSP subgraph reduction is applied to each subproblem. Let each of the resulting reduced subproblems be the child-node of the root in the solution problem tree. By continuously applying the factoring reduction and the TTSP subgraph reduction to the problem of each leaf node, the solution problem tree can be expanded. Note that if the resulting graph of a subproblem is a TTSP it can then be solved using algorithm 1 and no further expansion of this branch is required. The complete solution problem tree can then be constructed. In the solution problem tree, since the optimal solution with respect to each of the non-leaf node's problem can be determined by the solutions of its child-nodes, in the second step, we start the optimal solution computation from the leaf-nodes of the solution problem tree and then go upward until the root node is reached. The Steiner tree of the original graph will then be obtained. A detailed description of this proposed method will be discussed later in Section 3.2.

Now we define some notations and operations for graphs that will help to illustrate the proposed method. Given a weighted graph $G = (V, E, C)$ and a vertex u in G, let $G - u$ be a subgraph of G obtained by deleting u and all edges that incident to u from G. Let H be a TTSP subgraph with end vertices u and v in G. $G - H$ is a subgraph of G in which the vertex set of $G - H$ is $V(G) - V(H) \cup \{u, v\}$ and the edge set of $G - H$ is $E(G) - E(H)$. In graph G, *contracting* the TTSP subgraph H of G involves finding $G - H$ and then merging the end vertices u and v into a supervertex. Furthermore, if parallel edges occur in the resulting graph, only the edge with the least cost is retained. The graph obtained from G by contracting H is denoted by $G \cdot H$. Fig. 8 shows an example for TTSP subgraph deletion and contraction. Fig. 8a shows a graph G and a TTSP subgraph H with end vertices u and v, which is enclosed by a dashed line. The subgraph $G - H$ is shown in Fig. 8b. The subgraph $G \cdot H$ can be obtained by merging u and v of $G - H$ into a supervertex u'. The resulting graph is shown in Fig. 8c.

The following subsections will detail the proposed method for solving the Steiner tree problem in general graphs. In Section 3.1, an $O(|E|)$ procedure to identify all TTSP subgraphs in a graph is first given. Based on this procedure, the TTSP subgraphs reduction is discussed. The factoring reduction and the complete algorithm to solve STP for general graphs will be discussed in Section 3.2.

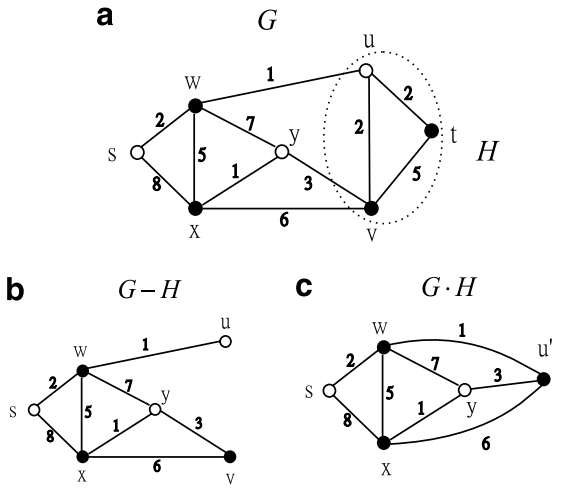

Fig. 8. An example for TTSP subgraph deletion and contraction.

3.1. The TTSP subgraphs reduction

A more stricter and more formal definition of TTSP subgraphs, called the *maximal strict TTSP subgraphs*, is given as follows:

Definition 1. A *strict TTSP subgraph* H_{uv} of a graph G is a TTSP with end vertices u and v, such that there does not exist any edge e_{xy} in G, where $x \in V(G) - V(H_{uv})$ and $y \in V(H_{uv}) - \{u, v\}$.

Definition 2. A *strict TTSP subgraph* H_{uv} of a graph G is called *maximal* if H_{uv} is not a proper subgraph of any other strict TTSP subgraph of G.

Consider a graph G as shown in Fig. 9a. The maximal strict TTSP subgraphs H_{ad} and H_{cg} are shown by heavy line and dash line in Fig. 9b, respectively. In the following, a recognized procedure called MaxStri-TTSP-Recog is proposed to identify every maximal strict TTSP subgraph in a given graph. We will illustrate this procedure step by step by taking Fig. 9 as an example. Note that any TTSP subgraph can be recognized and constructed alternatively by series composition and parallel composition. For example, as shown in Fig. 9b, the TTSP subgraph H_{cg} can be constructed using the following steps. At first, the series composition of two edges e_{ce} and e_{ef} in G can be easily recognized since the degree of the incidence node e is 2. We then update the graph G by removing node e from G and then add a new edge e_{cf} into G; that is $G = G - e + e_{cf}$. Since, e_{cf} is already an edge in G before updating, the resulting graph will therefore contain two parallel edges, which reveals a parallel composition case. Now we retain only one edge in the resulting graph and

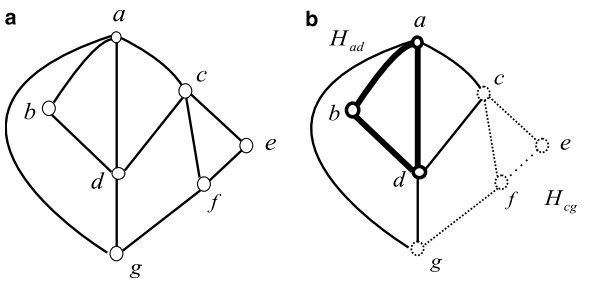

Fig. 9. An example of maximal strict TTSP subgraphs.

consequently another series composition case occurrs due to node degree of f becoming 2. Continue the same approach until no series or parallel composition case is found. A maximal strict TTSP subgraph H_{cg} will then be obtained.

In the MaxStri-TTSP-Recog procedure, let node set S be the collection of nodes whose degree are 2, P a multi-set of edges that contains every parallel edge in graph G, and T a subgraph set that contains the TTSP subgraphs found so far. Initially, we compute the degree of each node in G. In the case of a node with degree 2 we put it into set S. Fig. 10a shows the results after the initial step is applied, where $S = \{b, e\}$, $P = \emptyset$, and $T = \emptyset$. A node in S is arbitrarily chosen, say node b. Since the degree of node b is 2, a series composition case in G is found. We let the newly found TTSP subgraph be $H_{ad} = (\{a, b, d\}, \{e_{ab}, e_{bd}\})$ and then put it into set T. The graph G is update to $G = G - b + e_{ad}$. Note that the resulting graph contains two parallel edges. We use notation e^2_{ad} to denote these two parallel edges and then put e^2_{ad} into set P. Similarly, the same operations are made on node $e \in S$. The results are shown in Fig. 10b. Since set $P = \{e^2_{ad}, e^2_{cf}\}$, two parallel composition cases were found. For the parallel edges e^2_{cf} in P, we only retain one edge in G and update the node degrees of c and f by decreasing 1, respectively. After the updating process, since the node degree of f becomes 2, another series composition case is found and we add node f into S for later manipulation. The respective TTSP subgraph $H_{cf} \in T$ is updated by adding edge e_{cf} into it. Similarly, the same operations are made on parallel edges e^2_{ad} in P. The results are shown in Fig. 11a. The same steps are continued until both set S and P are empty, and every maximal strict TTSP subgraph in G is recognized (see Fig. 11b). Notice that any edge in the resulting graph G may represent a single edge or a TTSP subgraph in the original graph. The complete description for MaxStri-TTSP-recog is shown in Fig. 12. Obviously, the computation time for procedure MaxStri-TTSP-recog is $O(|E|)$.

Let $H^1_{u_1v_1}, H^2_{u_2v_2}, \ldots, H^m_{u_mv_m}$ be all the maximal strict TTSP subgraphs of G that were recognized by MaxStri-TTSP-recog (see Fig. 13a). If we apply algorithm 1 to each TTSP subgraph $H^i_{u_iv_i}$ $(1 \leqslant i \leqslant m)$, then five elemen-

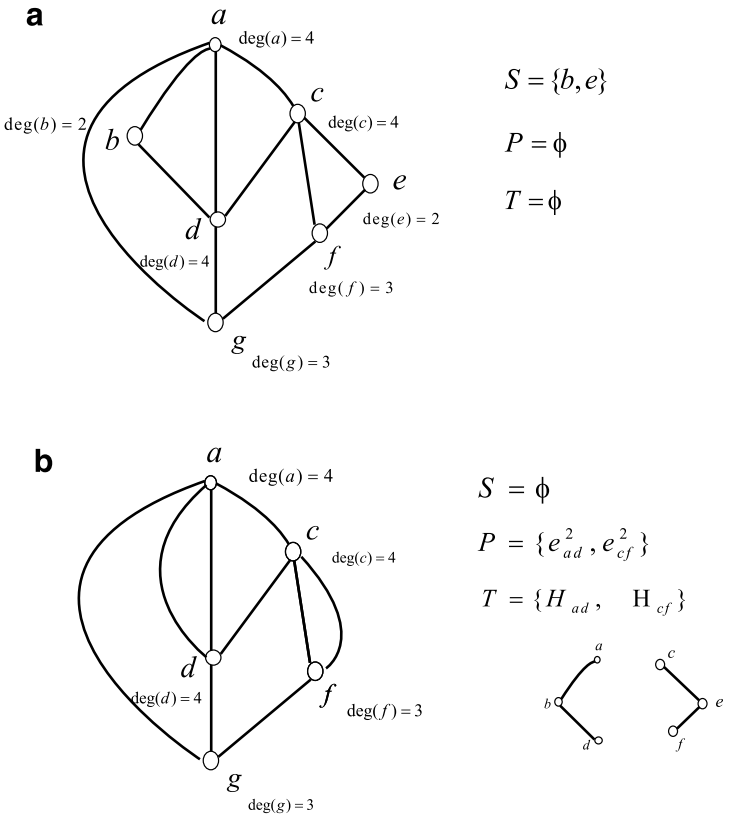

Fig. 10. An example for applying procedure MaxStri-TTSP-recog.

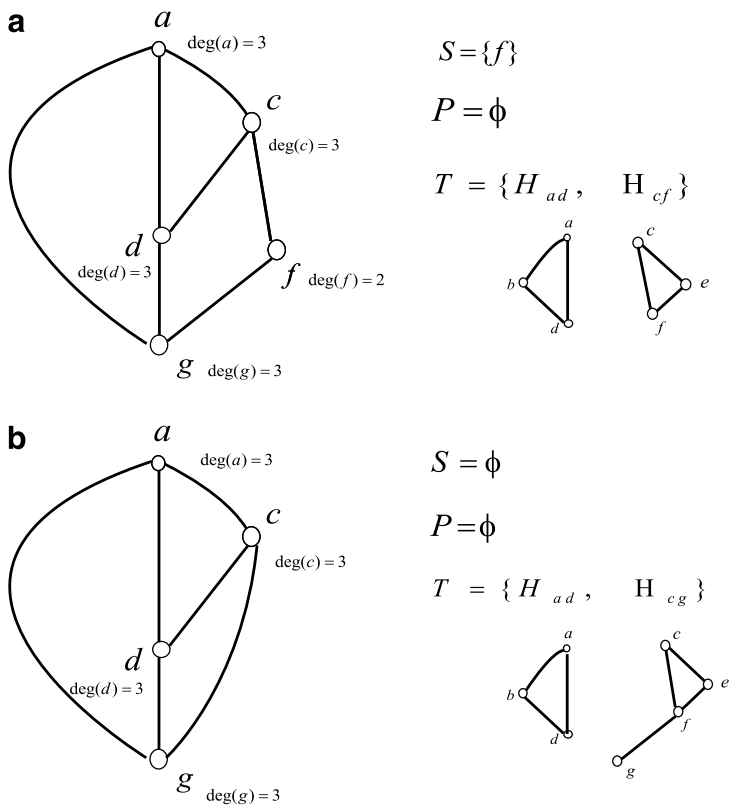

Fig. 11. An example for applying procedure MaxStri-TTSP-recog (cont.).

tary Steiner subgraphs $(T^*_{K'+u_i+v_i}(H^i_{u_iv_i}), T^*_{K'+u_i-v_i}(H^i_{u_iv_i}), T^*_{K'+v_i-u_i}(H^i_{u_iv_i}), T^*_{K'-u_i-v_i}(H^i_{u_iv_i}), ST^*_{K'+u_i+v_i}(H^i_{u_iv_i}))$ with respect to $H^i_{u_iv_i}$ are obtained, where $K' \subset K$ is the collection of Steiner vertices in $H^i_{u_iv_i}$. In the next subsection we will show that the intersection of the Steiner tree and the TTSP subgraph $H^i_{u_iv_i}$ is equal to one of the following elementary Steiner subgraphs $T^*_{K'+u_i+v_i}(H^i_{u_iv_i})$, $T^*_{K'+u_i-v_i}(H^i_{u_iv_i})$, $T^*_{K'+v_i-u_i}(H^i_{u_iv_i})$, $T^*_{K'-u_i-v_i}(H^i_{u_iv_i})$, or $ST^*_{K'+u_i+v_i}(H^i_{u_iv_i})$. Based on this fact, for each TTSP subgraph $H^i_{u_iv_i}$, we only need to retain its respective elementary Steiner subgraphs for later Steiner tree computation. Thus, we replace $H^i_{u_iv_i}$ $(1 \leqslant i \leqslant m)$ in G with only a single edge $e_{u_iv_i}$ and associate the respective elementary Steiner subgraphs with this edge. A problem that is reduced and equivalent to the original STP will then be obtained. We call the above operation the TTSP subgraphs reduction.

3.2. The factoring reduction

Let H be a TTSP subgraph with end vertices u and v in G, subset $K' \subseteq K$ the collection of Steiner vertices in H. Let $(T^*_{K'+u+v}(H), T^*_{K'+u-v}(H), T^*_{K'+v-u}(H), T^*_{K'-u-v}(H), ST^*_{K'+u+v}(H))$ be the elementary Steiner subgraphs for H. Without loss of generality, we assume $K' \neq \emptyset$. We call a subtree found by $H \cap T^*_K(G)$ an induced subtree of $T^*_K(G)$ and denote it as $\langle H \rangle$. Note that every path from a vertex in H to a vertex in $G - H$ must pass u or v. Because $K' \neq \emptyset$, tree $T^*_K(G)$ at least contained an end vertex u or v. On the condition of the connectivity between u and v for $T^*_K(G)$, we have

Case 1: $T^*_K(G)$ does not contain a path from u to v. Since $T^*_K(G)$ is a tree and at least has an end vertex u or v, case 1 can be divided into two subcases: one is $u \in T^*_K(G)$, $v \notin T^*_K(G)$ and the other is $u \notin T^*_K(G)$, $v \in T^*_K(G)$. In subcase $u \in T^*_K(G)$, $v \notin T^*_K(G)$, it is clear that $\langle H \rangle = T^*_K(G) \cap H = T^*_{K'+u-v}(H)$ since $T^*_K(G)$ is connected (see Fig. 14b). Similarly, in subcase $u \notin T^*_K(G)$, $v \in T^*_K(G)$, we have $\langle H \rangle = T^*_K(G) \cap H = T^*_{K'+v-u}(H)$ (see Fig. 14c).

Procedure MaxStri-TTSP-Recog;
input: an undirected graph $G = (V, E)$;
output: all maximal strict TTSP subgraphs in G.
begin
{ **Initial step** }
compute $deg(v), \forall v \in V$;
let $S = \{v | deg(v) = 2, \forall v \in V\}$, $P = \emptyset$, and $T = \emptyset$;

{ **Iteration step** }
repeat

{ **The series composition case** }
while $(S \neq \emptyset)$ do
arbitrarily choose a node $v \in S$ and let e_{uv}, e_{vw} be the edges that incident on v;
let H_{uv} and H_{vw} be the correspondent TTSP subgraphs of edge e_{uv} and e_{vw} in T, respectively; Note that, if $(H_{uv} \notin T)$ then $H_{uv} \leftarrow \{e_{uv}\}$ and if $(H_{vw} \notin T)$ then $H_{vw} \leftarrow \{e_{vw}\}$;
if $(e_{uw} \in E$ and $e_{uw} \notin P)$ then $P \leftarrow P \cup \{e_{uw}, e_{uw}\}$;
else if $(e_{uw} \in E$ and $e_{uw} \in P)$ then $P \leftarrow P \cup \{e_{uw}\}$;
$G \leftarrow G - v + e_{uw}$;
$T \leftarrow T - \{H_{uv}, H_{vw}\} \cup \{H_{uv} * H_{vw}\}$;
end while

{ **The parallel composition case** }
while $(P \neq \emptyset)$ do
arbitrarily choose two parallel edges e_{uv} and e'_{uv} in P;
let H_{uv} and H'_{uv} be the correspondent TTSP subgraphs of edge e_{uv} and e'_{uv} in T, respectively; Note that, if $(H_{uv} \notin T)$ then $H_{uv} \leftarrow \{e_{uv}\}$ and if $(H'_{uv} \notin T)$ then $H'_{uv} \leftarrow \{e'_{uv}\}$;
$G \leftarrow G - e'_{uv}$;
$deg(u) \leftarrow deg(u) - 1$;
$deg(v) \leftarrow deg(v) - 1$;
if $(deg(u) == 2)$ then $S \leftarrow S \cup \{u\}$;
if $(deg(v) == 2)$ then $S \leftarrow S \cup \{v\}$;
if the number of parallel edges e_{uv} in P is equal to 2 then
$P \leftarrow P - \{e_{uv}, e'_{uv}\}$
else
$P \leftarrow P - \{e'_{uv}\}$
$T \leftarrow T - \{H_{uv}, H'_{uv}\} \cup \{H_{uv} \vee H'_{uv}\}$;
end while
until $(S = \emptyset$ and $P = \emptyset)$
end

Fig. 12. Procedure MaxStri-TTSP-Recog.

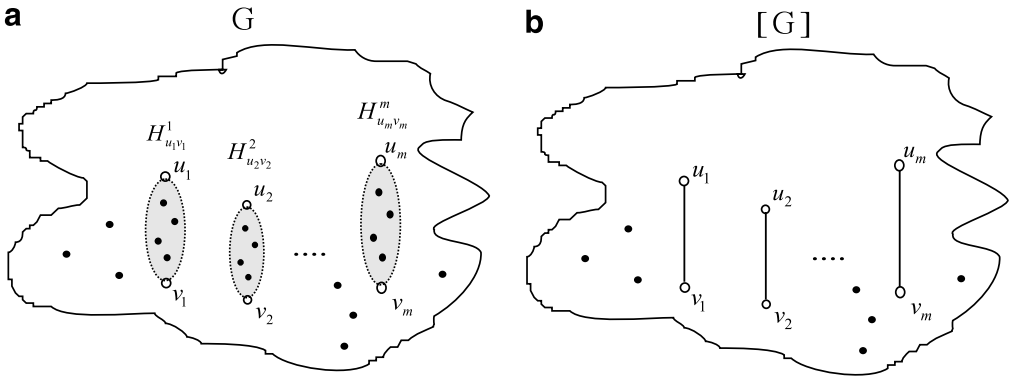

Fig. 13. The TTSP subgraphs reduction.

Case 2: $T^*_K(G)$ contains a path P_{uv} from u to v. We also divide case 2 into two subcases: one is $P_{uv} \not\subseteq H$, and the other is $P_{uv} \subseteq H$. In subcase $P_{uv} \not\subseteq H$, it is clear that $\langle H \rangle = ST^*_{K'+u+v}(H)$ (see Fig. 14d). In the other subcase $P_{uv} \subseteq H$, we have $\langle H \rangle = T^*_{K'+u+v}(H)$ (see Fig. 14e).

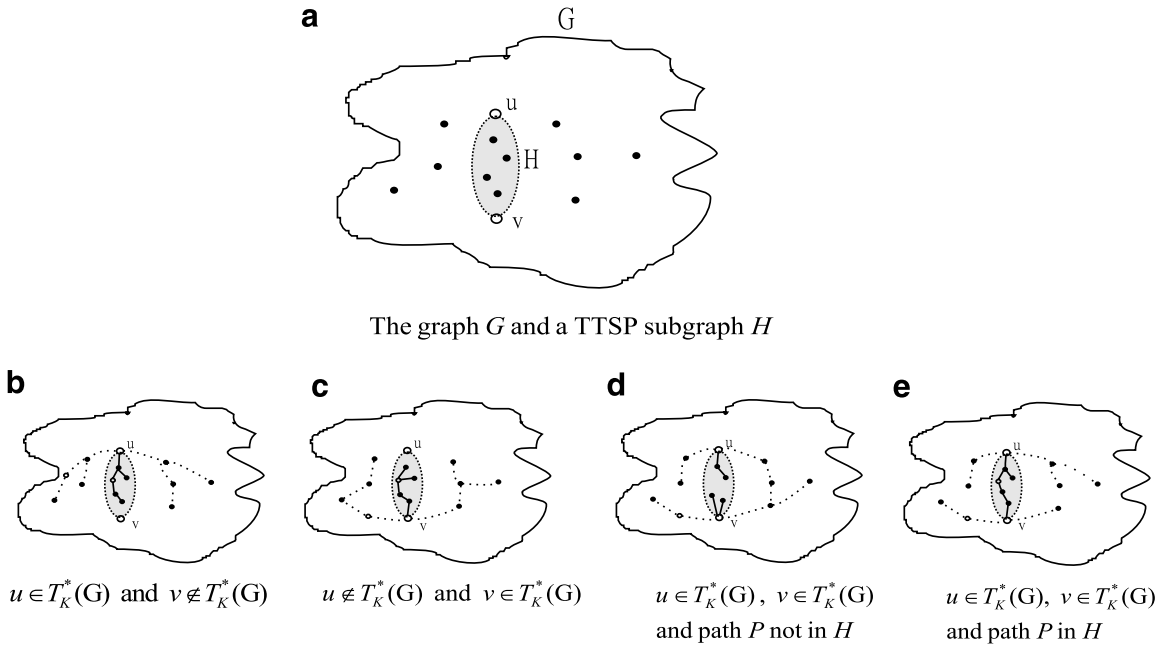

Fig. 14. Possible cases for $T_K^*(G) \cap H$.

Therefore, we claim that

$$\langle H \rangle = \begin{cases} T^*_{K'+u-v}(H) & \text{if } u \in T_K^*(G),\ v \notin T_K^*(G) \\ T^*_{K'+v-u}(H) & \text{if } u \notin T_K^*(G),\ v \in T_K^*(G) \\ ST^*_{K'+u+v}(H) & \text{if } u, v \in T_K^*(G),\ \text{path}P_{uv} \not\subseteq H \\ T^*_{K'+u+v}(H) & \text{if } u, v \in T_K^*(G),\ \text{path}P_{uv} \subseteq H \end{cases} \tag{20}$$

This result implies that we only need to maintain the elementary Steiner subgraphs for each TTSP subgraph H of G, and based on subgraph H, the STP for G can be composed into four subproblems (four subcases). Consider a graph G as shown in Fig. 15. The Steiner vertex set $K = \{r, s, x, y\}$ is indicated by black points. To find a Steiner tree $T_K^*(G)$ for G, we apply MaxStri-TTSP-Recog to identify every maximal strict TTSP subgraph of G. Assume that subgraph H is a maximal strict TTSP subgraph of G. Thus, the elementary Steiner subgraphs $(T^*_{K'+u+v}(H), T^*_{K'+u-v}(H), T^*_{K'+v-u}(H), T^*_{K'-u-v}(H), ST^*_{K'+u+v}(H))$ for H can be obtained using algorithm 1 where $K' = \{x, y\}$ (see Fig. 15).

Using Eq. (20), we decompose the STP for G into four subproblems. Fig. 16 shows the tree that is generated using our solution method. Node 0 of the tree has four child nodes that represent four cases (subproblems). Node 1 of the tree shown in Fig. 16 represents case 1. In case 1, $\langle H \rangle = T^*_{K'+u-v}(H)$. Since $T^*_{K'+u-v}(H)$ is a rooted tree with root vertex u, and does not contain vertex v, we shrink tree $T^*_{K'+u-v}(H)$ into vertex u and then consider STP for graph $G_1 = G - H - v$ with Steiner vertex set $K_1 = K - K' + \{u\}$. Similarly, we shrink tree $T^*_{K'+v-u}(H)$ into vertex v and consider STP for graph $G_2 = G - H - u$ with Steiner vertex set $K_2 = K - K' + \{v\}$ (see Node 2 in Fig. 16). Since tree $ST^*_{K'+u+v}(H)$ contains two subtrees $T_{U^*+u-v}(H)$ and $T_{(K'-U^*)+v-u}(H)$, we shrink $T_{U^*+u-v}(H)$ and $T_{(K'-U^*)+v-u}(H)$ into vertices u and v, respectively, and then consider STP for graph $G_3 = G - H$ with Steiner vertex set $K_3 = K - K' + \{u, v\}$ (see Node 3 in Fig. 16).

Node 4 of the tree shown in Fig. 16 represents case 4. Note that tree $T^*_{K'+u+v}(H)$ contains vertices u and v. We apply the contracting operation on H and obtain the graph $G \cdot H$. After contraction, we consider STP on resulting graph $G \cdot H$ with Steiner vertex set $K_4 = K - K' + \{u'\}$ where vertex u' is the supervertex that is merged by u and v. In this way, we can solve this problem recursively. The optimal solution $T_K^*(G)$ is the tree such that

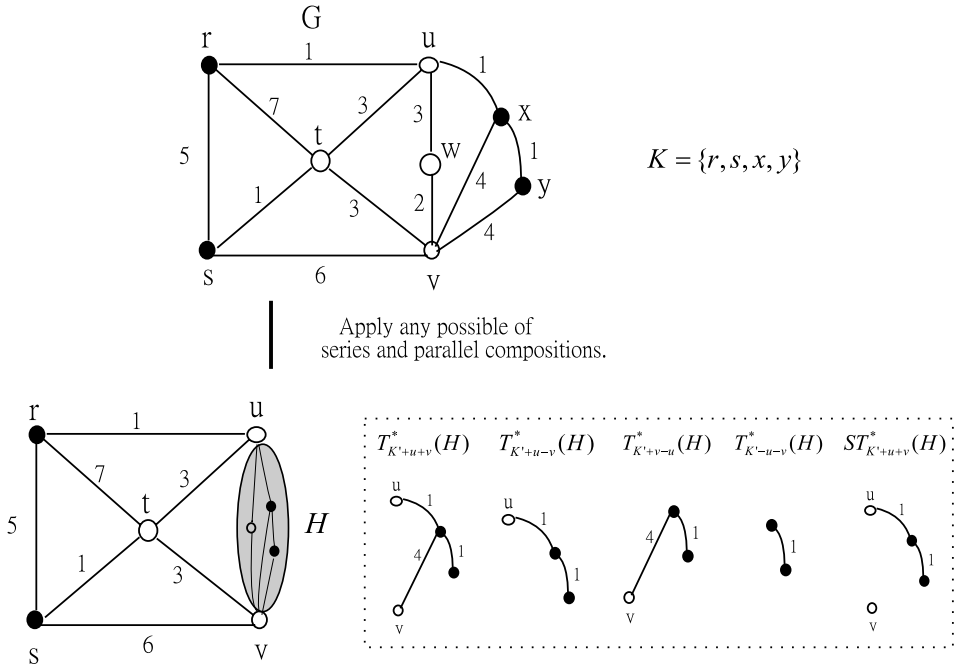

Fig. 15. A numerical example for finding minimum Steiner tree in graph G.

$$w(T^*_K(G)) = \min\{w(T^*_{K'+u-v}(H)) + w(T^*_{K_1}(G-H-v)), w(T^*_{K'+v-u}(H)) + w(T^*_{K_2}(G-H-u)), w(ST^*_{K'+u+v}(H)) + w(T^*_{K_3}(G-H)), w(T^*_{K'+u+v}(H)) + w(T^*_{K_4}(G\cdot H))\} \tag{21}$$

We call Eq. (21) the *factoring theorem*. In Fig. 16, since graphs G_1, G_2 and G_3 are TTSP graphs, we can solve these subproblems directly and obtain the optimal solutions T_1, T_2 and T_3, respectively. Because graph G_4 is not an SP graph, we choose a maximal strict TTSP subgraph from G_4, say edge (r,s) for applying the factoring theorem. Node 5 represents subproblem 1 for G_4. Note that if vertex $s \in K$ then this case ($r \in K_5$ and $s \notin K_5$) violates the constraint of the STP (i.e., $s \in T^*_K(G)$). Thus this node is bounded. Similarly, node 6 is also bounded. We solve STPs for TTSP graphs G_7 and G_8 and obtain the minimum Steiner trees T_7 and T_8. Since $w(T^*_{K_4}(G_4)) = \min\{w(T_7), w(T_8)\} = \min\{11, 12\} = 11$. Thus, we have the optimal solution T_7 for STP in graph G_4 with Steiner vertex set $K_4 = \{r, s, u'\}$. Now, we claim that the optimal solution $T^*_K(G) = T_1$ since $w(T^*_K(G)) = \min\{w(T_1), w(T_2), w(T_3), w(T_7)\} = \min\{7, 14, 10, 11\} = 7$. The complete algorithm for solving STP for general graphs is shown in Fig. 17.

Let us consider the complexity of this algorithm. Note that if G is reduced into a TTSP graph using the factoring theorem, then it takes only $O(|E|)$ to solve it. Assume in the worst case, graph G applies the factoring theorem c times before graph G is decomposed into TTSP graphs. The complexity for algorithm 2 is then $O(2^{2c} \cdot |E|)$. Using the above arguments through this section, we have the following theorem:

Theorem 4. *Algorithm 2 takes* $O(2^{2c} \cdot |E|)$ *to determine the minimum Steiner tree of a weighted undirected graph G.*

The c value in Theorem 4 is strongly related to the topology of a given graph G. For one extreme case, assume G is a TTSP graph, and regardless of how many vertices are in the graph, the value c will be zero. That is, no factoring theorem needs to be applied and the time complexity becomes polynomial time. In the worst case, assume G is a complete graph K_n. Let $c(K_n)$ denote the number of factoring theorem applications on K_n before K_n is decomposed into TTSP graphs, and let the vertex set V be $\{v_1, v_2, \ldots, v_n\}$. Assume we choose edge $e = (u, v)$ for applying the factoring theorem and then K_n will be decomposed into four subcases of graphs; they are $G_1 = K_n - e - u$, $G_2 = K_n - e - v$, $G_3 = K_n - e$, and $G_4 = K_n \cdot e$. Because the topology of $G_3 = K_n - e$ is more complex than the others, the problem of determining the number of factoring theorem

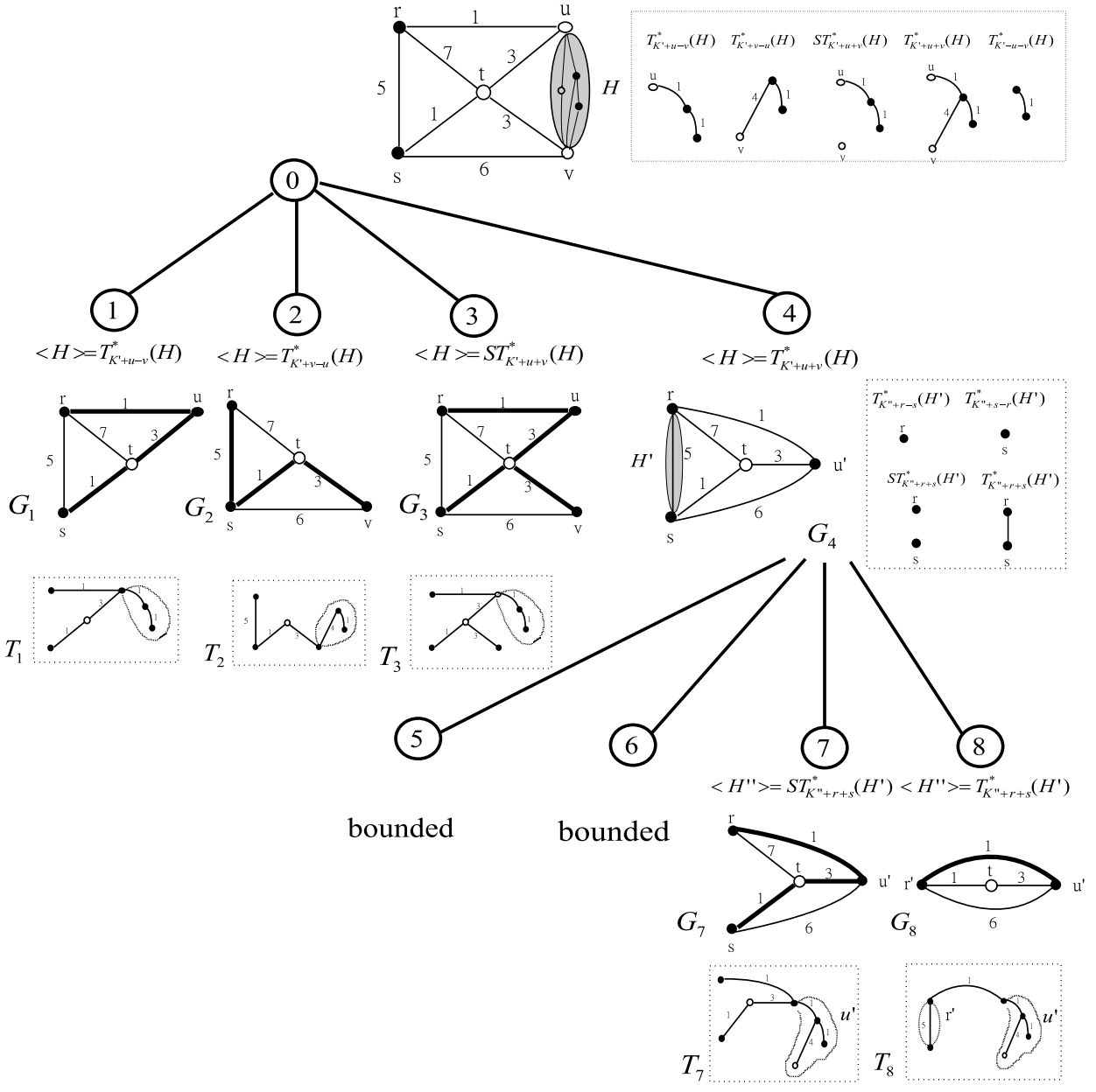

Fig. 16. The solution tree for finding minimum Steiner tree in graph H.

applications on the subcases of graph G_1, G_2, and G_4 is dominated by the subcase of G_3. Now, we will only consider this subcase. If we choose edges $(v_1, v_2), (v_1, v_3), (v_1, v_4), \ldots, (v_1, v_{n-2})$ to apply the factoring theorem iteratively, then the resulting graph will become K_{n-1}. That is, $[[\cdots[[[K_n - (v_1, v_2)] - (v_1, v_3)] - (v_1, v_4)]\cdots] - (v_1, v_{n-2})] = K_{n-1}$. Thus we have $c(K_n) = (n-3) + c(K_{n-1})$. Using the fact of $c(K_3) = 0$ to solve the above recurrence relation, we then have $c(K_n) = (n-2)(n-3)/2 = O(n^2)$, for $n \geqslant 3$. Our proposed algorithm becomes worse in such an extreme case.

Note that a better choice of a TTSP subgraph for factoring may cause the logarithmic term to become small. However, a bad choice will not. It is an important issue of our proposed method to determine which TTSP subgraph to apply the factoring theorem to, such that the value c is as small as possible. Fortunately, Bein et al. [5] showed that for any DAG (directed acyclic graph) G there exists an edges sequence $e_1, e_2, \ldots, e_{c^*}$

Algorithm 2 Factoring algorithm for general graphs;
input: a weighted graph G and a Steiner vertex set K;
output: a minimum cost tree which spans every vertex in K.
begin
if G is an SP graph then apply algorithm 1;
else begin
apply TTSP subgraphs reduction on G and let the resulting graph as $[G]$;
choose any maximal strict TTSP subgraph H with Steiner vertex set $K' \neq \emptyset$
and apply factoring theorem;
end
end

Fig. 17. Factoring algorithm for general graphs.

called a reduction sequence. If we choose the edge in the above edge sequence for factoring iteratively, then the logarithmic term c of the algorithm will be minimized. They also presented an algorithm to find the minimum reduction sequence in $O(|V|^{2.5})$. Because our considered graph G is undirected, a transformation which converts graph G into a DAG is required. The transformation procedures are given as follows. Firstly, apply the breadth-first-search on graph G, where the starting node is chosen randomly. Secondly, label each node sequentially from 1 to $|V|$ according to the order of visiting. Lastly, convert each undirected edge in graph G into a directed edge (u, v) if $\text{label}(u) < \text{label}(v)$. Obviously, the resulting graph is a DAG. Apply the algorithm of Bein et al. on the resulting graph and then an edge reduction sequence will be obtained. The edge reduction sequence will provide a good choice for factoring in our proposed method, with the consequence that the value c becomes small.

4. Concluding remarks

In this paper, we consider the Steiner tree problem for weighted graphs and propose a factoring algorithm with time complexity $O(2^{2c} \cdot |E|)$ to solve it. Comparing this algorithm's logarithmic term of time complexity with other algorithms, our algorithm is strongly related to the topology of the graph, while the others are related to the number of Steiner vertices. In the future, we will try to enhance our proposed method to minimize the c value for any given undirected graph. A distributed algorithm to solve STP will also be the focus of our future work.

References

[1] L. Bahiense, F. Barahona, O. Porto, Solving Steiner tree problems in graphs with lagrangian relaxation, Journal of Combinatorial Optimization 7 (2003) 259–282.
[2] A. Balakrishnan, N.R. Patel, Problem reduction methods and a tree generation algorithm for the Steiner network problem, Networks 17 (1987) 65–85.
[3] F. Bauer, A. Varma, Distributed algorithms for multicast path setup in data networks, IEEE/ACM Trans. on Networking 4 (1996) 181–191.
[4] J.E. Beasley, An algorithm for the Steiner problem in graphs, Networks 14 (1984) 147–159.
[5] W.W. Bein, J. Kamburowski, M.F.M. Stallmann, Optimal reduction of two-terminal directed acyclic graphs, SIAM Journal on Computing 21 (1992) 1112–1129.
[6] S.E. Drefus, R.A. Wagner, The Steiner problem in graphs, Networks 1 (1971) 195–207.
[7] M. Dror, B. Gavish, J. Choquette, Directed Steiner tree problem on a graph: models, relaxations and algorithms, INFOR 28 (1990) 266–281.
[8] R. Floren, A note on a faster approximation algorithm for the Steiner problem in graphs, Inf. Process. Lett. 38 (1991) 177–178.
[9] L.R. Foulds, P.B. Gibbons, A branch and bound approach to the Steiner problem in graphs, in: Proc. 14th Annual Conference of the Operational Research Society of New Zealand, New Zealand, 1978, pp. 61–70.
[10] M.R. Garey, D.S. Johnson, Computer and intractability: a guide to the theory of NP-completeness, Freeman and Company, San Francisco, 1979.
[11] L. Gatani, G. Lore, A. Urso, Distributed algorithms for multicast tree construction, in: Proc. of IEEE International Symposium on Control, Communications and Signal Processing, Hammamet, Tunisia, 2004, pp. 361–364.
[12] S.L. Hakimi, Steiner's problem in graphs and it's implications, Networks 1 (1971) 113–133.

[13] S. Hougardy, S. kirchner, Lower bounds for the relative greedy algorithm for approximating Steiner trees, Networks 47 (2005) 111–115.
[14] F.K. Hwang, D.S. Richards, Steiner tree problem, Networks 22 (1992) 55–89.
[15] L.J. Jia, G.L. Lin, G. Noubir, R. Rajaraman, R. Sundaram, Universal approximations for TSP, Steiner tree, and set cover, in: Proc. 37th Annual ACM Symposium on Theory of Computing, Maryland, USA, 2005, pp. 386–395.
[16] S. Kamei, H. Kakugawa, A self-stabilizing distributed algorithm for the Steiner tree problem, IEICE Transactions on Information and Systems E87-D (2004) 299–307.
[17] A.J. Levin, Algorithm for the shortest connection of a group of graph vertices, Soviet Math. Doklady 12 (1971) 1477–1481.
[18] W.G. Liu, A lower bound for the Steiner tree problem in directed graphs, Networks 20 (1990) 765–778.
[19] K. Mehlhorn, A faster approximation algorithm for the Steiner problem in graphs, Inf. Process. Lett. 27 (1988) 125–128.
[20] D. Molle, S. Richter, P. Rossmanith, A faster algorithm for the Steiner tree problem, in: Proc. 23th International Symposium on Theoretical Aspects of Computer Science, Marseille, France, 2006, pp. 561–570.
[21] A. Prodon, T.M. Liebling, H. Groflin, Steiner's problem on two-trees, Tech. Rep. RO 850315, Dept. de Math. Ecole Polytechnique Federale de Lausanne, Suisse, 1985.
[22] H.J. Promel, A. Steger, A new approximation algorithm for the Steiner tree problem with performance ratio 5/3, Journal of Algorithms 36 (2000) 89–101.
[23] V.J. Rayward-Smith, A. Clare, On finding Steiner vertices, Networks 16 (1986) 283–294.
[24] G. Robins, A. Zelikovsky, Improved Steiner tree approximation in graphs, in: Proc. 11th Annual ACM-SIAM Symposium on Discrete Algorithms, San Francisco, USA, 2000, pp. 770–779.
[25] G. Robins, A. Zelikovsky, Tighter bounds for graph Steiner tree approximations, SIAM Journal on Discrete Mathematics 19 (2005) 122–134.
[26] M.P. Saltouros, E.A. Verentziotis, M.E. Markaki, M.E. Theologou, I.S. Venieris, Efficient hybrid genetic algorithm for finding (near) optimal Steiner trees: an approach to routing of multipoint connections, International Journal of Computers and Applications 22 (2000) 159–165.
[27] M.L. Shore, L.R. Foulds, P.B. Gibbons, An algorithm for the Steiner problem in graphs, Networks 12 (1982) 323–333.
[28] H. Takahashi, A. Matsuyama, An approximate solution for the Steiner problem in graphs, Math. Jap. 24 (1980) 573–577.
[29] J.A. Wald, C.J. Colbourn, Steiner trees, partial 2-trees and minimum IFI networks, Networks 13 (1983) 159–167.
[30] P. Winter, Steiner problem in networks: A survey, Networks 17 (1987) 129–167.
[31] P. Winter, J.M. Smith, Path-distance heuristics for the Steiner problem in undirected networks, Algorithmica 7 (1992) 309–327.
[32] R.T. Wong, A dual ascent approach for Steiner tree problems on a directed graphs, Math. Programming 28 (1984) 271–287.
[33] Y.Y. Yang, O. Wing, An algorithm for the wiring problem, in: Digest of the IEEE Int. Symp. on Electrical Networks, 1971, pp. 14–15.
[34] J. Zhao, C.H. Chen, Performance of probabilistic approaches to Steiner tree prolem, in: Proc. 2nd Annual IEEE Northeast Workshop on Circuits and Systems, Montreal, Canada, 2004, pp. 125–128.
[35] X.W. Zhou, C.J. Chen, G. Zhu, A genetic algorithm for multicasting routing problem, in: Proc. of the International Conference on Communication Technology, Beijing, China, 2000, pp. 1248–1253.