Discrete Applied Mathematics 157 (2009) 2620–2633

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

Partitioning graphs of supply and demand

Takehiro Ito [*], Xiao Zhou, Takao Nishizeki

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

ARTICLE INFO

Article history:
Received 1 February 2006
Received in revised form 6 October 2006
Accepted 6 August 2008
Available online 25 November 2008

Keywords:
Demand
Graph partition problem
Series–parallel graph
Supply
Partial k-tree

ABSTRACT

Assume that each vertex of a graph G is either a supply vertex or a demand vertex and is assigned a positive integer, called a supply or a demand. Each demand vertex can receive "power" from at most one supply vertex through edges in G. One thus wishes to partition G into connected components by deleting edges from G so that each component C has exactly one supply vertex whose supply is no less than the sum of demands of all demand vertices in C. If G does not have such a partition, one wishes to partition G into connected components so that each component C either has no supply vertex or has exactly one supply vertex whose supply is no less than the sum of demands in C, and wishes to maximize the sum of demands in all components with supply vertices. We deal with such a maximization problem, which is NP-hard even for trees and strongly NP-hard for general graphs. In this paper, we show that the problem can be solved in pseudo-polynomial time for series–parallel graphs and partial k-trees – that is, graphs with bounded tree-width.

1. Introduction

Let $G = (V, E)$ be a graph with vertex set V and edge set E. The set V is partitioned into two sets V_s and V_d. Let $|V| = n$, $|V_s| = n_s$ and $|V_d| = n_d$, then $n = n_s + n_d$. Each vertex $u \in V_s$ is called a *supply vertex* and is assigned a positive integer $\text{sup}(u)$, called a *supply of* u, while each vertex $v \in V_d$ is called a *demand vertex* and is assigned a positive integer $\text{dem}(v)$, called a *demand of* v. Each demand vertex can receive "power" from at most one supply vertex through edges in G. One thus wishes to partition G into connected components by deleting edges from G so that each component C has exactly one supply vertex whose supply is no less than the sum of demands of all demand vertices in C. However, such a partition does not always exist. So we wish to partition G into connected components so that each component C either has no supply vertex or has exactly one supply vertex whose supply is no less than the sum of demands of all demand vertices in C, and wish to maximize the "fulfillment", that is, the sum of demands of the demand vertices in all components with supply vertices. We call the problem the *maximum partition problem*. Fig. 1 illustrates a solution of the maximum partition problem for a graph G, whose fulfillment is $(2 + 7) + (8 + 7) + (3 + 6) + 4 = 37$, where each supply vertex is drawn as a rectangle and each demand vertex as a circle, the supply or demand is written inside, the deleted edges are drawn by thick dotted lines, and each connected component is indicated by a thin dotted line.

The maximum partition problem has some applications to the power supply problem for power delivery networks [4,9,10,13]. Let G be a graph of a power delivery network. Each supply vertex v represents a "feeder", which can supply at most an amount $\text{sup}(v)$ of electrical power. Each demand vertex v represents a "load", which requires an amount $\text{dem}(v)$ of electrical power supplied from exactly one of the feeders through a network. Each edge of G represents a cable segment, which can be "turned off" by a switch. Then the maximum partition problem represents the "power supply switching problem" to maximize the sum of all loads that can be supplied powers in a network "reconfigured" by turning off some cable segments.

[*] Corresponding author. Fax: +81 22 263 9304.
E-mail addresses: takehiro@ecei.tohoku.ac.jp (T. Ito), zhou@ecei.tohoku.ac.jp (X. Zhou), nishi@ecei.tohoku.ac.jp (T. Nishizeki).

0166-218X/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2008.08.012

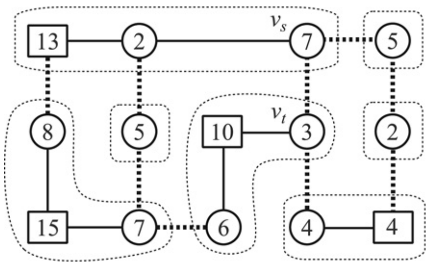


Fig. 1. Partition P' of a series–parallel graph G with maximum fulfillment.

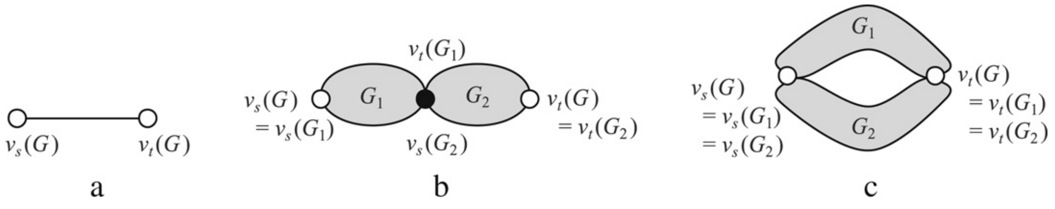


Fig. 2. (a) A series–parallel graph with a single edge, (b) series connection, and (c) parallel connection.

The maximum partition problem is a generalization of the maximum subset sum problem and the knapsack problem [7–9]. The maximum partition problem is NP-hard even for trees, because the maximum subset sum problem can be easily reduced in linear time to the maximum partition problem for a tree [9]. Thus, it is very unlikely that the maximum partition problem can be solved even for trees in polynomial time. However, the problem can be solved in pseudo-polynomial-time for trees, and there is a fully polynomial-time approximation scheme (FPTAS) for the problem on trees [9]. A strongly NP-complete problem called 3-PARTITION [7] can be easily reduced in polynomial time to the maximum partition problem for a complete bipartite graph, and hence the maximum partition problem is strongly NP-hard for general graphs. Therefore, there is no pseudo-polynomial-time algorithm for the problem on general graphs unless P = NP. One may thus expect to obtain a pseudo-polynomial-time algorithm for a class of graphs, larger than the class of trees.

In this paper we first give a pseudo-polynomial-time algorithm to solve the maximum partition problem for series–parallel graphs. (A series–parallel graph will be defined in Section 2.) It takes time $O\left((m_s)^4 n\right)$, where m_s is the maximum supply, that is, $m_s = \max\{\sup(u) \mid u \in V_s\}$. Thus, the algorithm takes polynomial time if m_s is bounded by a polynomial in n, and takes linear time if m_s is a fixed constant. We then show that our algorithm can be extended for partial k-trees, that is, graphs with bounded tree-width [1,3]. (A partial k-tree will be defined in Section 4.) The algorithm takes time $O\left((2m_s + 1)^{2(k+1)} n\right)$. Telle and Proskurowski present a theory of algorithm design for a large class of vertex partitioning problems restricted to partial k-trees [12], but they deal with only unweighted partial k-trees. Many combinatorial problems can be efficiently solved for partial k-trees if they can be expressed in extended monadic second-order logic (EMSOL) [5,6]. However, our maximum partition problem is very unlikely to be expressible in EMSOL; if the problem is expressible in EMSOL, then it would be solvable for partial k-trees in polynomial time.

2. Terminology and definitions

In this section we give some definitions.

A (*two-terminal*) *series–parallel graph* is defined recursively as follows [11]:

(1) A graph G with a single edge is a series–parallel graph. The ends of the edge are called the *terminals* of G and denoted by $v_s(G)$ and $v_t(G)$. (See Fig. 2(a).)
(2) Let G_1 be a series–parallel graph with terminals $v_s(G_1)$ and $v_t(G_1)$, and let G_2 be a series–parallel graph with terminals $v_s(G_2)$ and $v_t(G_2)$.
  (a) A graph G obtained from G_1 and G_2 by identifying $v_t(G_1)$ with $v_s(G_2)$ is a series–parallel graph, whose terminals are $v_s(G) = v_s(G_1)$ and $v_t(G) = v_t(G_2)$. Such a connection is called a *series connection*, and G is denoted by $G = G_1 \bullet G_2$. (See Fig. 2(b).)
  (b) A graph G obtained from G_1 and G_2 by identifying $v_s(G_1)$ with $v_s(G_2)$ and identifying $v_t(G_1)$ with $v_t(G_2)$ is a series–parallel graph, whose terminals are $v_s(G) = v_s(G_1) = v_s(G_2)$ and $v_t(G) = v_t(G_1) = v_t(G_2)$. Such a connection is called a *parallel connection*, and G is denoted by $G = G_1 \parallel G_2$. (See Fig. 2(c).)

The terminals $v_s(G)$ and $v_t(G)$ of G are often denoted simply by v_s and v_t, respectively. Since we deal with the maximum partition problem, 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" [11]. 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

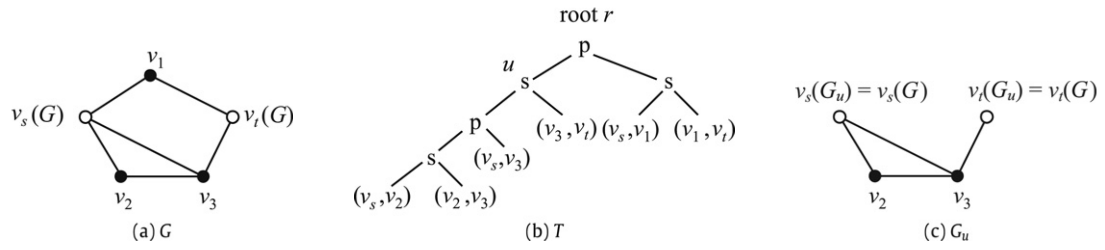


Fig. 3. (a) A series–parallel graph G, (b) a binary decomposition tree T of G, and (c) a subgraph G_u.

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 u of T corresponds to a subgraph G_u of G induced by all edges represented by the leaves that are descendants of u in T. Fig. 3(c) depicts G_u for the left child u of the root r of T in Fig. 3(b). G_u is a series–parallel graph for each node u 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 [11], we may assume that a series–parallel graph G and its binary decomposition tree T are given. We solve the maximum partition problem by a dynamic programming approach based on a binary decomposition tree T.

3. Algorithm for the maximum partition problem

In this section, we give a pseudo-polynomial-time algorithm to solve the maximum partition problem for series–parallel graphs. The main result of this section is the following theorem.

Theorem 1. *The maximum partition problem can be solved for every series–parallel graph G in time* $O\left((m_s)^4 n\right)$, *where n is the number of vertices in G and* m_s *is the maximum supply.*

In the remainder of this section we give an algorithm to solve the maximum partition problem in time $O\left((m_s)^4 n\right)$ as a proof of Theorem 1. In Section 3.1 we define some terms and present ideas of our algorithm. In Sections 3.2 and 3.3 we present our algorithm. In Section 3.4 we show that our algorithm takes time $O\left((m_s)^4 n\right)$.

3.1. Terms and ideas

We partition a series–parallel graph G into connected components by deleting edges from G so that

(a) each component contains at most one supply vertex; and
(b) if a component C contains a supply vertex, then the supply is no less than the sum of demands of all demand vertices in C.

Such a partition P is called a *partition* of G. The *fulfillment* $f(P)$ *of a partition* P is the sum of demands of all demand vertices in components with supply vertices. Thus, $f(P)$ corresponds to the maximum sum of all loads that are supplied electrical power from feeders through a network reconfigured by cutting off some edges. The *maximum partition problem* is to find a partition of G with the maximum fulfillment. The *maximum fulfillment* $f(G)$ *of a graph* G is the maximum fulfillment $f(P)$ among all partitions P of G. For every partition P of G, there is a partition P' of G such that

(a) $f(P) = f(P')$; and
(b) if a component C does not contain a supply vertex, then $|C| = 1$.

Our algorithm indeed finds such a partition P' with $f(P') = f(G)$. For the series–parallel graph G in Fig. 1, our algorithm finds the partition P' indicated by thin dotted lines.

Every partition of a series–parallel graph G naturally induces a partition of its subgraph G_u for a node u of a binary decomposition tree T of G. The partition P of G in Fig. 4(a) induces a partition P_u of G_u in Fig. 4(b), for which the terminals $v_s(G_u)$ and $v_t(G_u)$ of G_u are contained in the same component of P_u. On the other hand, the partition P of G in Fig. 5(a) induces

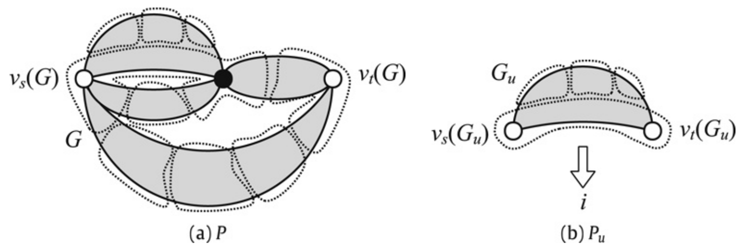


Fig. 4. (a) A partition P of a series–parallel graph G, and (b) a connected partition P_u of a subgraph G_u.

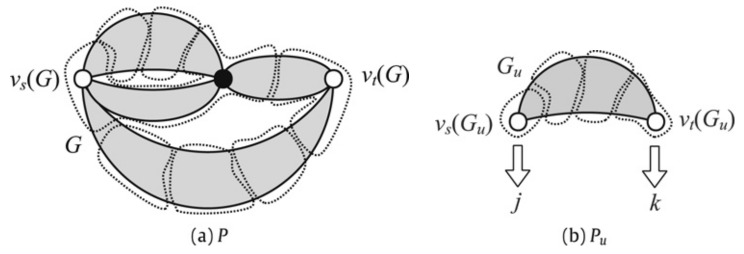


Fig. 5. (a) A partition P of a series–parallel graph G, and (b) a separated partition P_u of a subgraph G_u.

a partition P_u of G_u in Fig. 5(b), for which $v_s(G_u)$ and $v_t(G_u)$ are contained in different components of P_u. We need to consider these two types of partitions, called a "connected partition" and a "separated partition", which will be formally defined later. We will later show that

> if a component of P_u with a terminal contains a supply vertex, then the component may have the "marginal" power, the amount of which is no greater than m_s; otherwise, the component may have the "deficient" power, the amount of which should be no greater than m_s.

We will thus introduce two functions $g : (\mathcal{G}, \mathbb{Z}_{m_s}) \to \mathbb{Z}^+$ and $h : (\mathcal{G}, \mathbb{Z}_{m_s}, \mathbb{Z}_{m_s}) \to \mathbb{Z}^+$, where $\mathcal{G}$ denotes the set of all series–parallel graphs, $\mathbb{Z}^+$ denotes the set of all nonnegative integers, and $\mathbb{Z}_{m_s}$ denotes the set of all integers whose absolute values are no greater than m_s. A positive integer in $\mathbb{Z}_{m_s}$ means an amount of marginal power, while a nonpositive integer in $\mathbb{Z}_{m_s}$ means an amount of deficient power. For $G_u \in \mathcal{G}$ and $i \in \mathbb{Z}_{m_s}$, the value $g(G_u, i) \in \mathbb{Z}^+$ represents the maximum fulfillment of a connected partition of G_u having the marginal or deficient power i in the component with terminals $v_s(G_u)$ and $v_t(G_u)$. For $G_u \in \mathcal{G}$ and $j, k \in \mathbb{Z}_{m_s}$, the value $h(G_u, j, k) \in \mathbb{Z}^+$ represents the maximum fulfillment of a separated partition of G_u having the marginal or deficient power j in the component with $v_s(G_u)$ and k in the component with $v_t(G_u)$. Our idea is to compute $g(G_u, i)$ and $h(G_u, j, k)$ from the leaves of T to the root r of T by means of dynamic programming.

We now formally define the notion of connected partitions and separated partitions of a series–parallel graph G. Let P be a partition of a subgraph G_u of G for a node u of a binary decomposition tree T of G, and let $v_s = v_s(G_u)$ and $v_t = v_t(G_u)$. Let $C(P, v_s)$ be the component of P containing v_s. We denote also by $C(P, v_s)$ the set of all vertices in the component $C(P, v_s)$. Similarly we define $C(P, v_t)$. If $C(P, v_s) = C(P, v_t)$, that is, the two terminals v_s and v_t are contained in the same component of P, then we call P a *connected partition* of G_u. (See Fig. 4(b).) If $C(P, v_s) \neq C(P, v_t)$, that is, the two terminals v_s and v_t are contained in different components of P, we call P a *separated partition* of G_u. (See Fig. 5(b).)

We then classify both connected partitions and separated partitions further into several classes, called i-connected partitions and (j, k)-separated partitions for $i, j, k \in \mathbb{Z}_{m_s}$. The "power flow" around a terminal depends on whether the terminal is a supply vertex or a demand vertex. Since we want to deal with the two cases uniformly, we introduce a virtual graph G_u^* for a subgraph G_u of G; G_u^* is obtained from G_u by regarding each of the two terminals v_s and v_t as a demand vertex whose demand is zero. We denote by $\mathrm{dem}^*(x)$ the demand of a demand vertex x in G_u^*, and hence

$$\mathrm{dem}^*(x) = \begin{cases} 0 & \text{if } x \text{ is } v_s \text{ or } v_t; \\ \mathrm{dem}(x) & \text{otherwise.} \end{cases}$$

Clearly every partition of G_u is a partition of G_u^*. However, a partition P of G_u^* is not necessarily a partition of G_u; for example, if v_s is a supply vertex of G, $C(P, v_s)$ does not contain any supply vertex of G_u^*, and $\sum_{x \in C(P,v_s)} \mathrm{dem}^*(x) > \sup(v_s)$, then P is not a partition of G_u. For a partition P of G_u^*, we denote by $f^*(P)$ the fulfillment of P for G_u^*. We denote by G_u^{in} the graph obtained from G_u by deleting the two terminals v_s and v_t as illustrated in Fig. 6(b), while we denote by G_u^{out} the graph obtained from G by deleting all the vertices of G_u except v_s and v_t as illustrated in Fig. 6(c).

If P is a connected partition of G_u^*, then $C(P, v_s) = C(P, v_t)$ and we denote it simply by $C(P)$. For each integer $i \in \mathbb{Z}_{m_s}$, we call P an *i-connected partition* of G_u^* if P satisfies the following two conditions (a) and (b):

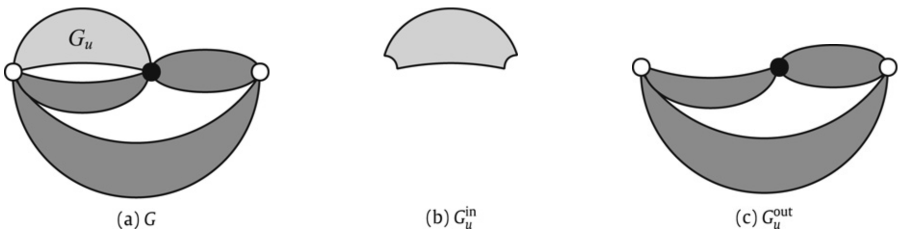


Fig. 6. (a) A series–parallel graph G, (b) a subgraph G_u^{in} of G_u, and (c) a subgraph G_u^{out} of G.

(a) if $i > 0$, then $C(P)$ contains a supply vertex w and

$$i + \sum_{x \in C(P)-\{w\}} \text{dem}^*(x) \leq \text{sup}(w);$$

and

(b) if $i \leq 0$, then $C(P)$ contains no supply vertex and

$$\sum_{x \in C(P)} \text{dem}^*(x) \leq |i| = -i. \tag{1}$$

An i-connected partition P of G_u^* with $i > 0$ corresponds to a partition of the whole graph G in which all demand vertices in $C(P)$ are supplied power from a supply vertex w in $C(P)$; an amount i of the remaining power of w can be delivered to G_u^{out} through $v_s(G_u)$ or $v_t(G_u)$; and hence the "margin" of $C(P)$ is i. On the other hand, an i-connected partition P of G_u^* with $i \leq 0$ corresponds to a partition of G in which all (demand) vertices in $C(P)$ are supplied power from a supply vertex in G_u^{out}; an amount $|i|$ of power must be delivered to G_u^{in} through $v_s(G_u)$ or $v_t(G_u)$, and hence the "deficiency" of $C(P)$ is $|i|$. It should be noted that if P is a 0-connected partition of G_u^* then $C(P) = \{v_s, v_t\}$ and G_u has an edge (v_s, v_t); note that all supplies and demands are assumed to be positive integers. For an i-connected partition P of G_u^*, let

$$f(P, i) = \begin{cases} f^*(P) & \text{if } 0 < i \leq m_s, \\ f^*(P) + \sum_{x \in C(P)} \text{dem}^*(x) & \text{if } -m_s \leq i \leq 0. \end{cases} \tag{2}$$

Thus, $f(P, i)$ with $0 < i \leq m_s$ represents the fulfillment of P for G_u^* when an amount i of power can be delivered to G_u^{out} through $v_s(G_u)$ or $v_t(G_u)$, while $f(P, i)$ with $-m_s \leq i \leq 0$ represents the fulfillment of P for G_u^* when an amount $|i|$ of power is delivered to G_u^{in} from a supply vertex w in G_u^{out}; if either $v_s(G_u)$ or $v_t(G_u)$ is a supply vertex, then it must be w. According to the definition of an i-connected partition, a connected partition P of G_u^* is not a 0-connected partition of G_u^* if $C(P)$ contains a supply vertex w ($\neq v_s, v_t$) and

$$\sum_{x \in C(P)-\{w\}} \text{dem}^*(x) = \text{sup}(w);$$

it should be noted that such a partition P of G_u^* is not a partition of G_u and hence we need not to take P into account; if v_s or v_t is a supply vertex of G, then $C(P)$ would contain two or three supply vertices of G_u including w; if both v_s and v_t are demand vertices of G, then their demands are positive and hence

$$\sum_{x \in C(P)-\{w\}} \text{dem}(x) > \text{sup}(w).$$

For each pair of integers j and k in $\mathbb{Z}_{m_s}$, we call a separated partition P of G_u^* a (j, k)-*separated partition* if P satisfies the following four conditions (a), (b), (c) and (d):

(a) if $j > 0$, then $C(P, v_s)$ contains a supply vertex w and

$$j + \sum_{x \in C(P, v_s)-\{w\}} \text{dem}^*(x) \leq \text{sup}(w);$$

(b) if $j \leq 0$, then $C(P, v_s)$ contains no supply vertex and

$$\sum_{x \in C(P, v_s)} \text{dem}^*(x) \leq -j; \tag{3}$$

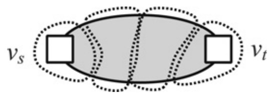


Fig. 7. Partition of G for Case (a).

(c) if $k > 0$, then $C(P, v_t)$ contains a supply vertex w and

$$k + \sum_{x \in C(P,v_t)-\{w\}} \mathrm{dem}^*(x) \le \mathrm{sup}(w);$$

and

(d) if $k \le 0$, then $C(P, v_t)$ contains no supply vertex and

$$\sum_{x \in C(P,v_t)} \mathrm{dem}^*(x) \le -k. \tag{4}$$

A (j, k)-separated partition P of G_u^* with $j > 0$ corresponds to a partition of the whole graph G in which all demand vertices in $C(P, v_s)$ are supplied power from a supply vertex w in $C(P, v_s)$; an amount j of the remaining power of w can be delivered to G_u^{out} through $v_s(G_u)$, and hence the margin of $C(P, v_s)$ is j. A (j, k)-separated partition P of G_u^* with $j \le 0$ corresponds to a partition of G in which all (demand) vertices in $C(P, v_s)$ are supplied power from a supply vertex in G_u^{out}; an amount $|j|$ of power must be delivered to G_u^{in} through $v_s(G_u)$, and hence the deficiency of $C(P, v_s)$ is $|j|$. Clearly $C(P, v_s) = \{v_s\}$ if P is a $(0, k)$-separated partition of G_u^*. A (j, k)-separated partition P of G_u^* with $k > 0$ or $k \le 0$ corresponds to a partition of G similarly as above. For a (j, k)-separated partition P of G_u^*, let

$$f(P, j, k) = \begin{cases} f^*(P) & \text{if } 0 < j, k \le m_s, \\ f^*(P) + \sum_{x \in C(P,v_t)} \mathrm{dem}^*(x) & \text{if } 0 < j \le m_s \text{ and } -m_s \le k \le 0, \\ f^*(P) + \sum_{x \in C(P,v_s)} \mathrm{dem}^*(x) & \text{if } -m_s \le j \le 0 \text{ and } 0 < k \le m_s, \\ f^*(P) + \sum_{x \in C(P,v_s) \cup C(P,v_t)} \mathrm{dem}^*(x) & \text{if } -m_s \le j, k \le 0. \end{cases} \tag{5}$$

Thus, $f(P, j, k)$ with nonpositive j or k represents the fulfillment of P for G_u^* when an amount $|j|$ or $|k|$ of power is delivered to G_u^{in} from a supply vertex in G_u^{out} through v_s or v_t, respectively.

We now formally define a function $g : (\mathcal{G}, \mathbb{Z}_{m_s}) \to \mathbb{Z}^+$ for a series–parallel graph $G_u^* \in \mathcal{G}$ and an integer $i \in \mathbb{Z}_{m_s}$, as follows:

$$g(G_u^*, i) = \max\{f(P, i) \mid G_u^* \text{ has an } i\text{-connected partition } P\}. \tag{6}$$

If G_u^* has no i-connected partition, then let $g(G_u^*, i) = -\infty$. We then formally define a function $h : (\mathcal{G}, \mathbb{Z}_{m_s}, \mathbb{Z}_{m_s}) \to \mathbb{Z}^+$ for a series–parallel graph $G_u^* \in \mathcal{G}$ and a pair of integers j and k in $\mathbb{Z}_{m_s}$, as follows:

$$h(G_u^*, j, k) = \max\{f(P, j, k) \mid G_u^* \text{ has a } (j, k)\text{-separated partition } P\}. \tag{7}$$

If G_u^* has no (j, k)-separated partition, then let $h(G_u^*, j, k) = -\infty$.

Our algorithm computes $g(G_u^*, i)$ and $h(G_u^*, j, k)$ for each node u 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.

3.2. How to compute $f(G)$ from $g(G^*, i)$ and $h(G^*, j, k)$

Suppose that $g(G_r^*, i)$ and $h(G_r^*, j, k)$ have been computed for the root r of T. Since $G = G_r$, one can easily compute the maximum fulfillment $f(G)$ from $g(G^*, i)$ and $h(G^*, j, k)$ in time $O(1)$, as in the following three cases (a), (b) and (c), where $v_s = v_s(G)$, $v_t = v_t(G)$, and let P be a partition of G^* corresponding to a partition of G having the maximum fulfillment $f(G)$. One may assume that $|C| = 1$ for every component C of P containing no supply vertex.

Case (a): *both v_s and v_t are supply vertices in G.*

In this case, $C(P, v_s) \neq C(P, v_t)$ and hence P is a separated partition as illustrated in Fig. 7. Since the partition P of G^* corresponds to a partition of G, we have

$$\sum_{x \in C(P,v_s)} \mathrm{dem}^*(x) \le \mathrm{sup}(v_s)$$

and

$$\sum_{x \in C(P,v_t)} \mathrm{dem}^*(x) \le \mathrm{sup}(v_t).$$

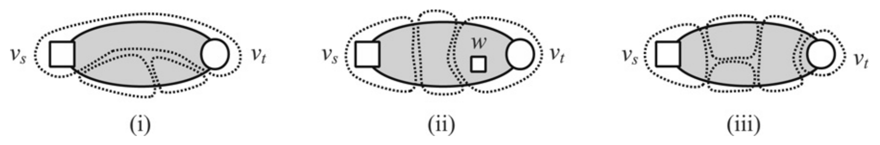


Fig. 8. Partitions of G for the three cases in Case (b).

Thus, Eqs. (3) and (4) hold for P with $j = -\sup(v_s)$ and $k = -\sup(v_t)$, and hence P is a (j, k)-separated partition of G^* for such j and k. Since P corresponds to a partition of G having the maximum fulfillment $f(G)$, by Eqs. (5) and (7) we have

$$\begin{aligned} f(G) &= f^*(P) + \sum_{x \in C(P,v_s) \cup C(P,v_t)} \mathrm{dem}^*(x) \\ &= f\left(P, -\sup(v_s), -\sup(v_t)\right) \\ &\le h\left(G^*, -\sup(v_s), -\sup(v_t)\right). \end{aligned} \tag{8}$$

Conversely, every $(-\sup(v_s), -\sup(v_t))$-separated partition P' of G^* yields a partition of G with fulfillment $f\left(P', -\sup(v_s), -\sup(v_t)\right)$, and hence by Eq. (7) we have

$$f(G) \ge h\left(G^*, -\sup(v_s), -\sup(v_t)\right). \tag{9}$$

By Eqs. (8) and (9) we have

$$f(G) = h\left(G^*, -\sup(v_s), -\sup(v_t)\right). \tag{10}$$

Thus, by Eq. (10) one can compute $f(G)$ from $h(G^*, j, k)$ in time $O(1)$.

Case (b): *one of v_s and v_t is a supply vertex and the other is a demand vertex in G.*

In this case one may assume without loss of generality that v_s is a supply vertex and v_t is a demand vertex. Then there are the following three cases (i), (ii) and (iii) for the partition of G having the maximum fulfillment, as illustrated in Fig. 8:

(i) v_t is supplied power from v_s;
(ii) v_t is supplied power from a supply vertex w other than v_s; and
(iii) v_t is not supplied power from any supply vertex.

For Case (i), $C(P, v_s) = C(P, v_t) = C(P)$ and hence P is a connected partition as illustrated in Fig. 8(i). Since v_t is supplied power from v_s, we have

$$\mathrm{dem}(v_t) + \sum_{x \in C(P)} \mathrm{dem}^*(x) \le \sup(v_s),$$

and hence

$$\sum_{x \in C(P)} \mathrm{dem}^*(x) \le \sup(v_s) - \mathrm{dem}(v_t).$$

Thus, Eq. (1) holds for P with $i = -\sup(v_s) + \mathrm{dem}(v_t) (\le 0)$, and hence P is an i-connected partition of G_u^*. Thus, by Eqs. (2) and (6) we have

$$\begin{aligned} f(G) &= \left(f^*(P) + \sum_{x \in C(P)} \mathrm{dem}^*(x) \right) + \mathrm{dem}(v_t) \\ &= f\left(P, -\sup(v_s) + \mathrm{dem}(v_t)\right) + \mathrm{dem}(v_t) \\ &= g\left(G^*, -\sup(v_s) + \mathrm{dem}(v_t)\right) + \mathrm{dem}(v_t). \end{aligned} \tag{11}$$

Similarly, for Case (ii), P is a (j, k)-separated partition of G^* for $j = -\sup(v_s)$ and $k = \mathrm{dem}(v_t)$, and hence we have

$$f(G) = h\left(G^*, -\sup(v_s), \mathrm{dem}(v_t)\right) + \mathrm{dem}(v_t). \tag{12}$$

For Case (iii), $C(P, v_t) = \{v_t\}$ since $|C| = 1$ for every component C of P containing no supply vertex. We therefore have

$$\sum_{x \in C(P,v_t)} \mathrm{dem}^*(x) = \mathrm{dem}^*(v_t) = 0,$$

and hence P is a (j, k)-separated partition of G^* for $j = -\sup(v_s)$ and $k = 0$. We thus have

$$f(G) = h\left(G^*, -\sup(v_s), 0\right). \tag{13}$$

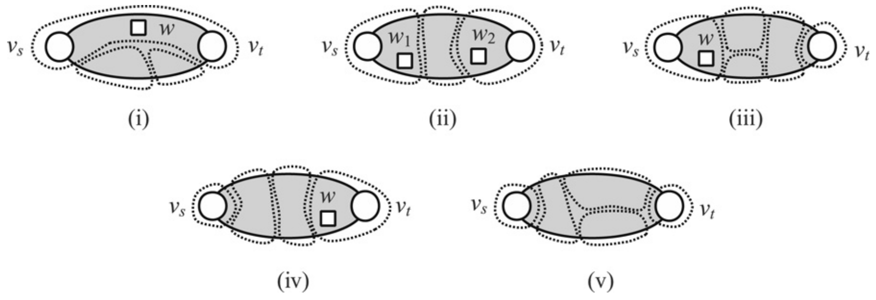


Fig. 9. Partitions of G for the five cases in Case (c).

Clearly Case (i) occurs only if $\sup(v_s) \geq \operatorname{dem}(v_t)$. Thus, by Eqs. (11)–(13) we have

$$f(G) = \max\left\{g\left(G^*, -\sup(v_s) + \operatorname{dem}(v_t)\right) + \operatorname{dem}(v_t), h\left(G^*, -\sup(v_s), \operatorname{dem}(v_t)\right) + \operatorname{dem}(v_t),\right.$$
$$\left. h\left(G^*, -\sup(v_s), 0\right)\right\} \tag{14}$$

if $\sup(v_s) \geq \operatorname{dem}(v_t)$, and

$$f(G) = \max\left\{h\left(G^*, -\sup(v_s), \operatorname{dem}(v_t)\right) + \operatorname{dem}(v_t), h\left(G^*, -\sup(v_s), 0\right)\right\} \tag{15}$$

if $\sup(v_s) < \operatorname{dem}(v_t)$.

Thus, by Eqs. (14) and (15) one can compute $f(G)$ from $g(G^*, i)$ and $h(G^*, j, k)$ in time $O(1)$.

Case (c): *both v_s and v_t are demand vertices in G.*

In this case, there are the following five cases (i)–(v), as illustrated in Fig. 9:

(i) both v_s and v_t are supplied power from the same supply vertex w in G;
(ii) v_s and v_t are supplied power from different supply vertices w_1 and w_2 in G, respectively;
(iii) v_s is supplied power from a supply vertex w in G, and v_t is not supplied power;
(iv) v_s is not supplied power, and v_t is supplied power from a supply vertex w in G; and
(v) both v_s and v_t are not supplied power.

Noting that $|C| = 1$ for every component C of P containing no supply vertex, one can easily observe

$$f(G) = \max\left\{g\left(G^*, \operatorname{dem}(v_s) + \operatorname{dem}(v_t)\right) + \operatorname{dem}(v_s) + \operatorname{dem}(v_t), h\left(G^*, \operatorname{dem}(v_s), \operatorname{dem}(v_t)\right) + \operatorname{dem}(v_s) + \operatorname{dem}(v_t),\right.$$
$$\left. h\left(G^*, \operatorname{dem}(v_s), 0\right) + \operatorname{dem}(v_s), h\left(G^*, 0, \operatorname{dem}(v_t)\right) + \operatorname{dem}(v_t), h(G^*, 0, 0)\right\}. \tag{16}$$

Thus, by Eq. (16) one can compute $f(G)$ from $g(G^*, i)$ and $h(G^*, j, k)$ in time $O(1)$.

3.3. How to compute $g(G_u^*, i)$ and $h(G_u^*, j, k)$

In this subsection, we explain how to compute $g(G_u^*, i)$ and $h(G_u^*, j, k)$ for each node u of T.

We first compute $g(G_u^*, i)$ and $h(G_u^*, j, k)$ for each leaf u of T, for which G_u^* contains exactly one edge as illustrated in Fig. 2(a). Since the two terminals of G_u^* are regarded as demand vertices of demands zero, by Eq. (6) we have

$$g(G_u^*, i) = \begin{cases} 0 & \text{if } -m_s \leq i \leq 0; \\ -\infty & \text{otherwise.} \end{cases} \tag{17}$$

Similarly, by Eq. (7) we have

$$h(G_u^*, j, k) = \begin{cases} 0 & \text{if } -m_s \leq j, k \leq 0; \\ -\infty & \text{otherwise.} \end{cases} \tag{18}$$

We next compute $g(G_u^*, i)$ and $h(G_u^*, j, k)$ for each internal node u of T from the counterparts of the two children of u in T.

We first consider a parallel connection.

[Parallel connection]

Let $G_u = G_1 \parallel G_2$, and let $v_s = v_s(G_u^*)$ and $v_t = v_t(G_u^*)$. (See Figs. 2(c), 10 and 11.)

We first compute $h(G_u^*, j, k)$. Every separated partition P of G_u^* can be obtained by combining a separated partition P_1 of G_1^* with a separated partition P_2 of G_2^*, as illustrated in Fig. 10. We thus know that, for each pair $(j, k) \in (\mathbb{Z}_{m_s})^2$, $h(G_u^*, j, k)$ can be computed as follows:

$$h(G_u^*, j, k) = \max_{j_1, j_2, k_1, k_2} \{h(G_1^*, j_1, k_1) + h(G_2^*, j_2, k_2)\} \tag{19}$$

where the maximum is taken over all integers j_1, j_2, k_1 and k_2 such that

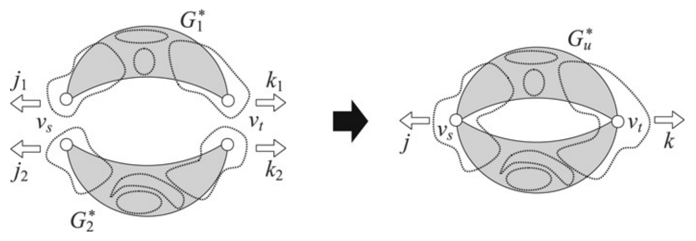


Fig. 10. Combining a partition P_1 of G_1^* and a partition P_2 of G_2^* to a separated partition P of $G_u^* = G_1^* \parallel G_2^*$.

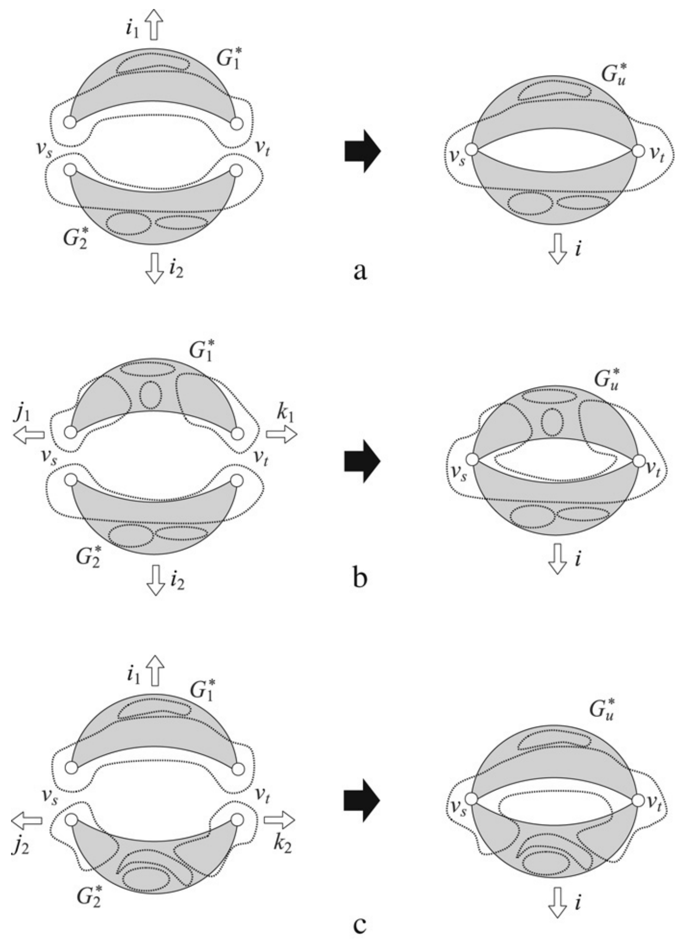


Fig. 11. Combining a partition P_1 of G_1^* and a partition P_2 of G_2^* to a connected partition P of $G_u^* = G_1^* \parallel G_2^*$.

(a) $j_1, j_2, k_1, k_2 \in \mathbb{Z}_{m_s}$;
(b) $j_1 + j_2 = j$ and $k_1 + k_2 = k$;
(c) if $j \leq 0$, then $j_1, j_2 \leq 0$;
(d) if $j > 0$, then exactly one of the two integers j_1 and j_2 is positive;
(e) if $k \leq 0$, then $k_1, k_2 \leq 0$; and
(f) if $k > 0$, then exactly one of the two integers k_1 and k_2 is positive.

We next compute $g(G_u^*, i)$. Every connected partition P of G_u^* can be obtained by combining a partition P_1 of G_1^* with a partition P_2 of G_2^*, as illustrated in Fig. 11(a), (b) and (c). There are the following three Cases (a), (b) and (c) to consider, and we define the three functions $g^a(G_u^*, i)$, $g^b(G_u^*, i)$ and $g^c(G_u^*, i)$ for the three cases, respectively.

Case (a): *both P_1 and P_2 are connected partitions.* (See Fig. 11(a).)

We define $g^a(G_u^*, i)$ for each integer $i \in \mathbb{Z}_{m_s}$, as follows:

$$g^a(G_u^*, i) = \max_{i_1, i_2}\{g(G_1^*, i_1) + g(G_2^*, i_2)\} \tag{20}$$

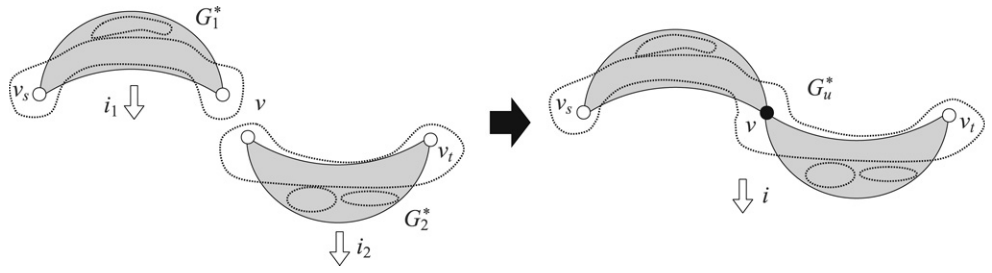


Fig. 12. Combining a partition P_1 of G_1^* and a partition P_2 of G_2^* to a connected partition P of G_u^*, where $G_u = G_1 \bullet G_2$.

where the maximum is taken over all integers i_1 and i_2 such that

(a) $i_1, i_2 \in \mathbb{Z}_{m_s}$;
(b) $i_1 + i_2 = i$;
(c) if $i \le 0$, then $i_1, i_2 \le 0$; and
(d) if $i > 0$, then exactly one of the two integers i_1 and i_2 is positive.

Case (b): *P_1 is a separated partition and P_2 is a connected partition.* (See Fig. 11(b).)
We define $g^b(G_u^*, i)$ for each integer $i \in \mathbb{Z}_{m_s}$, as follows:

$$g^b(G_u^*, i) = \max_{j_1,k_1,i_2} \{h(G_1^*, j_1, k_1) + g(G_2^*, i_2)\} \tag{21}$$

where the maximum is taken over all integers j_1, k_1 and i_2 such that

(a) $j_1, k_1, i_2 \in \mathbb{Z}_{m_s}$;
(b) $j_1 + k_1 + i_2 = i$;
(c) if $i \le 0$, then $j_1, k_1, i_2 \le 0$; and
(d) if $i > 0$, then exactly one of the three integers j_1, k_1 and i_2 is positive.

Case (c): *P_1 is a connected partition and P_2 is a separated partition.* (See Fig. 11(c).)
Analogously to Case (b), we define $g^c(G_u^*, i)$ for each integer $i \in \mathbb{Z}_{m_s}$, as follows:

$$g^c(G_u^*, i) = \max_{i_1,j_2,k_2} \{g(G_1^*, i_1) + h(G_2^*, j_2, k_2)\} \tag{22}$$

where the maximum is taken over all integers i_1, j_2 and k_2 such that

(a) $i_1, j_2, k_2 \in \mathbb{Z}_{m_s}$;
(b) $i_1 + j_2 + k_2 = i$;
(c) if $i \le 0$, then $i_1, j_2, k_2 \le 0$; and
(d) if $i > 0$, then exactly one of the three integers i_1, j_2 and k_2 is positive.

From g^a, g^b and g^c above, one can compute $g(G_u^*, i)$ as follows:

$$g(G_u^*, i) = \max\{g^a(G_u^*, i), g^b(G_u^*, i),\ g^c(G_u^*, i)\}. \tag{23}$$

We next consider a series connection.

[Series connection]

Let $G_u = G_1 \bullet G_2$, and let v be the vertex of G identified by the series connection, that is, $v = v_t(G_1) = v_s(G_2)$. (See Figs. 2(b), 12 and 13.) We define $sd(v)$ as follows:

$$sd(v) = \begin{cases} \sup(v) & \text{if } v \text{ is a supply vertex,} \\ -\text{dem}(v) & \text{if } v \text{ is a demand vertex.} \end{cases}$$

For the sake of convenience, we define $\text{dem}(w) = 0$ for each supply vertex w in G.

We first compute $g(G_u^*, i)$. Every connected partition P of G_u^* can be obtained by combining a connected partition P_1 of G_1^* with a connected partition P_2 of G_2^*, as illustrated in Fig. 12. Therefore, $g(G_u^*, i)$ can be computed for each integer $i \in \mathbb{Z}_{m_s}$, as follows:

$$g(G_u^*, i) = \max_{i_1,i_2}\{g(G_1^*, i_1) + g(G_2^*, i_2) + \text{dem}(v)\} \tag{24}$$

where the maximum is taken over all integers i_1 and i_2 such that

(a) $i_1, i_2 \in \mathbb{Z}_{m_s}$;
(b) $i_1 + i_2 + sd(v) = i$;
(c) if $i \le 0$, then v is a demand vertex and $i_1, i_2 \le 0$; and

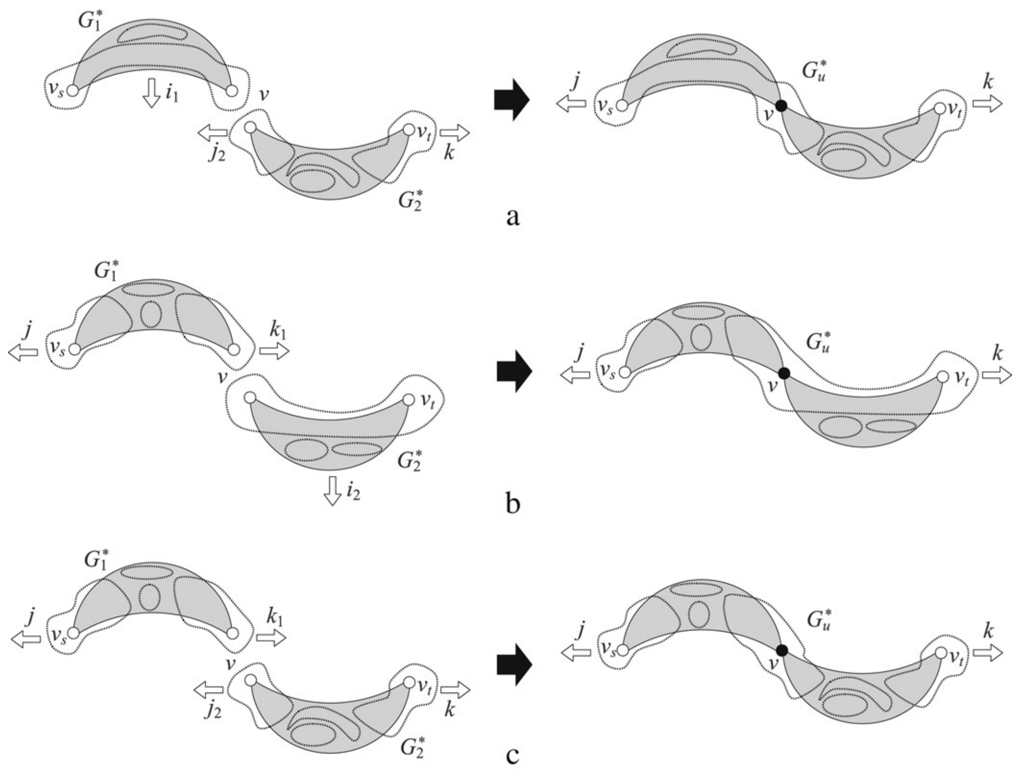


Fig. 13. Combining a partition P_1 of G_1^* and a partition P_2 of G_2^* to a separated partition P of G_u^*, where $G_u = G_1 \bullet G_2$.

(d) if $i > 0$, then exactly one of the three integers i_1, i_2 and $sd(v)$ is positive.

If such integers i_1 and i_2 do not exist, then we let $g(G_u^*, i) = -\infty$.

We next compute $h(G_u^*, j, k)$. Every separated partition P of G_u^* can be obtained by combining a partition P_1 of G_1^* with a partition P_2 of G_2^*, as illustrated in Fig. 13(a), (b) and (c). There are the following three Cases (a), (b) and (c) to consider, and we define the three functions $h^a(G_u^*, j, k)$, $h^b(G_u^*, j, k)$ and $h^c(G_u^*, j, k)$ for the three cases, respectively.

Case (a): *P_1 is a connected partition and P_2 is a separated partition.* (See Fig. 13(a).)

We define $h^a(G_u^*, j, k)$ for each pair (j, k), as follows:

$$h^a(G_u^*, j, k) = \max_{i_1, j_2}\{g(G_1^*, i_1) + h(G_2^*, j_2, k) + \mathrm{dem}(v)\} \tag{25}$$

where the maximum is taken over all integers i_1 and j_2 such that

(a) $i_1, j_2 \in \mathbb{Z}_{m_s}$;
(b) $i_1 + j_2 + sd(v) = j$;
(c) if $j \le 0$, then v is a demand vertex and $i_1, j_2 \le 0$; and
(d) if $j > 0$, then exactly one of the three integers i_1, j_2 and $sd(v)$ is positive.

If such integers i_1 and j_2 do not exist, then we define $h^a(G_u^*, j, k) = -\infty$.

Case (b): *P_1 is a separated partition and P_2 is a connected partition.* (See Fig. 13(b).)

Analogously to Case (a), we define $h^b(G_u^*, j, k)$ for each pair (j, k), as follows:

$$h^b(G_u^*, j, k) = \max_{k_1, i_2}\{h(G_1^*, j, k_1) + g(G_2^*, i_2) + \mathrm{dem}(v)\} \tag{26}$$

where the maximum is taken over all integers k_1 and i_2 such that

(a) $k_1, i_2 \in \mathbb{Z}_{m_s}$;
(b) $k_1 + i_2 + sd(v) = k$;
(c) if $k \le 0$, then v is a demand vertex and $k_1, i_2 \le 0$; and
(d) if $k > 0$, then exactly one of the three integers k_1, i_2 and $sd(v)$ is positive.

If such integers k_1 and i_2 do not exist, then we define $h^b(G_u^*, j, k) = -\infty$.

Case (c): *both P_1 and P_2 are separated partitions.* (See Fig. 13(c).)

In this case, either (i) all demand vertices in $C(P_1, v) \cup C(P_2, v)$ are supplied power or (ii) none of them is supplied power. For the first case (i), we define $h^{\mathrm{i}}(G_u^*, j, k)$ for each pair (j, k) as follows:

$$h^{\mathrm{i}}(G_u^*, j, k) = \max_{k_1, j_2}\{h(G_1^*, j, k_1) + h(G_2^*, j_2, k) + \mathrm{dem}(v)\} \tag{27}$$

where the maximum is taken over all integers k_1 and j_2 such that

(a) $k_1, j_2 \in \mathbb{Z}_{m_s}$;
(b) $k_1 + j_2 + sd(v) \geq 0$; and
(c) exactly one of the three integers k_1, j_2 and $sd(v)$ is positive.

If such integers k_1 and j_2 do not exist, then we define $h^{\mathrm{i}}(G_u^*, j, k) = -\infty$.

For the second case (ii), we define $h^{\mathrm{ii}}(G_u^*, j, k)$ for each pair (j, k) as follows:

$$h^{\mathrm{ii}}(G_u^*, j, k) = h(G_1^*, j, 0) + h(G_2^*, 0, k). \tag{28}$$

We then define $h^c(G_u^*, j, k)$ for each pair (j, k) as follows:

$$h^c(G_u^*, j, k) = \max\{h^{\mathrm{i}}(G_u^*, j, k), h^{\mathrm{ii}}(G_u^*, j, k)\}. \tag{29}$$

From h^a, h^b and h^c above, one can compute $h(G_u^*, j, k)$ as follows:

$$h(G_u^*, j, k) = \max\{h^a(G_u^*, j, k), h^b(G_u^*, j, k), h^c(G_u^*, j, k)\}. \tag{30}$$

3.4. Computation time

In this subsection, we show that our algorithm takes time $O\left((m_s)^4 n\right)$.

For each leaf u of T and all integers i, j and k, by Eqs. (17) and (18) one can easily compute $g(G_u^*, i)$ and $h(G_u^*, j, k)$ in time $O(m_s)$ and $O\left((m_s)^2\right)$, respectively. Since G is a simple series–parallel graph, G has at most $2n - 3$ edges and hence T has at most $2n - 3$ leaves. One can thus compute $g(G_u^*, i)$ and $h(G_u^*, j, k)$ for all leaves u of T in time $O\left((m_s)^2 n\right)$.

For each p-node u of T and all integers i, j and k in $\mathbb{Z}_{m_s}$ by Eqs. (19)–(23) one can compute $g(G_u^*, i)$ and $h(G_u^*, j, k)$ in time $O\left((m_s)^3\right)$ and $O\left((m_s)^4\right)$, respectively. For each s-node u of T and all integers i, j and k in $\mathbb{Z}_{m_s}$, by Eqs. (24)–(30) one can compute $g(G_u^*, i)$ and $h(G_u^*, j, k)$ in time $O\left((m_s)^2\right)$ and $O\left((m_s)^4\right)$, respectively. In this way one can compute $g(G_u^*, i)$ and $h(G_u^*, j, k)$ for each internal node u of T in time $O\left((m_s)^4\right)$ regardless of whether u is a p-node or an s-node. Since T is a binary tree and has at most $2n - 3$ leaves, T has at most $2n - 4$ internal node. One can thus compute $g(G^*, i)$ and $h(G^*, j, k)$ in time $O\left((m_s)^4 n\right)$ since $G = G_r$ for the root r of T.

By Eqs. (10) and (14)–(16) one can compute the maximum fulfillment $f(G)$ of G from $g(G^*, i)$ and $h(G^*, j, k)$ in time $O(1)$. Thus, the maximum partition problem can be solved in time $O\left((m_s)^4 n\right)$. This completes a proof of Theorem 1.

4. Partial *k*-trees

In this section we have the following theorem.

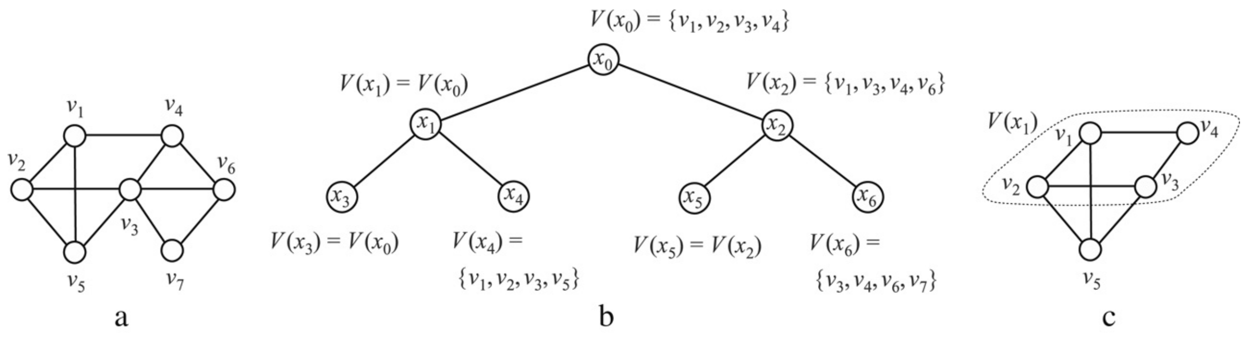
Theorem 2. *The maximum partition problem can be solved in time $O\left((m_s)^{2(k+1)} n\right)$ for every partial k-tree, where k is a constant.*

The algorithm for partial k-trees is similar to that for series–parallel graphs in the previous section. So we only give an outline of the algorithm.

A graph G is a *k-tree* if either it is a complete graph on k vertices or it has a vertex u whose neighbors induce a clique of size k and $G - \{u\}$ is again a k-tree. A graph is a *partial k-tree* if it is a subgraph of a k-tree [2].

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. 14(b).) The tree structure is called a *binary decomposition tree T of G* [1,3]. Each node u of T corresponds to a set $V(u)$ of $k + 1$ or fewer vertices of G, and corresponds to a subgraph G_u of G induced by the set $\bigcup\{V(w) \mid w \text{ is a descendant of } u \text{ in } T\}$. For example, Fig. 14 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_u. Let π be the number of all partitions of set $V(u)$ into pairwise disjoint nonempty subsets. Then $\pi \leq (k+1)^{k+1}$ and hence π is a constant whenever k is a constant. For a partial k-tree G, we consider π kinds of partitions of G_u. Let Q_i, $1 \leq i \leq \pi$, be the ith partition of set $V(u)$, let $\rho(i)$ be the number of subsets in the partition Q_i, and let $Q_i = \{V_1, V_2, \ldots, V_{\rho(i)}\}$. Clearly $1 \leq \rho(i) \leq k + 1$. In every partition P of



$V(x_0) = \{v_1, v_2, v_3, v_4\}$
$V(x_1) = V(x_0)$
$V(x_2) = \{v_1, v_3, v_4, v_6\}$
$V(x_3) = V(x_0)$
$V(x_4) = \{v_1, v_2, v_3, v_5\}$
$V(x_5) = V(x_2)$
$V(x_6) = \{v_3, v_4, v_6, v_7\}$
a
b
c

Fig. 14. (a) A partial 3-tree G, (b) its binary decomposition tree T, and (c) a subgraph G_{x_1}.

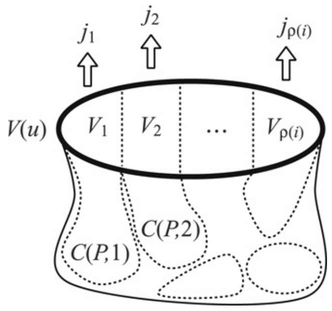


Fig. 15. Partition P of G_u of ith kind.

G_u of the ith kind, its lth connected component, $1 \le l \le \rho(i)$, contains all the vertices in the lth subset $V_l(\subseteq V(u))$ in Q_i. (See Fig. 15.) Let $C(P, l)$ be the set of all vertices in the lth connected component. Let G_u^* be a graph obtained from G_u by regarding each vertex in $V(u)$ as a demand vertex of demand zero; we denote by $\text{dem}^*(x)$ the demand of a demand vertex x in G_u^*, and hence

$$\text{dem}^*(x) = \begin{cases} 0 & \text{if } x \in V(u); \\ \text{dem}(x) & \text{otherwise.} \end{cases}$$

For each $\rho(i)$-tuple $(j_1, j_2, \dots, j_{\rho(i)})$ of integers in $\mathbb{Z}_{m_s}$, we call the partition P of G_u^* a $(j_1, j_2, \dots, j_{\rho(i)})$-partition if P satisfies the following two conditions (a) and (b) for each index l, $1 \le l \le \rho(i)$:

(a) if $j_l > 0$, then $C(P, l)$ contains exactly one supply vertex w and

$$j_l + \sum_{x \in C(P,l)-\{w\}} \text{dem}^*(x) \le \text{sup}(w);$$

and

(b) if $j_l \le 0$, then $C(P, l)$ contains no supply vertex and

$$\sum_{x \in C(P,l)} \text{dem}^*(x) \le |j_l| = -j_l.$$

Let $f(P, j_1, j_2, \dots, j_{\rho(i)}) = f(P) + \sum \text{dem}^*(x)$, where the summation is taken over all vertices $x \in C(P, l)$ such that $1 \le l \le \rho(i)$ and $-m_s \le j_l \le 0$. We consider a set of functions $h_i(G_u^*, j_1, j_2, \dots, j_{\rho(i)})$, $1 \le i \le \pi$, defined as follows: for a partial k-tree G_u^* and a $\rho(i)$-tuple $(j_1, j_2, \dots, j_{\rho(i)})$ of integers, let

$$h_i(G_u^*, j_1, j_2, \dots, j_{\rho(i)}) = \max\{f(P, j_1, j_2, \dots, j_{\rho(i)}) \mid G_u^* \text{ has a } (j_1, j_2, \dots, j_{\rho(i)})\text{-partition } P\}.$$

If G_u^* has no $(j_1, j_2, \dots, j_{\rho(i)})$-partition, then let $h_i(G_u^*, j_1, j_2, \dots, j_{\rho(i)}) = -\infty$. One can observe that the set of functions for an internal node u of T can be computed from the counterparts of the two children of u in T in time $O\left((2m_s + 1)^{2(k+1)}\right)$. Thus, the set of functions for G_r can be computed in time $O\left((2m_s + 1)^{2(k+1)}n\right)$ where r is the root of T. One can immediately compute $f(G)$ from the set of functions for $G = G_r$. The hidden coefficient in the complexity is $\pi^2 \left(\le (k+1)^{2(k+1)}\right)$.

5. Conclusions

In this paper we first obtained a pseudo-polynomial-time algorithm to compute the maximum fulfillment $f(G)$ of a given series–parallel graph G. The algorithm takes time $O\left((m_s)^4 n\right)$, and hence takes polynomial time if m_s is bounded by a polynomial in n. It is easy to modify the algorithm so that it actually finds a partition of a series–parallel graph. We then showed that our algorithm for series–parallel graphs can be extended for partial k-trees – that is, graphs with bounded tree-width [1–3]. The extended algorithm takes time $O\left((m_s)^{2(k+1)} n\right)$.

As we mentioned in Section 1, one wishes to partition a graph G into connected components so that each component C has exactly one supply vertex whose supply is no less than the sum of demands of all demand vertices in C. The *partition problem* is a decision problem which asks whether G has such a partition. The partition problem can be solved in linear time for trees [9]. However, the partition problem is NP-complete for series–parallel graphs, because the "set partition problem" [7] can be easily reduced to the partition problem for a complete bipartite graph $K_{2,n-2}$ in linear time and $K_{2,n-2}$ is a series–parallel graph. Using the algorithms for the maximum partition problem in the paper, one can solve the partition problem in time $O\left((m_s)^4 n\right)$ and $O\left((m_s)^{2(k+1)} n\right)$ for series–parallel graphs and partial k-trees, respectively. However, slightly modifying the algorithms, one can improve the complexities to $O\left((m_s)^2 n\right)$ and $O\left((m_s)^{2k} n\right)$ for series–parallel graphs and partial k-trees, respectively.

References

[1] S. Arnborg, J. Lagergren, D. Seese, Easy problems for tree-decomposable graphs, J. Algorithms 12 (2) (1991) 308–340.
[2] S. Arnborg, A. Proskurowski, Linear time algorithms for NP-hard problems restricted to partial k-trees, Discrete Appl. Math. 23 (1989) 11–24.
[3] H.L. Bodlaender, Polynomial algorithms for graph isomorphism and chromatic index on partial k-trees, J. Algorithms 11 (4) (1990) 631–643.
[4] N.G. Boulaxis, M.P. Papadopoulos, Optimal feeder routing in distribution system planning using dynamic programming technique and GIS facilities, IEEE Trans. Power Delivery 17 (1) (2002) 242–247.
[5] B. Courcelle, The monadic second-order logic of graphs I: Recognizable sets of finite graphs, Inform. Comput. 85 (1990) 12–75.
[6] B. Courcelle, M. Mosbath, Monadic second-order evaluations on tree-decomposable graphs, Theoret. Comput. Sci. 109 (1993) 49–82.
[7] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, San Francisco, CA, 1979.
[8] O.H. Ibarra, C.E. Kim, Fast approximation algorithms for the knapsack and sum of subset problems, J. Assoc. Comput. Mach. 22 (4) (1975) 463–468.
[9] T. Ito, X. Zhou, T. Nishizeki, Partitioning trees of supply and demand, Internat. J. Foundations Comput. Sci. 16 (4) (2005) 803–827.
[10] A.B. Morton, I.M.Y. Mareels, An efficient brute-force solution to the network reconfiguration problem, IEEE Trans. Power Delivery 15 (3) (2000) 996–1000.
[11] K. Takamizawa, T. Nishizeki, N. Saito, Linear-time computability of combinatorial problems on series–parallel graphs, J. Assoc. Comput. Mach. 29 (3) (1982) 623–641.
[12] J.A. Telle, A. Proskurowski, Algorithms for vertex partitioning problems on partial k-trees, SIAM J. Discrete Math. 10 (4) (1997) 529–550.
[13] J.-H. Teng, C.-N. Lu, Feeder-switch relocation for customer interruption cost minimization, IEEE Trans. Power Delivery 17 (1) (2002) 254–259.