Partitioning a Weighted Graph to Connected Subgraphs of Almost Uniform Size

Takehiro Ito, Xiao Zhou, and Takao Nishizeki

Graduate School of Information Sciences, Tohoku University
Aoba-yama 05, Sendai, 980-8579, Japan
take@nishizeki.ecei.tohoku.ac.jp, {zhou,nishi}@ecei.tohoku.ac.jp

Abstract. Assume that each vertex of a graph G is assigned a nonnegative integer weight and that l and u are nonnegative integers. One wish 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 similarly. 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 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 of n vertices. The algorithms can be easily extended for partial k-trees, that is, graphs with bounded tree-width.

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 with small $u - l$. We call such an almost uniform partition an (l, u)-*partition* of G. Figures 1(a) and (b) illustrate two $(10, 20)$-partitions of the same graph, where each vertex is drawn by a circle, the weight of each vertex is written inside the circle, and the deleted edges are drawn by dotted lines. In this paper we deal with 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 similarly. The *p-partition problem* is to find an (l, u)-partition of G with a fixed number p of components. The

J. Hromkovič, M. Nagl, and B. Westfechtel (Eds.): WG 2004, LNCS 3353, pp. 365–376, 2004.

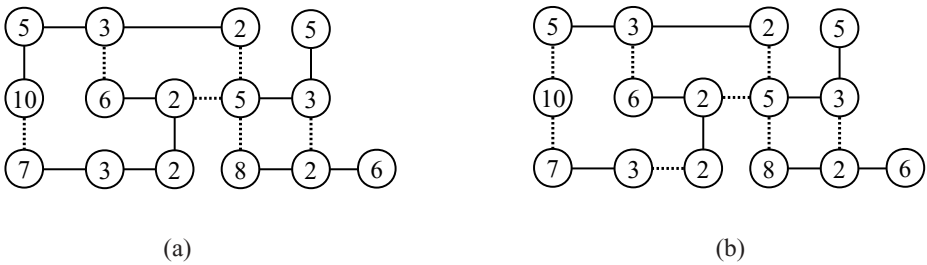

Fig. 1. (a) Solution for the minimum partition problem, and (b) solution for the maximum partition problem, where $l = 10$ and $u = 20$.

$(10, 20)$-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 the image processing [5, 7], the paging system of operation system [10], and the political districting [3, 11]. Consider a map of a country, which is divided into several regions. Let G be a dual graph of the map. Each vertex v of G represents a region, and the weight $\omega(v)$ represents 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 with small $u - l$. 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 [8]. The other is to partition a tree into the minimum number of subtrees so that the total weight of each subtree is at most u [6]. 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 [6, 8], except for paths; all the three partition problems can be solved for paths in linear time [7].

An NP-complete problem, called the set partition problem [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, 9]. One can also observe from the reduction above 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.

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 easily 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.)

2 Terminology and Definitions

In this section we give some definitions.

A (*two-terminal*) *series-parallel graph* is defined recursively as follows [9]:

(1) A graph G of a single edge is a series-parallel graph. The ends 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).)

  (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 the partition problems, we may assume without loss of generality that G is a simple graph and hence G has no multiple edges.

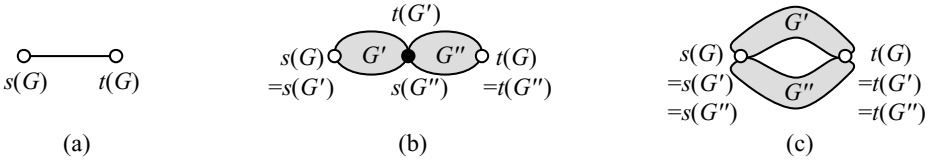

Fig. 2. (a) A series-parallel graph of a single edge, (b) series connection, and (c) parallel connection.

A series-parallel graph G can be represented by a "binary decomposition tree" [9]. Figure 3 illustrates a series-parallel graph G and its binary decomposition tree T. 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. Since a binary decomposition tree of a given series-parallel graph G can be found in linear time [9], 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 decomposition tree T.

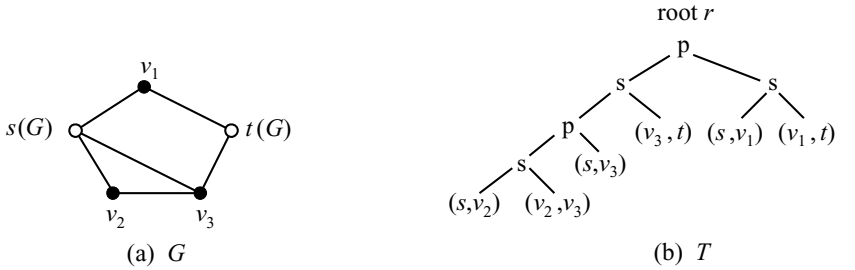

Fig. 3. (a) A series-parallel graph G, and (b) its binary decomposition tree T.

3 Minimum and Maximum Partition Problems

In this section we have the following theorem.

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, because the maximum partition problem can be similarly 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 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 are illustrated in Fig. 4 and will be formally defined later. Roughly speaking, two functions $f(G_v, x)$ and $h(G_v, x, y)$, $0 \leq x, y \leq u$, 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 leaves v to the root r of T by means of dynamic programming.

(a) (b)

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

We now formally define the connected partition and the separated partition of a series-parallel graph $G = (V, E)$. Let $\mathcal{P} = \{P_1, P_2, \cdots, P_r\}$ be a partition of vertex set V of G into r nonempty subsets $P_1, P_2, \cdots, P_r$ for some integer $r \geq 1$. Thus $|\mathcal{P}| = r$. The partition $\mathcal{P}$ of V is called a *partition of* G if P_i induces a connected subgraph of G for each index $i, 1 \leq i \leq r$. 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 $\omega(P_{st}) \leq u$; and
(b) $l \leq \omega(P) \leq u$ for each set $P \in \mathcal{P} - \{P_{st}\}$.

Note that the equation $l \leq \omega(P_{st})$ 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 $s \in P_s$, $t \in P_t$, $\omega(P_s) \leq u$, and $\omega(P_t) \leq u$; and
(b) $l \leq \omega(P) \leq u$ for each set $P \in \mathcal{P} - \{P_s, P_t\}$.

Note that the equations $l \leq \omega(P_s)$ and $l \leq \omega(P_t)$ 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 $f(G, x)$ for a series-parallel graph G and an integer $x, 0 \leq x \leq u$, as follows:

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

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 $h(G, x, y)$ for a series-parallel graph G and a pair $(x, y), 0 \leq x, y \leq u$, as follows:

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

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 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:

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

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 $x = 0$

$$f(G_v, 0) = 0, \quad (4)$$

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

$$h(G_v, 0, 0) = 0. \quad (5)$$

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

$$f(G_v, x) = +\infty, \quad (6)$$

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

$$h(G_v, x, y) = +\infty. \quad (7)$$

By Eqs. (4)–(7) one can compute $f(G_v, x)$ in time $O(u)$ for each leaf v of T and all integers $x \le u$, and compute $h(G_v, x, y)$ in time $O(u^2)$ for each leaf v and all pairs (x, y) with $x, y \le u$. 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 $x + \omega(s) \le u$ and $y + \omega(t) \le u$. Then every separated partition $\mathcal{P}$ of G_v can be obtained by combining a separated partition $\mathcal{P}'$ of G' with a separated partition $\mathcal{P}''$ of G'' as illustrated in Fig. 5(a). We thus have

$$h(G_v, x, y) = \min\{h(G', x', y') + h(G'', x - x', y - y') \mid 0 \le x', y' \le u\}. \quad (8)$$

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 $x + \omega_{st}(G_v) \le u$. Then every

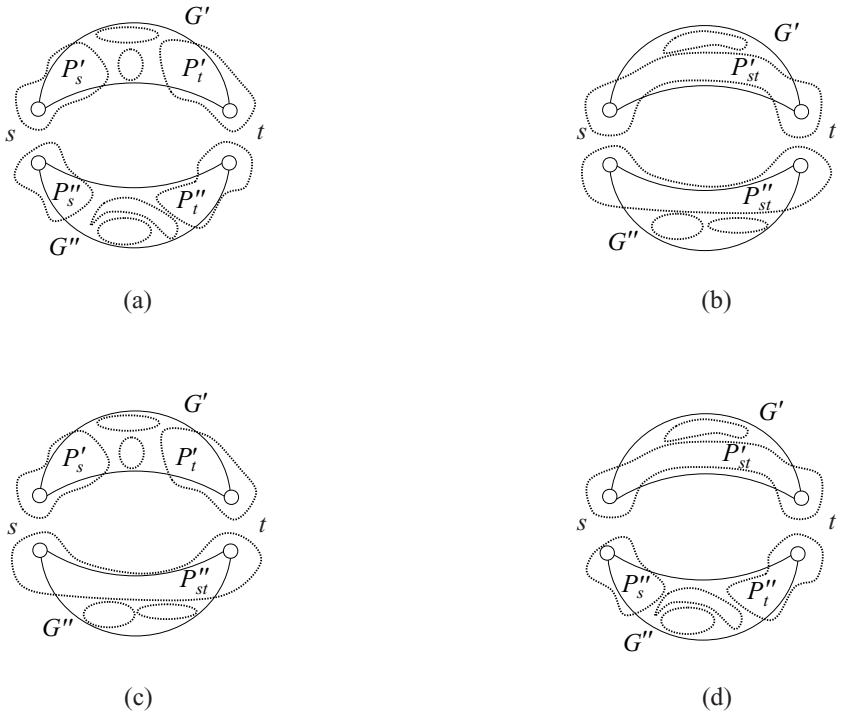

Fig. 5. The combinations of a partition $\mathcal{P}'$ of G' and a partition $\mathcal{P}''$ of G'' for a partition $\mathcal{P}$ of $G_v = G' \parallel G''$.

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), (c) and (d). There are the following three Cases (a)–(c), and we define three functions $f^a(G_v, x)$, $f^b(G_v, x)$ and $f^c(G_v, x)$ for the three cases, respectively.

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

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

Case (*b*): $\mathcal{P}'$ *is a separated partition, and* $\mathcal{P}''$ *is a connected partition.* (See Fig. 5(c).)
Let

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

Case (*c*): $\mathcal{P}'$ *is a connected partition, and* $\mathcal{P}''$ *is a separated partition.* (See Fig. 5(d).)
Let

$$f^c(G_v, x) = \min\{f(G', x - x'' - y'') + h(G'', x'', y'') \mid 0 \le x'', y'' \le u\}. \tag{11}$$

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

$$f(G_v, x) = \min\{f^a(G_v, x), f^b(G_v, x), f^c(G_v, x)\}. \tag{12}$$

By Eq. (8) one can compute the function $h(G_v, x, y)$ for all pairs $(x, y), 0 \leq x, y \leq u$, in time $O(u^4)$, and by Eqs. (9)–(12) one can compute the function $f(G_v, x)$ for all integers $x, 0 \leq x \leq u$, in time $O(u^3)$. Thus one can compute the functions $f(G_v, x)$ and $h(G_v, x, y)$ for each p-node v of T in time $O(u^4)$.

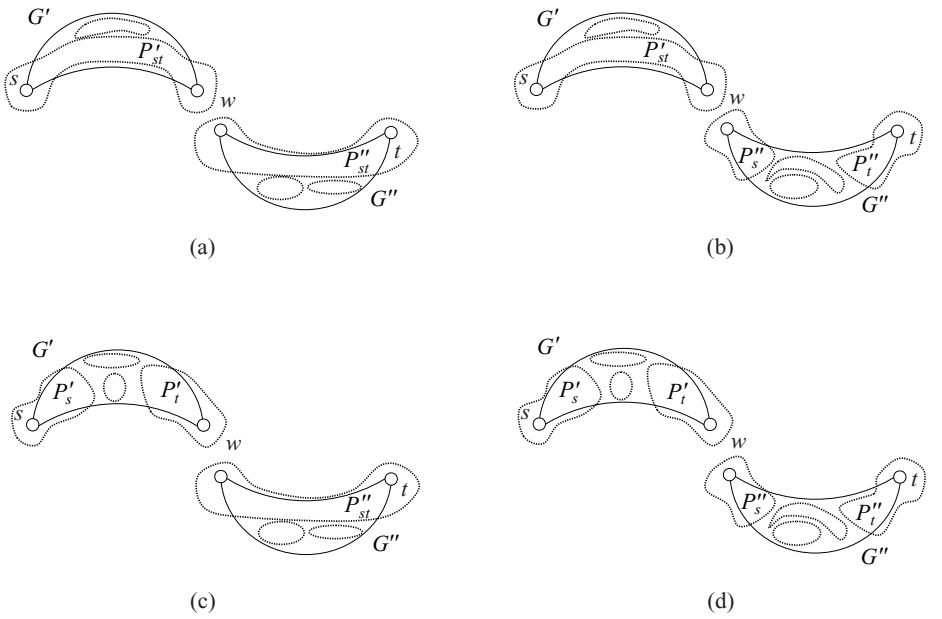

Fig. 6. The combinations of a partition $\mathcal{P}'$ of G' and a partition $\mathcal{P}''$ of G'' for a partition $\mathcal{P}$ of $G_v = G' \bullet G''$.

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 $x + \omega_{st}(G_v) \leq u$. 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

$$f(G_v, x) = \min\{f(G', x') + f(G'', x'') \mid 0 \leq x', x'' \leq u, \\ x' + x'' + \omega(w) = x\}. \tag{13}$$

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 $x + \omega(s) \leq u$ and $y + \omega(t) \leq u$. 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), (c) and (d). There are the following three Cases (a)–(c), and we define three functions $h^a(G_v, x, y), h^b(G_v, x, y)$ and $h^c(G_v, x, y)$ for the three cases, respectively.

Case (a): $\mathcal{P}'$ is a connected partition, and $\mathcal{P}''$ is a separated partition. (See Fig. 6(b).)

Let

$$h^a(G_v, x, y) = \min\{f(G', x') + h(G'', x'', y) \mid 0 \le x', x'' \le u, \\ x' + x'' + \omega(w) = x\}. \quad (14)$$

Case (b): $\mathcal{P}'$ is a separated partition, and $\mathcal{P}''$ is a connected partition. (See Fig. 6(c).)

Let

$$h^b(G_v, x, y) = \min\{h(G', x, y') + f(G'', x'') \mid 0 \le y', x'' \le u, \\ y' + x'' + \omega(w) = y\}. \quad (15)$$

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

Let

$$h^c(G_v, x, y) = \min\{h(G', x, y') + h(G'', x'', y) + 1 \mid 0 \le y', x'' \le u, \\ l \le y' + x'' + \omega(w) \le u\}. \quad (16)$$

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

$$h(G_v, x, y) = \min\{h^a(G_v, x, y), h^b(G_v, x, y), h^c(G_v, x, y)\}. \quad (17)$$

By Eq. (13) one can compute the function $f(G_v, x)$ for all integers $x, 0 \le x \le u$, in time $O(u^2)$, and by Eqs. (14)–(17) one can compute the function $h(G_v, x, y)$ for all pairs $(x, y), 0 \le x, y \le u$, in time $O(u^4)$. Thus one can compute the functions $f(G_v, x)$ and $h(G_v, x, y)$ for each s-node v of T in time $O(u^4)$.

In this way one can compute the functions $f(G_v, x)$ 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 the functions $f(G, x)$ 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 the functions $f(G, x)$ 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 a 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 similar 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 $q, 0 \leq q \leq p-1$, we define a set $F(G,q)$ of nonnegative integers x as follows:

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

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

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

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

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 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 the following condition* (a) *or* (b) *holds*:

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

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 $q(\leq p-1)$, and compute in time $O(p^2u^4)$ the sets $F(G_v,q)$ and $H(G_v,q)$ for each internal node v of T and all integers $q(\leq p-1)$ 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 $O(p^2u^4n)$. 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 $O(p^2u^4n)$.

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-trees, where $k = O(1)$.*

The algorithm for partial k-trees is similar 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. 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 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 $\pi \leq (2^{k+1})^{k+1} = O(1)$ since we assume $k = O(1)$ in the paper. For a partial k-tree G, we consider π kinds of partitions of G_v. Let $\mathcal{V}_i, 1 \leq i \leq \pi$, be the ith partition of set $V(v)$, let $\rho(i)$ be the number of subsets in the partition $\mathcal{V}_i$, and let $\mathcal{V}_i = \{V_1, V_2, \cdots, V_{\rho(i)}\}$. Clearly $1 \leq \rho(i) \leq k+1$. In every partition of G_v of the ith kind, its jth connected component, $1 \leq j \leq \rho(i)$, contains all the vertices in the jth subset $V_j (\subseteq V(v))$ in $\mathcal{V}_i$. We consider a set of functions $h_i(G_v, x_1, x_2, \cdots, x_{\rho(i)}), 1 \leq i \leq \pi$, defined similarly to Eqs. (1) and (2). Variable $x_j, 1 \leq j \leq \rho(i)$, represents the sum of weights of all vertices in the jth component except for the vertices in V_j. Thus $0 \leq x_j \leq u$. 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 $\pi^2 (\leq 2^{2(k+1)^2})$.

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 easily 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 $O({}_nC_{p-1} + n)$.

Acknowledgments

We thank Takeshi Tokuyama for suggesting us the partition problems.

References

1. S. Arnborg and J. Lagergren. Easy problem for tree-decomposable graphs. *J. Algorithms*, Vol. 12, No. 2, pp. 308–340, 1991.
2. H. L. Bodlaender. Polynomial algorithms for graph isomorphism and chromatic index on partial k-trees. *J. Algorithms*, Vol. 11, No. 4, pp. 631–643, 1990.
3. B. Bozkaya, E. Erkut and G. Laporte. A tabu search heuristic and adaptive memory procedure for political districting. *European J. Operational Research*, Vol. 144, pp. 12–26, 2003.
4. M. R. Garey and D. S. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. *Freeman, San Francisco, CA*, 1979.
5. R. C. Gonzales and P. Wintz. Digital Image Processing. *Addison-Wesley, Reading, MA*, 1977.
6. S. Kundu and J. Misra. A linear tree-partitioning algorithm. *SIAM J. Comput.*, Vol. 6, pp. 131–134, 1977.
7. M. Lucertini, Y. Perl and B. Simeone. Most uniform path partitioning and its use in image processing. *Discrete Applied Mathematics*, Vol. 42, pp. 227–256, 1993.
8. Y. Perl and S. R. Schach. Max-min tree-partitioning. *J. ACM*, Vol. 28, pp. 5–15, 1981.
9. K. Takamizawa, T. Nishizeki and N. Saito. Linear-time computability of combinatorial problems on series-parallel graphs. *J. ACM*, Vol. 29, No. 3, pp. 623–641, 1982.
10. D. C. Tsichritzis and P. A. Bernstein. Operating Systems. *Academic Press*, New York, 1981.
11. J. C. Williams Jr. Political redistricting: a review. *Papers in Regional Science*, Vol. 74, pp. 12–40, 1995.