Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Discrete Algorithms 4 (2006) 142–154
JOURNAL OF DISCRETE ALGORITHMS
www.elsevier.com/locate/jda

Partitioning a graph of bounded tree-width to connected subgraphs of almost uniform size

Takehiro Ito *, Xiao Zhou, Takao Nishizeki

Graduate School of Information Sciences, Tohoku University, Aoba-yama 6-6-05, Sendai, 980-8579, Japan

Available online 29 January 2005

Abstract

Assume that each vertex of a graph G is assigned a nonnegative integer weight and that l and u are nonnegative integers. One wishes to partition G into connected components by deleting edges from G so that the total weight of each component is at least l and at most u. Such an "almost uniform" partition is called an (l, u)-partition. We deal with three problems to find an (l, u)-partition of a given graph; the minimum partition problem is to find an (l, u)-partition with the minimum number of components; the maximum partition problem is defined analogously; and the p-partition problem is to find an (l, u)-partition with a fixed number p of components. All these problems are NP-complete or NP-hard, respectively, even for series-parallel graphs. In this paper we show that both the minimum partition problem and the maximum partition problem can be solved in time $O(u^4 n)$ and the p-partition problem can be solved in time $O(p^2 u^4 n)$ for any series-parallel graph with n vertices. The algorithms can be extended for partial k-trees, that is, graphs with bounded tree-width.

Keywords: Algorithm; Lower bound; (l, u)-partition; Maximum partition problem; Minimum partition problem; Partial k-tree; Series-parallel graph; Upper bound

* Corresponding author.
E-mail addresses: take@nishizeki.ecei.tohoku.ac.jp (T. Ito), zhou@ecei.tohoku.ac.jp (X. Zhou), nishi@ecei.tohoku.ac.jp (T. Nishizeki).

1570-8667/$ – see front matter
doi:10.1016/j.jda.2005.01.005

1. Introduction

Let $G = (V, E)$ be an undirected graph with vertex set V and edge set E, and let $|V| = n$. Assume that each vertex $v \in V$ is assigned a nonnegative integer $\omega(v)$, called the *weight* of v. Let l and u be nonnegative integers, called the *lower bound* and *upper bound* on component size, respectively. We wish to partition G into connected components by deleting edges from G so that the total weights of all components are almost uniform, that is, the sum of weights of all vertices in each component is at least l and at most u for some bounds l and u. We call such an almost uniform partition an (l, u)-*partition* of G. In this paper we deal with the following three partition problems to find an (l, u)-partition of a given graph G: the *minimum partition problem* is to find an (l, u)-partition of G with the minimum number of components; the minimum number is denoted by $p_{\min}(G)$; the *maximum partition problem* is defined analogously; and the *p-partition problem* is to find an (l, u)-partition of G with a fixed number p of components.

Figs. 1(a) and (b) illustrate two $(10, 20)$-partitions of the same graph, where each vertex is drawn as a circle, the weight of each vertex is written inside the circle, and the deleted edges are drawn as dotted lines. The $(10, 20)$-partition with four components in Fig. 1(a) is a solution for the minimum partition problem, and hence $p_{\min}(G) = 4$ for the graph G in Fig. 1(a). The $(10, 20)$-partition with six components in Fig. 1(b) is a solution for the maximum partition problem.

The three partition problems often appear in many practical situations such as with image processing [5,8], paging systems of operation systems [11], and political districting [3,12]. Consider, for example, political districting. Let M be a map of a country, which is divided into several regions. Let G be the dual graph of the map M. Each vertex v of G represents a region, and let the weight $\omega(v)$ represent the number of voters in region v. Each edge (u, v) of G represents the adjacency of the two regions u and v. For the political districting, one wishes to divide the country into electoral zones. Each zone must consist of connected regions, that is, the regions in each zone must induce a connected subgraph of G. There must be an almost equal number of voters in each zone, that is, the sum of $\omega(v)$ for all regions v in each zone is at least l and at most u for some bounds l and u. Such electoral zoning corresponds to an (l, u)-partition of the plane graph G.

Two related problems have been studied for trees. One is to partition a tree into the maximum number of subtrees so that the total weight of each subtree is at least l [9]. The other is to partition a tree into the minimum number of subtrees so that the total weight of

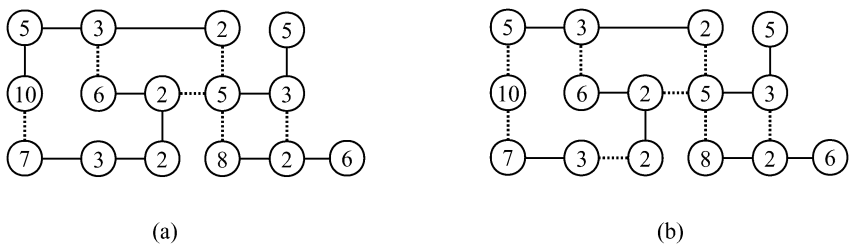

Fig. 1. (a) Solution for the minimum partition problem, and (b) solution for the maximum partition problem, where $l = 10$ and $u = 20$.

each subtree is at most u [7]. Both can be solved for trees in linear time. Our three partition problems are generalizations of these problems. One may expect that there would exist efficient algorithms for the three partition problems on trees, but our problems are more difficult than the two problems in [7,9], except for paths; all the three partition problems can be solved for paths in linear time [8].

An NP-complete problem called the set partition problem PARTITION [4] can be easily reduced in linear time to our problems for a complete bipartite graph $K_{2,n-2}$, and $K_{2,n-2}$ is a series-parallel graph. (A definition of a series-parallel graph will be given in Section 2.) Therefore, the p-partition problem for general p is NP-complete and both the minimum partition problem and the maximum partition problem for general l and u are NP-hard even for series-parallel graphs. Hence, it is very unlikely that the three partition problems can be solved for series-parallel graphs in polynomial time, although a number of combinatorial problems including many NP-complete problems on general graphs can be solved for series-parallel graphs and partial k-trees in polynomial time or even in linear time [1,2,10]. One can also observe from the above reduction that, for any $\varepsilon > 0$, there is no polynomial-time ε-approximation algorithm for the minimum partition problem or the maximum partition problem on series-parallel graphs unless P = NP.

A strong NP-complete problem called 3-PARTITION [4] can be easily reduced in pseudo-polynomial time to our problems for a complete graph. Hence the p-partition problem is strong NP-complete and both the minimum partition problem and the maximum partition problem are strong NP-hard for general graphs. Therefore, there is no pseudo-polynomial-time algorithm for any of our three problems for general graphs unless P = NP.

In this paper we first obtain pseudo-polynomial-time algorithms to solve the three partition problems for series-parallel graphs. More precisely, we show that both the minimum partition problem and the maximum partition problem can be solved in time $O(u^4 n)$ and hence in time $O(n)$ for any bounded constant u, and that the p-partition problem can be solved in time $O(p^2 u^4 n)$. We then show that our algorithms can be extended for partial k-trees, that is, graphs with bounded tree-width [1,2]. (A definition of a partial k-tree will be given in Section 5.) An early version of the paper has been presented at [6].

2. Terminology and definitions

In this section we give some definitions.

A (*two-terminal*) *series-parallel graph* is defined recursively as follows [10]:

(1) A graph G with a single edge is a series-parallel graph. The end vertices of the edge are called the *terminals* of G and denoted by $s(G)$ and $t(G)$. (See Fig. 2(a).)
(2) Let G' be a series-parallel graph with terminals $s(G')$ and $t(G')$, and let G'' be a series-parallel graph with terminals $s(G'')$ and $t(G'')$.
   (a) A graph G obtained from G' and G'' by identifying vertex $t(G')$ with vertex $s(G'')$ is a series-parallel graph, whose terminals are $s(G) = s(G')$ and $t(G) = t(G'')$. Such a connection is called a *series connection*, and G is denoted by $G = G' \bullet G''$. (See Fig. 2(b).)

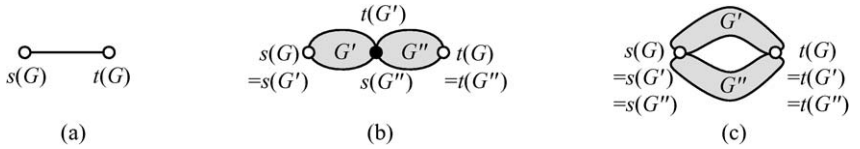

Fig. 2. (a) A series-parallel graph with a single edge, (b) series connection, and (c) parallel connection.

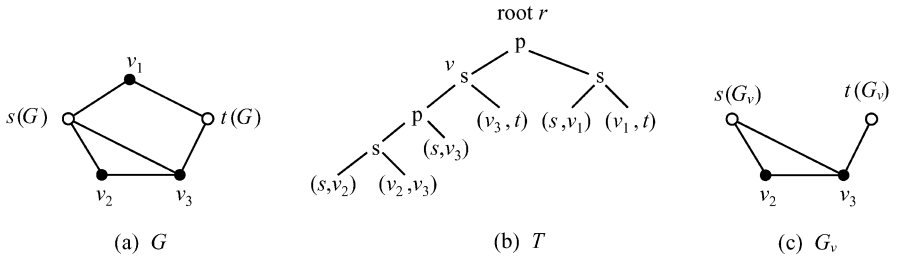

Fig. 3. (a) A series-parallel graph G, (b) its binary decomposition tree T, and (c) a subgraph G_v.

(b) A graph G obtained from G' and G'' by identifying $s(G')$ with $s(G'')$ and identifying $t(G')$ with $t(G'')$ is a series-parallel graph, whose terminals are $s(G) = s(G') = s(G'')$ and $t(G) = t(G') = t(G'')$. Such a connection is called a *parallel connection*, and G is denoted by $G = G' \parallel G''$. (See Fig. 2(c).)

The terminals $s(G)$ and $t(G)$ of G are often denoted simply by s and t, respectively. Since we deal with partition problems, we may assume without loss of generality that G is a simple graph and hence G has no multiple edges.

A series-parallel graph G can be represented by a "binary decomposition tree" [10]. Fig. 3(a) illustrates a series-parallel graph G, and Fig. 3(b) depicts a binary decomposition tree T of G. Labels s and p attached to internal nodes in T indicate series and parallel connections, respectively. Nodes labeled s and p are called *s- and p-nodes*, respectively. Every leaf of T represents a subgraph of G induced by a single edge. Each node v of T corresponds to a subgraph G_v of G induced by all edges represented by the leaves that are descendants of v in T. Thus G_v is a series-parallel graph for each node v of T, and $G = G_r$ for the root r of T. Fig. 3(c) depicts G_v for the left child v of the root r of T. Since a binary decomposition tree of a given series-parallel graph G can be found in linear time [10], we may assume that a series-parallel graph G and its binary decomposition tree T are given. We solve the three partition problems by a dynamic programming approach based on a binary decomposition tree T.

3. Minimum and maximum partition problems

In this section we have the following theorem.

(a) (b)

Fig. 4. (a) A connected partition, and (b) a separated partition.

Theorem 1. *Both the minimum partition problem and the maximum partition problem can be solved for any series-parallel graph G in time $O(u^4 n)$, where n is the number of vertices in G and u is the upper bound on component size.*

In the remainder of this section we give an algorithm to solve the minimum partition problem as a proof of Theorem 1; the maximum partition problem can be analogously solved. We indeed show only how to compute the minimum number $p_{\min}(G)$. It is easy to modify our algorithm so that it actually finds an (l, u)-partition having the minimum number $p_{\min}(G)$ of components.

Every (l, u)-partition of a series-parallel graph G naturally induces a partition of its subgraph G_v for a node v of a binary decomposition tree T of G. The induced partition is not always an (l, u)-partition of G_v but is either a "connected partition" or a "separated partition" of G_v, which will be formally defined later and are illustrated in Fig. 4 where s and t represent the terminals of G_v. We introduce two functions $f : (\mathcal{G}, \{0, 1, \ldots, u\}) \to \mathbb{Z}^+$ and $h : (\mathcal{G}, \{0, 1, \ldots, u\}, \{0, 1, \ldots, u\}) \to \mathbb{Z}^+$, where $\mathcal{G}$ denotes the set of all series-parallel graphs and $\mathbb{Z}^+$ denotes the set of all nonnegative integers. For $G_v \in \mathcal{G}$ and $x, y \in \{0, 1, \ldots, u\}$, the values $f(G_v, x)$ and $h(G_v, x, y)$ represent the minimum number of components without terminals in connected partitions and separated partitions of G_v, respectively, and x and y represent the total weight of non-terminal vertices in a component with a terminal. Our idea is to compute $f(G_v, x)$ and $h(G_v, x, y)$ from the leaves of T to the root r of T by means of dynamic programming.

We now formally define the notion of connected and separated partitions of a series-parallel graph $G = (V, E)$. Let $\mathcal{P} = \{P_1, P_2, \ldots, P_m\}$ be a partition of the vertex set V of G into m nonempty subsets $P_1, P_2, \ldots, P_m$ for some integer $m \geqslant 1$. Thus $|\mathcal{P}| = m$. The partition $\mathcal{P}$ of V is called a *partition of* G if P_i induces a connected subgraph of G for each index i, $1 \leqslant i \leqslant m$. For a set $P \subseteq V$, we denote by $\omega(P)$ the total weight of vertices in P, that is, $\omega(P) = \sum_{v \in P} \omega(v)$. Let $\omega_{st}(G) = \omega(s) + \omega(t)$. We call a partition $\mathcal{P}$ of G a *connected partition* if $\mathcal{P}$ satisfies the following two conditions (see Fig. 4(a)):

(a) there exists a set $P_{st} \in \mathcal{P}$ such that $s, t \in P_{st}$ and $\omega(P_{st}) \leqslant u$; and
(b) $l \leqslant \omega(P) \leqslant u$ for each set $P \in \mathcal{P} - \{P_{st}\}$.

Note that the inequality $l \leqslant \omega(P_{st})$ does not necessarily hold for P_{st}. For a connected partition $\mathcal{P}$, we always denote by P_{st} the set in $\mathcal{P}$ containing both s and t. A partition $\mathcal{P}$ of G is called a *separated partition* if $\mathcal{P}$ satisfies the following two conditions (see Fig. 4(b)):

(a) there exist two distinct sets $P_s, P_t \in \mathcal{P}$ such that $s \in P_s$, $t \in P_t$, $\omega(P_s) \leqslant u$, and $\omega(P_t) \leqslant u$; and
(b) $l \leqslant \omega(P) \leqslant u$ for each set $P \in \mathcal{P} - \{P_s, P_t\}$.

Note that the inequalities $l \leqslant \omega(P_s)$ and $l \leqslant \omega(P_t)$ do not always hold for P_s and P_t. For a separated partition $\mathcal{P}$, we always denote by P_s the set in $\mathcal{P}$ containing s and by P_t the set in $\mathcal{P}$ containing t.

We then formally define a function $f : (\mathcal{G}, \{0, 1, \ldots, u\}) \to \mathbb{Z}^+$ for a series-parallel graph $G \in \mathcal{G}$ and an integer x, $0 \leqslant x \leqslant u$, as follows:

$$f(G, x) = \min\{q \geqslant 0 \mid G \text{ has a connected partition } \mathcal{P} \text{ such that } x = \omega(P_{st}) - \omega_{st}(G), \text{ and } q = |\mathcal{P}| - 1\}. \tag{1}$$

If G has no connected partition $\mathcal{P}$ such that $\omega(P_{st}) - \omega_{st}(G) = x$, then let $f(G, x) = +\infty$. We now formally define a function $h : (\mathcal{G}, \{0, 1, \ldots, u\}, \{0, 1, \ldots, u\}) \to \mathbb{Z}^+$ for a series-parallel graph $G \in \mathcal{G}$ and a pair of integers x and y, $0 \leqslant x, y \leqslant u$, as follows:

$$h(G, x, y) = \min\{q \geqslant 0 \mid G \text{ has a separated partition } \mathcal{P} \text{ such that } x = \omega(P_s) - \omega(s) \text{ and } y = \omega(P_t) - \omega(t), \text{ and } q = |\mathcal{P}| - 2\}. \tag{2}$$

If G has no separated partition $\mathcal{P}$ such that $\omega(P_s) - \omega(s) = x$ and $\omega(P_t) - \omega(t) = y$, then let $h(G, x, y) = +\infty$.

Our algorithm computes $f(G_v, x)$ and $h(G_v, x, y)$ for each node v of a binary decomposition tree T of a given series-parallel graph G from the leaves to the root r of T by means of dynamic programming. Since $G = G_r$, one can compute the minimum number $p_{\min}(G)$ of components from $f(G, x)$ and $h(G, x, y)$ as follows:

$$p_{\min}(G) = \min\{\min\{f(G, x) + 1 \mid l \leqslant x + \omega_{st}(G) \leqslant u\}, \min\{h(G, x, y) + 2 \mid l \leqslant x + \omega(s) \leqslant u,\ l \leqslant y + \omega(t) \leqslant u\}\}. \tag{3}$$

Note that $p_{\min}(G) = +\infty$ if G has no (l, u)-partition.

We first compute $f(G_v, x)$ and $h(G_v, x, y)$ for each leaf v of T, for which the subgraph G_v contains exactly one edge. For $x = 0$

$$f(G_v, 0) = 0, \tag{4}$$

and for $(x, y) = (0, 0)$

$$h(G_v, 0, 0) = 0. \tag{5}$$

For each integer x, $1 \leqslant x \leqslant u$,

$$f(G_v, x) = +\infty, \tag{6}$$

and for each pair (x, y), $1 \leqslant x, y \leqslant u$,

$$h(G_v, x, y) = +\infty. \tag{7}$$

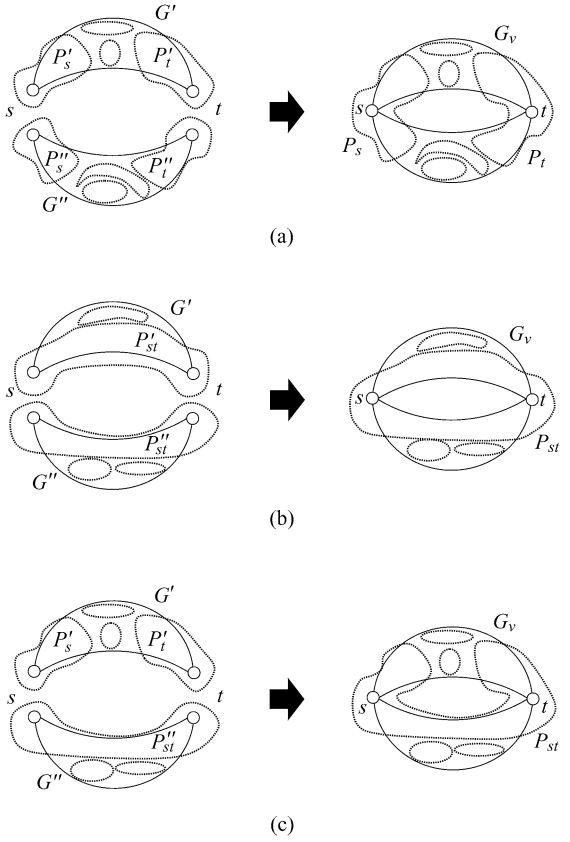

Fig. 5. The combinations of a partition $\mathcal{P}'$ of G' and a partition $\mathcal{P}''$ of G'' for a partition $\mathcal{P}$ of $G_v = G' \parallel G''$.

By Eqs. (4)–(7) one can compute $f(G_v, x)$ in time $O(u)$ for each leaf v of T and all integers $x \leqslant u$, and compute $h(G_v, x, y)$ in time $O(u^2)$ for each leaf v and all pairs (x, y) with $x, y \leqslant u$. Since G is a simple series-parallel graph, the number of edges in G is at most $2n - 3$ and hence the number of leaves in T is at most $2n - 3$. Thus one can compute $f(G_v, x)$ and $h(G_v, x, y)$ for all leaves v of T in time $O(u^2 n)$.

We next compute $f(G_v, x)$ and $h(G_v, x, y)$ for each internal node v of T from the counterparts of the two children of v in T. We first consider a parallel connection. Let $G_v = G' \parallel G''$, and let $s = s(G_v)$ and $t = t(G_v)$. (See Figs. 2(c) and 5.)

We first explain how to compute $h(G_v, x, y)$. The definitions of a separated partition and $h(G, x, y)$ imply that if $\omega(P_s) = x + \omega(s) > u$ or $\omega(P_t) = y + \omega(t) > u$ then $h(G_v, x, y) = +\infty$. One may thus assume that $x + \omega(s) \leqslant u$ and $y + \omega(t) \leqslant u$. Then every separated partition $\mathcal{P}$ of G_v can be obtained by combining any separated partition $\mathcal{P}'$ of G' with any separated partition $\mathcal{P}''$ of G'', as illustrated in Fig. 5(a). We thus have

$$h(G_v, x, y) = \min\bigl\{h(G', x', y') + h(G'', x - x', y - y') \mid 0 \leqslant x', y' \leqslant u\bigr\}. \tag{8}$$

We next explain how to compute $f(G_v, x)$. If $\omega(P_{st}) = x + \omega_{st}(G_v) > u$, then $f(G_v, x) = +\infty$. One may thus assume that $x + \omega_{st}(G_v) \leqslant u$. Then every connected partition $\mathcal{P}$ of G_v can be obtained by combining a partition $\mathcal{P}'$ of G' with a partition $\mathcal{P}''$ of G'', as illustrated in Figs. 5(b) and (c). There are the following two Cases (a) and (b), and we define two functions f^a and f^b for the two cases, respectively.

Case (*a*): *both $\mathcal{P}'$ and $\mathcal{P}''$ are connected partitions.* (See Fig. 5(b).)

Let

$$f^a(G_v, x) = \min\bigl\{f(G', x') + f(G'', x - x') \mid 0 \leqslant x' \leqslant u\bigr\}. \tag{9}$$

Case (*b*): *one of $\mathcal{P}'$ and $\mathcal{P}''$ is a separated partition and the other is a connected partition.*

One may assume without loss of generality that $\mathcal{P}'$ is a separated partition and $\mathcal{P}''$ is a connected partition. (See Fig. 5(c).) Let

$$f^b(G_v, x) = \min\bigl\{h(G', x', y') + f(G'', x - x' - y') \mid 0 \leqslant x', y' \leqslant u\bigr\}. \tag{10}$$

From f^a and f^b above, one can compute $f(G_v, x)$ as follows:

$$f(G_v, x) = \min\bigl\{f^a(G_v, x),\, f^b(G_v, x)\bigr\}. \tag{11}$$

By Eq. (8) one can compute $h(G_v, x, y)$ for all pairs $(x, y), 0 \leqslant x, y \leqslant u$, in time $O(u^4)$, and by Eqs. (9)–(11) one can compute $f(G_v, x)$ for all integers x, $0 \leqslant x \leqslant u$, in time $O(u^3)$. Thus one can compute $f(G_v, x)$ and $h(G_v, x, y)$ for each p-node v of T in time $O(u^4)$.

We next consider a series connection. Let $G_v = G' \bullet G''$, and let w be the vertex of G identified by the series connection, that is, $w = t(G') = s(G'')$. (See Figs. 2(b) and 6.)

We first explain how to compute $f(G_v, x)$. If $x + \omega_{st}(G_v) > u$, then $f(G_v, x) = +\infty$. One may thus assume that $x + \omega_{st}(G_v) \leqslant u$. Then every connected partition $\mathcal{P}$ of G_v can be obtained by combining a connected partition $\mathcal{P}'$ of G' with a connected partition $\mathcal{P}''$ of G'', as illustrated in Fig. 6(a). We thus have

$$f(G_v, x) = \min\bigl\{f(G', x') + f(G'', x'') \mid 0 \leqslant x', x'' \leqslant u,\ x' + x'' + \omega(w) = x\bigr\}. \tag{12}$$

We next explain how to compute $h(G_v, x, y)$. If $x + \omega(s) > u$ or $y + \omega(t) > u$, then $h(G_v, x, y) = +\infty$. One may thus assume that $x + \omega(s) \leqslant u$ and $y + \omega(t) \leqslant u$. Then every separated partition $\mathcal{P}$ of G_v can be obtained by combining a partition $\mathcal{P}'$ of G' with a partition $\mathcal{P}''$ of G'', as illustrated in Figs. 6(b) and (c). There are the following two cases (a) and (b), and we define two functions h^a and h^b for the two cases, respectively.

Case (*a*): *one of $\mathcal{P}'$ and $\mathcal{P}''$ is a connected partition and the other is a separated partition.*

One may assume without loss of generality that $\mathcal{P}'$ is a connected partition and $\mathcal{P}''$ is a separated partition. (See Fig. 6(b).) Let

$$h^a(G_v, x, y) = \min\bigl\{f(G', x') + h(G'', x'', y) \mid 0 \leqslant x', x'' \leqslant u,\ x' + x'' + \omega(w) = x\bigr\}. \tag{13}$$

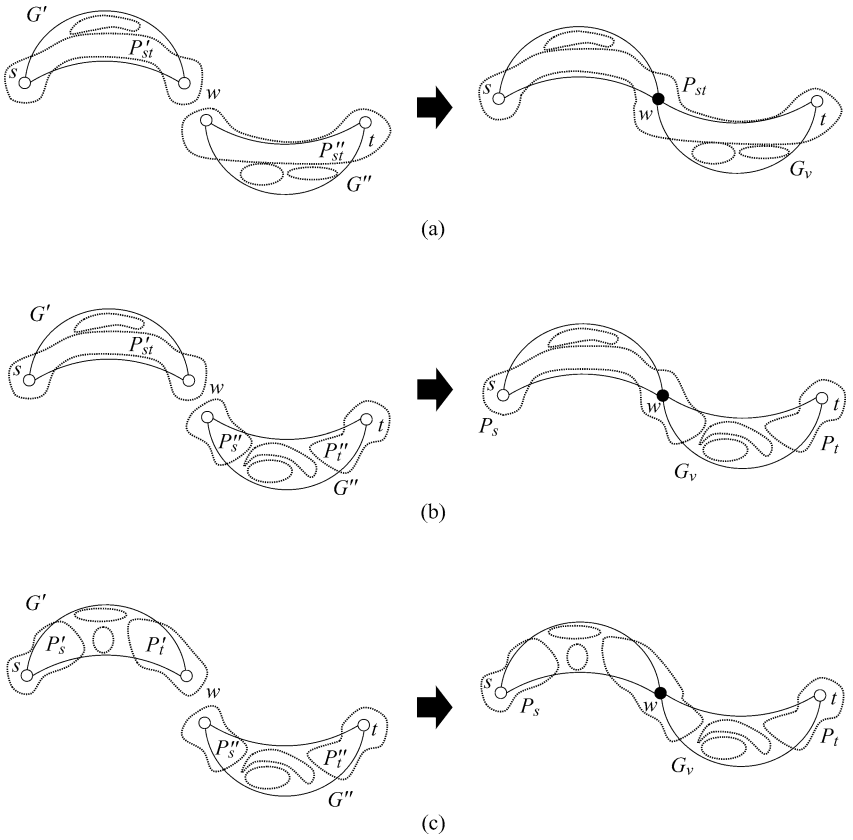

Fig. 6. The combinations of a partition $\mathcal{P}'$ of G' and a partition $\mathcal{P}''$ of G'' for a partition $\mathcal{P}$ of $G_v = G' \bullet G''$.

Case (*b*): *both* $\mathcal{P}'$ *and* $\mathcal{P}''$ *are separated partitions.* (See Fig. 6(c).)

Let

$$h^b(G_v, x, y) = \min\{h(G', x, y') + h(G'', x'', y) + 1 \mid 0 \leqslant y', x'' \leqslant u, \\ l \leqslant y' + x'' + \omega(w) \leqslant u\}. \tag{14}$$

From h^a and h^b above one can compute $h(G_v, x, y)$ as follows:

$$h(G_v, x, y) = \min\{h^a(G_v, x, y), h^b(G_v, x, y)\}. \tag{15}$$

By Eq. (12) one can compute $f(G_v, x)$ for all integers x, $0 \leqslant x \leqslant u$, in time $O(u^2)$, and by Eqs. (13)–(15) one can compute $h(G_v, x, y)$ for all pairs (x, y), $0 \leqslant x, y \leqslant u$, in time $O(u^4)$. Thus one can compute $f(G_v, x)$ and $h(G_v, x, y)$ for each s-node v of T in time $O(u^4)$.

In this way one can compute $f(G_v, x)$ and $h(G_v, x, y)$ for each internal node v of T in time $O(u^4)$. Since T is a binary tree and has at most $2n - 3$ leaves, T has at most $2n - 4$ internal nodes. Since $G = G_r$ for the root r of T, one can compute $f(G, x)$ and $h(G, x, y)$

in time $O(u^4 n)$. By Eq. (3) one can compute the minimum number $p_{\min}(G)$ of components in an (l, u)-partition of G from $f(G, x)$ and $h(G, x, y)$ in time $O(u^2)$. Thus the minimum partition problem can be solved in time $O(u^4 n)$. This completes our proof of Theorem 1.

4. p-partition problem

In this section we have the following theorem.

Theorem 2. *The p-partition problem can be solved for any series-parallel graph G in time $O(p^2 u^4 n)$, where n is the number of vertices in G, u is the upper bound on component size, and p is the fixed number of components.*

The algorithm for the p-partition problem is analogous to the algorithm for the minimum partition problem in the previous section. So we present only an outline.

For a series-parallel graph G and an integer q, $0 \leqslant q \leqslant p-1$, we define a set $F(G, q)$ of nonnegative integers x as follows:

$$F(G,q) = \big\{x \geqslant 0 \mid G \text{ has a connected partition } \mathcal{P} \text{ such that } x = \omega(P_{st}) - \omega_{st}(G), \text{ and } q = |\mathcal{P}| - 1\big\}.$$

For a series-parallel graph G and an integer q, $0 \leqslant q \leqslant p-2$, we define a set $H(G, q)$ of pairs of nonnegative integers x and y as follows:

$$H(G,q) = \big\{(x,y) \mid G \text{ has a separated partition } \mathcal{P} \text{ such that } x = \omega(P_s) - \omega(s) \text{ and } y = \omega(P_t) - \omega(t), \text{ and } q = |\mathcal{P}| - 2\big\}.$$

Clearly $|F(G, q)| \leqslant u + 1$ and $|H(G, q)| \leqslant (u + 1)^2$.

We compute $F(G_v, q)$ and $H(G_v, q)$ for each node v of a binary decomposition tree T of a given series-parallel graph G from the leaves to the root r of T by means of dynamic programming. Since $G = G_r$, the following lemma clearly holds.

Lemma 1. *A series-parallel graph G has an (l, u)-partition with p components if and only if one of the following conditions* (a) *and* (b) *holds*:

(a) $F(G, p-1)$ *contains at least one integer x such that $l \leqslant x + \omega_{st}(G) \leqslant u$; and*
(b) $H(G, p-2)$ *contains at least one pair of integers x and y such that $l \leqslant x + \omega(s) \leqslant u$ and $l \leqslant y + \omega(t) \leqslant u$.*

One can compute in time $O(p)$ the sets $F(G_v, q)$ and $H(G_v, q)$ for each leaf v of T and all integers $q (\leqslant p-1)$, and compute in time $O(p^2 u^4)$ the sets $F(G_v, q)$ and $H(G_v, q)$ for each internal node v of T and all integers $q (\leqslant p-1)$ from the counterparts of the two children of v in T. Since $G = G_r$ for the root r of T, one can compute the sets $F(G, p-1)$ and $H(G, p-2)$ in time $O(p^2 u^4 n)$. By Lemma 1 one can know from the sets in time $O(u^2)$ whether G has an (l, u)-partition with p components. Thus the p-partition problem can be solved in time $O(p^2 u^4 n)$.

5. Partial k-trees

In this section we have the following theorem.

Theorem 3. *The minimum and maximum partition problems can be solved in time* $O(u^{2(k+1)}n)$ *and the p-partition problem can be solved in time* $O(p^2u^{2(k+1)}n)$ *for any partial k-tree, where k is a constant.*

The algorithm for partial k-trees is analogous to those for series-parallel graphs in the previous sections. So we present only an outline of the algorithm for the minimum partition problem.

A graph G is a *k-tree* if either it is a complete graph on k vertices or it has a vertex v whose neighbors induce a clique of size k and $G - \{v\}$ is again a k-tree. A graph is a *partial k-tree* if it is a subgraph of a k-tree.

A series-parallel graph is a partial 2-tree. A partial k-tree G can be decomposed into pieces forming a tree structure with at most $k + 1$ vertices per piece. (See Fig. 7(b).) The tree structure is called a binary decomposition tree T of G [1,2]. Each node v of T corresponds to a set $V(v)$ of $k + 1$ or fewer vertices of G, and corresponds to a subgraph G_v of G. For example, Fig. 7 illustrates a partial 3-tree G, its binary decomposition tree T, and a subgraph G_{x_1} of G for a node x_1 of T.

For a series-parallel graph it suffices to consider only two kinds of partitions, a connected partition and a separated partition, while for a partial k-tree we have to consider many kinds of partitions of G_v. Let π be the number of all partitions of set $V(v)$ into pairwise disjoint nonempty subsets. Then $\pi \leqslant (2^{k+1})^{k+1}$ and hence π is a constant since we assume k is a constant in the paper. For a partial k-tree G, we consider π kinds of partitions

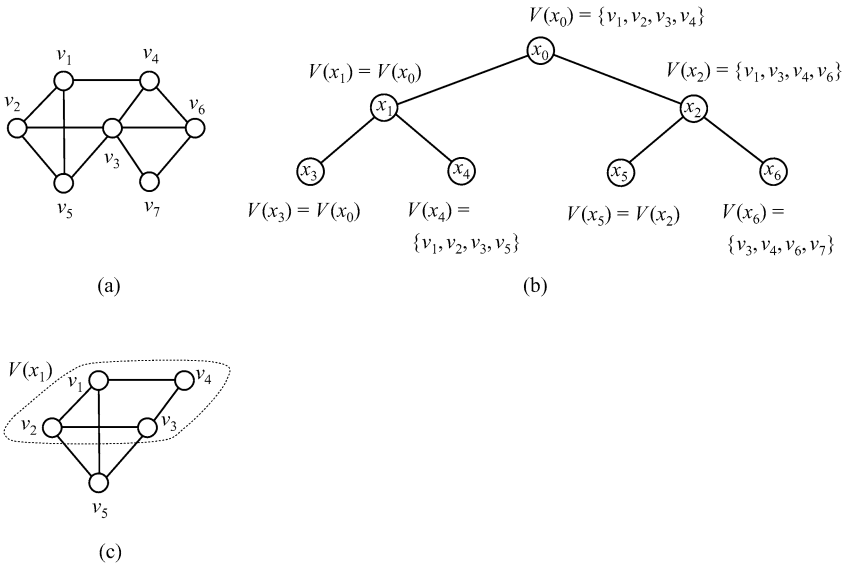

Fig. 7. (a) A partial 3-tree G, (b) its binary decomposition tree T, and (c) a subgraph G_{x_1}.

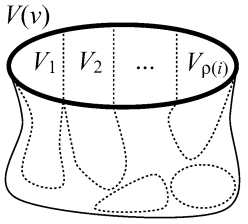

Fig. 8. The partition of G_v of ith kind.

of G_v. Let $\mathcal{V}_i$, $1 \leqslant i \leqslant \pi$, be the ith partition of set $V(v)$, let $\rho(i)$ be the number of subsets in the partition $\mathcal{V}_i$, and let $\mathcal{V}_i = \{V_1, V_2, \ldots, V_{\rho(i)}\}$. Clearly $1 \leqslant \rho(i) \leqslant k+1$. In every partition of G_v of the ith kind, its jth connected component, $1 \leqslant j \leqslant \rho(i)$, contains all the vertices in the jth subset $V_j (\subseteq V(v))$ in $\mathcal{V}_i$. (See Fig. 8.) We consider a set of functions $h_i(G_v, x_1, x_2, \ldots, x_{\rho(i)})$, $1 \leqslant i \leqslant \pi$, defined analogously to Eqs. (1) and (2). Variable x_j, $1 \leqslant j \leqslant \rho(i)$, represents the sum of weights of all vertices in the jth component except for the vertices in V_j. Thus $0 \leqslant x_j \leqslant u$. One can observe that the set of functions for G_v for an internal node v can be computed from the counterparts of the two children of v in T in time $O((u+1)^{2(k+1)})$. Thus the set of functions for G can be computed in time $O((u+1)^{2(k+1)}n)$. The hidden coefficient in the complexity is $\pi^2 (\leqslant 2^{2(k+1)^2})$.

6. Conclusions

In this paper we first obtained pseudo-polynomial-time algorithms for three partition problems on series-parallel graphs. Both the minimum partition problem and the maximum partition problem can be solved in time $O(u^4 n)$, and hence they can be solved in time $O(n)$ if $u = O(1)$. On the other hand, the p-partition problem can be solved in time $O(p^2 u^4 n)$. Thus these algorithms take polynomial time if u is bounded by a polynomial in n.

We then showed that our algorithms for series-parallel graphs can be extended for partial k-trees, that is, graphs of bounded tree-width. The extended algorithm takes time $O(u^{2(k+1)}n)$ for the minimum and maximum partition problems, and takes time $O(p^2 u^{2(k+1)}n)$ for the p-partition problem.

We finally remark that, for ordinary trees, one can solve the minimum and maximum partition problems in time $O(u^2 n)$ and the p-partition problem in time $O(p^2 u^2 n)$ or $O\big(n + \binom{n}{p-1}\big)$.

Acknowledgement

We thank Takeshi Tokuyama for suggesting us the partition problems.

References

[1] S. Arnborg, J. Lagergren, D. Seese, Easy problems for tree-decomposable graphs, J. Algorithms 12 (2) (1991) 308–340.
[2] H.L. Bodlaender, Polynomial algorithms for graph isomorphism and chromatic index on partial k-trees, J. Algorithms 11 (4) (1990) 631–643.
[3] B. Bozkaya, E. Erkut, G. Laporte, A tabu search heuristic and adaptive memory procedure for political districting, European J. Oper. Res. 144 (1) (2003) 12–26.
[4] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, San Francisco, CA, 1979.
[5] R.C. Gonzalez, P. Wintz, Digital Image Processing, Addison-Wesley, Reading, MA, 1977.
[6] T. Ito, X. Zhou, T. Nishizeki, Partitioning a weighted graph to connected subgraphs of almost uniform size, in: Proc. WG2004, Lecture Notes in Computer Science, vol. 3353, 2004, pp. 365–376.
[7] S. Kundu, J. Misra, A linear tree partitioning algorithm, SIAM J. Comput. 6 (1) (1977) 151–154.
[8] M. Lucertini, Y. Perl, B. Simeone, Most uniform path partitioning and its use in image processing, Discrete Appl. Math. 42 (2) (1993) 227–256.
[9] Y. Perl, S.R. Schach, Max-min tree partitioning, J. ACM 28 (1) (1981) 5–15.
[10] K. Takamizawa, T. Nishizeki, N. Saito, Linear-time computability of combinatorial problems on series-parallel graphs, J. ACM 29 (3) (1982) 623–641.
[11] D.C. Tsichritzis, P.A. Bernstein, Operating Systems, Academic Press, New York, 1974.
[12] J.C. Williams Jr., Political redistricting: a review, Papers in Regional Science 74 (1) (1995) 13–40.