

Partitioning Graphs of Supply and Demand

Takehiro Ito

Graduate School of Information Sciences, Tohoku University,
Aoba-yama 6-6-05, Sendai, 980-8579, Japan.
take@nishizeki.ecei.tohoku.ac.jp

Abstract. Suppose that each vertex of a graph G is either a supply vertex or a demand vertex and is assigned a positive real number, called the supply or the 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 so that each component C either has no supply vertex or has exactly one supply vertex whose supply is at least the sum of demands in C , and wishes to maximize the fulfillment, that is, the sum of demands in all components with supply vertices. This maximization problem is known to be NP-hard even for trees having exactly one supply vertex and strongly NP-hard for general graphs. In this paper, we focus on the approximability of the problem. We first show that the problem is MAXSNP-hard and hence there is no polynomial-time approximation scheme (PTAS) for general graphs unless $P = NP$. We then present a fully polynomial-time approximation scheme (FPTAS) for trees. The FPTAS can be extended for series-parallel graphs and partial k -trees, that is, graphs with bounded treewidth, if there is exactly one supply vertex in the graph. This is a joint work with E. Demaine, T. Nishizeki and X. Zhou.

Keywords: approximation algorithm, demand, FPTAS, maximum partition problem, MAXSNP-hard, supply, trees.

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 . Each vertex $u \in V_s$ is called a *supply vertex* and is assigned a positive real number $\text{sup}(u)$, called a *supply of u* , while each vertex $v \in V_d$ is called a *demand vertex* and is assigned a positive real number $\text{dem}(v)$, 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

M. Kaykobad and M. S. Rahman (Eds.): WALCOM 2007, pp. 162–179, 2007.

C has exactly one supply vertex whose supply is at least 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 at least 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 this problem the *maximum partition problem* [5]. The maximum partition problem has some applications to the power supply problem for power delivery networks [1, 5, 7]. Figure 1(a) illustrates a solution of the maximum partition problem for a graph, whose fulfillment is $(2 + 7) + (8 + 7) + (3 + 6) + (4 + 8) = 45$. In Fig. 1(a) 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 with a supply vertex is shaded.

Given a set A of integers and an upper bound (integer) b , the *maximum subset sum problem* [2, 3] asks to find a subset C of A such that the sum of integers in C is no greater than the bound b and is maximum among all such subsets C . The maximum subset sum problem can be reduced in linear time to the maximum partition problem for a particular tree, called a star, with exactly one supply vertex at the center, as illustrated in Fig. 1(c) [5]. Since the maximum subset sum problem is NP-hard, the maximum partition problem is also NP-hard even for stars. Thus it is very unlikely that the maximum partition problem can be exactly solved in polynomial time even for trees. Since there is a fully polynomial-time approximation scheme (FPTAS) for the maximum subset sum problem [3], one may expect that there is an FPTAS for the maximum partition problem.

In this paper, we study the approximability of the maximum partition problem. We first show that the maximum partition problem is MAXSNP-hard, and hence there is no polynomial-time approximation scheme (PTAS) for the problem on general graphs unless $P = NP$. We then present an FPTAS for trees. The FPTAS for trees can be extended to series-parallel graphs and partial k -trees, that is, graphs with bounded treewidth, if there is exactly one supply vertex in a graph [4]. Figure 1(b) depicts a partition of a tree T found by our FPTAS.

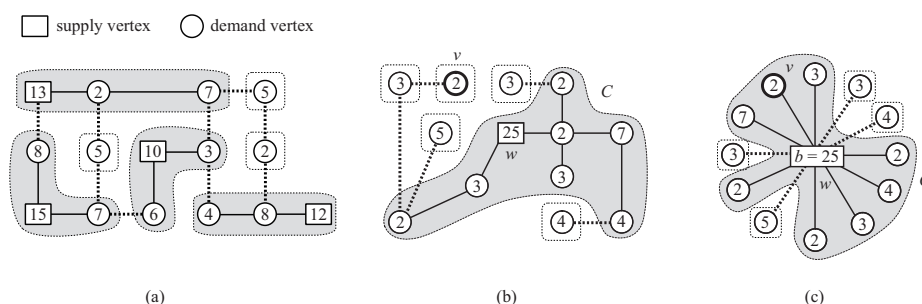


Fig. 1. (a) Partition of a graph G with maximum fulfillment, (b) partition of a tree T , and (c) a star S with a supply vertex at the center.

One might think that it would be straightforward to extend the FPTAS for the maximum subset sum problem in [3] to an FPTAS for the maximum partition problem on trees. However, this is not the case since we must take a graph structure into account. For example, the vertex v of demand 2 drawn by a thick circle in Fig. 1(b) cannot be supplied power even though the supply vertex w has marginal power $25 - (2 + 3 + 2 + 2 + 3 + 7 + 4) = 2$, while the vertex v in Fig. 1(c) can be supplied power from the supply vertex w in the star having the same supply and demands as in Fig. 1(b). Indeed, we not only extend the “scaling and rounding” technique but also employ many new ideas to derive our FPTAS. This is a joint work with E. Demaine, T. Nishizeki and X. Zhou [4, 5].

The rest of the paper is organized as follows. In Section 2 we show that the maximum partition problem is MAXSNP-hard. In Section 3 we present a pseudo-polynomial-time algorithm for trees. In Section 4 we present an FPTAS based on the algorithm in Section 3.

2 MAXSNP-hardness

The main result of this section is the following theorem.

Theorem 1. *The maximum partition problem is MAXSNP-hard for bipartite graphs.*

Proof. As in [8, 9], we use the concept of “L-reduction” which is a special kind of reduction that preserves approximability. Suppose that both A and B are maximization problems. Then we say that A can be L-reduced to B if there exist two polynomial-time algorithms Q and R and two positive constants α and β which satisfy the following two conditions (1) and (2) for each instance I_A of A :

- (1) the algorithm Q returns an instance $I_B = Q(I_A)$ of B such that $OPT_B(I_B) \leq \alpha \cdot OPT_A(I_A)$, where $OPT_A(I_A)$ and $OPT_B(I_B)$ denote the maximum solution values of I_A and I_B , respectively; and
- (2) for each feasible solution of I_B with value c_B , the algorithm R returns a feasible solution of I_A with value c_A such that $OPT_A(I_A) - c_A \leq \beta \cdot (OPT_B(I_B) - c_B)$.

Note that, by condition (2) of the L-reduction, R must return the optimal solution of I_A for the optimal solution of I_B .

We show that a MAXSNP-hard problem, called “3-occurrence MAX3SAT” [8, 9], can be L-reduced to the maximum partition problem for bipartite graphs. However, due to the page limitation, we only show in this extended abstract that condition (1) of the L-reduction holds.

We now show that condition (1) of the L-reduction holds for $\alpha = 26$. An instance Φ of 3-occurrence MAX3SAT consists of a collection of m clauses C_1, C_2, \dots, C_m on n variables x_1, x_2, \dots, x_n such that each clause has exactly three literals and each variable appears at most three times in the clauses. The 3-occurrence MAX3SAT problem is to find a truth assignment for the variables which satisfies the maximum number of clauses. Then it suffices to show that,

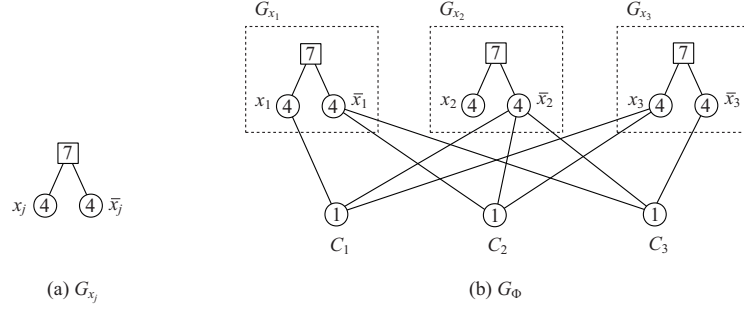


Fig. 2. (a) Variable gadget G_{x_j} , and (b) the bipartite graph G_{Φ} corresponding to an instance Φ with three clauses $C_1 = (x_1 \vee \bar{x}_2 \vee x_3)$, $C_2 = (\bar{x}_1 \vee \bar{x}_2 \vee x_3)$ and $C_3 = (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3)$.

from each instance Φ of 3-occurrence MAX3SAT, one can construct in polynomial time a bipartite graph G_{Φ} as an instance of the maximum partition problem such that

$$OPT_{MPP}(G_{\Phi}) \leq 26 \cdot OPT_{SAT}(\Phi), \quad (1)$$

where $OPT_{MPP}(G_{\Phi})$ is the maximum solution value of the maximum partition problem for G_{Φ} and $OPT_{SAT}(\Phi)$ is the maximum solution value of 3-occurrence MAX3SAT for Φ .

We first make a *variable gadget* G_{x_j} for each variable x_j , $1 \leq j \leq n$; G_{x_j} is a binary tree with three vertices as illustrated in Fig. 2(a); the root is a supply vertex of supply 7, and two leaves x_j and \bar{x}_j are demand vertices of demands 4. The graph G_{Φ} is constructed as follows. For each variable x_j , $1 \leq j \leq n$, put the variable gadget G_{x_j} to the graph, and for each clause C_i , $1 \leq i \leq m$, put a demand vertex C_i of demand 1 to the graph. Finally, for each clause C_i , $1 \leq i \leq m$, join a demand vertex x_j (or \bar{x}_j) in G_{x_j} with the demand vertex C_i if and only if the literal x_j (or \bar{x}_j) is in C_i , as illustrated in Fig. 2(b). Clearly, G_{Φ} can be constructed in polynomial time, and is a bipartite graph. It should be noted that, since each variable x_j , $1 \leq j \leq n$, appears at most three times in the clauses, the supply vertex in G_{x_j} has enough “power” to supply all demand vertices C_i whose corresponding clauses have x_j or \bar{x}_j . Then one can verify Eq. (1), whose proof is omitted from this extended abstract. *Q.E.D.*

3 Pseudo-polynomial-time algorithm for trees

The maximum partition problem is NP-hard even for trees. However, in this section, we give a pseudo-polynomial-time algorithm to solve the maximum partition problem for trees. The main result of this section is the following theorem.

Theorem 2. *The maximum partition problem can be solved for a tree $T = (V, E)$ in time $O(F^2n)$ if the demands and supplies are integers, where $n = |V|$ and $F = \min\{\sum_{v \in V_d} dem(v), \sum_{v \in V_s} sup(v)\}$.*

In the remainder of this section we give an algorithm to solve the maximum partition problem for trees in time $O(F^2n)$ as a proof of Theorem 2. In Subsection 3.1 we define some terms and present ideas of our algorithm. We then present our algorithm in Subsection 3.2. We finally show, in Subsection 3.3, that our algorithm takes time $O(F^2n)$.

3.1 Terms and ideas

We partition a 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 is called a *partition* of G . Thus a partition P of G corresponds to a family of sets of vertices in G . The *fulfillment* $f(P)$ of a partition P is the sum of demands of all demand vertices in components with supply vertices. 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 . Clearly $f(G) \leq F$. For the graph G in Fig. 1(a) the partition P has the maximum fulfillment, and hence $f(G) = f(P) = 45$.

One may assume without loss of generality that T is a rooted tree. Let r be the root of T . For each vertex v of T , we denote by T_v the subtree of T which is rooted at v and is induced by all descendants of v in T . We denote by \mathbb{R} the set of all real numbers. Let $\mathbb{R}^+ = \{x \in \mathbb{R} \mid x \geq 0\}$. Our idea is to consider three types of partitions of the rooted subtree T_v of T . The first is called a j -out partition, which can deliver an amount j of marginal power outside T_v through v . The second is called a j -in partition, which needs an amount j of deficient power to be delivered inside T_v through v . The third is called an isolated partition, in which the set containing v is a singleton. For these partitions, we introduce three functions $g_{\text{out}} : (\mathcal{T}, \mathbb{R}) \rightarrow \mathbb{R}^+ \cup \{-\infty\}$, $g_{\text{in}} : (\mathcal{T}, \mathbb{R}) \rightarrow \mathbb{R}^+ \cup \{+\infty\}$ and $g_0 : (\mathcal{T}, \mathbb{R}) \rightarrow \{0, +\infty\}$, where \mathcal{T} denotes the set of all trees. For $T_v \in \mathcal{T}$ and $x \in \mathbb{R}$, the value $g_{\text{out}}(T_v; x)$ represents the maximum marginal power of a j -out partition P of T_v such that $f(P) \geq x$, $g_{\text{in}}(T_v; x)$ represents the minimum deficient power of a j -in partition P of T_v such that $f(P) \geq x$, and $g_0(T_v; x) = 0$ if T_v has an isolated partition P such that $f(P) \geq x$, otherwise, $g_0(T_v; x) = +\infty$. Our idea is to compute g_{out} , g_{in} and g_0 from the leaves of T to the root r of T by means of dynamic programming.

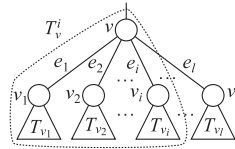


Fig. 3. Tree T_v .

Let v be a vertex of T , let v_1, v_2, \dots, v_l be the children of v ordered arbitrarily, and let e_i , $1 \leq i \leq l$, be the edge joining v and v_i , as illustrated in Fig. 3. T_{v_i} , $1 \leq i \leq l$, is the subtree of T which is rooted at v_i and is induced by all descendants of v_i in T . We denote by T_v^i the subtree of T which consists of the vertex v , the edges e_1, e_2, \dots, e_i and the subtrees $T_{v_1}, T_{v_2}, \dots, T_{v_i}$. In Fig. 3, T_v^i is indicated by a dotted line. Clearly $T_v = T_v^l$. For the sake of notational convenience, we denote by T_v^0 the subtree of a single vertex v .

Let P be a partition of a rooted subtree T_v of T , and let $C(P)$ be the set of all vertices in the connected component containing the root v of T_v in P , as illustrated in Figs. 4(a), 5(a) and 6(a). For each real number $j \in \mathbb{R}^+$, we now formally define j -out, j -in and isolated partitions as follows.

(a) A partition P of T_v is called a j -out partition if $C(P)$ contains a supply vertex w and $\sup(w) \geq j + \sum_{u \in C(P) - \{w\}} \text{dem}(u)$. (See Fig. 4(a).) A j -out partition of T_v corresponds to a partition of the whole tree T in which all demand vertices in $C(P)$ are supplied power from a supply vertex in T_v ; an amount j of power can be delivered outside T_v through v , and hence the “margin” of P is j . If a virtual demand vertex v_d with $\text{dem}(v_d) = j$ is joined to the root v of T_v as illustrated in Fig. 4(b), then v_d and all demand vertices in $C(P)$ can be supplied power from w in the resulting tree T_v^+ . A j -out partition P of T_v induces a partition P^+ of T_v^+ such that $f(P^+) = f(P) + j$, where the connected component containing v in P^+ contains the vertices in $C(P) \cup \{v_d\}$ and each of the other components in P^+ is the same as the counterpart in P , as illustrated in Fig. 4.

(b) A partition P of T_v is called a j -in partition if $C(P)$ contains no supply vertex and $\sum_{u \in C(P)} \text{dem}(u) \leq j$. (See Fig. 5(a).) Thus, if P is a j -in partition, then v is a demand vertex and $\text{dem}(v) \leq j$. A j -in partition of T_v corresponds to a partition of T in which all (demand) vertices in $C(P)$ including v are supplied power from a supply vertex outside T_v ; an amount j of power must be delivered inside T_v through v , and hence the “deficiency” of P is j . If a virtual supply vertex v_s with $\sup(v_s) = j$ is joined to the root v of T_v as illustrated in Fig. 5(b), then all (demand) vertices in $C(P)$ can be supplied power from v_s in the resulting

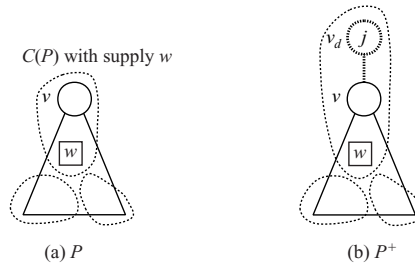


Fig. 4. (a) Tree T_v and (b) tree T_v^+ .

tree T_v^* . For a j -in partition P of T_v , let $f^*(P) = f(P) + \sum_{u \in C(P)} \text{dem}(u)$. Clearly, a j -in partition P of T_v induces a partition P^* of T_v^* such that $f(P^*) = f^*(P)$.

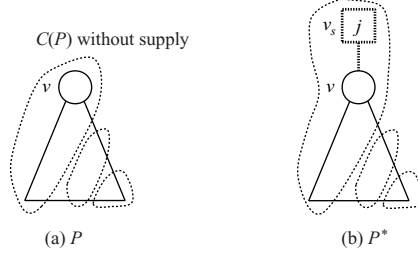


Fig. 5. (a) Tree T_v and (b) tree T_v^* .

(c) A partition P of T_v is called an *isolated partition* if $C(P)$ consists of a single demand vertex v . (See Fig. 6(a).) An isolated partition P of T_v corresponds to a partition P' of T in which v is not supplied power from any supply vertex in T . Such a partition P' of T does not always induce an isolated partition P of T_v . However, subdividing the component of P' containing v into singletons, one can transform P' to a partition P'' of T such that $f(P'') = f(P')$ and P'' induces an isolated partition P of T_v , as illustrated in Figs. 6(b) and (c).

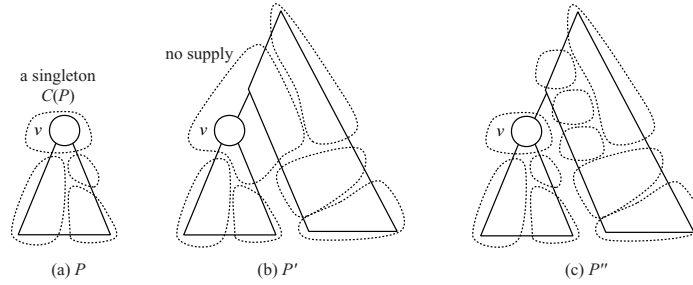


Fig. 6. (a) Tree T_v , and (b), (c) tree T .

We are now ready to give a formal definition of the three functions g_{out} , g_{in} and g_0 . We first define $g_{\text{out}} : (\mathcal{T}, \mathbb{R}) \rightarrow \mathbb{R}^+ \cup \{-\infty\}$ for a tree $T_v \in \mathcal{T}$ and a real number $x \in \mathbb{R}$, as follows:

$$g_{\text{out}}(T_v; x) = \max\{j \in \mathbb{R}^+ \mid T_v \text{ has a } j\text{-out partition } P \text{ such that } f(P) \geq x\} \quad (2)$$

Thus, $g_{\text{out}}(T_v; x)$ is the maximum amount j of “marginal power” of a j -out partition P such that $f(P) \geq x$. If T_v has no j -out partition P with $f(P) \geq x$ for any number $j \in \mathbb{R}^+$, then let $g_{\text{out}}(T_v; x) = -\infty$. We then similarly define

$g_{\text{in}} : (\mathcal{T}, \mathbb{R}) \rightarrow \mathbb{R}^+ \cup \{+\infty\}$ for a tree $T_v \in \mathcal{T}$ and a real number $x \in \mathbb{R}$, as follows:

$$g_{\text{in}}(T_v; x) = \min\{j \in \mathbb{R}^+ \mid T_v \text{ has a } j\text{-in partition } P \text{ such that } f^*(P) \geq x\}. \quad (3)$$

Thus, $g_{\text{in}}(T_v; x)$ is the minimum amount j of “deficient power” of a j -in partition P such that $f^*(P) \geq x$. If T_v has no j -in partition P with $f^*(P) \geq x$ for any number $j \in \mathbb{R}^+$, then let $g_{\text{in}}(T_v; x) = +\infty$. We finally define $g_0 : (\mathcal{T}, \mathbb{R}) \rightarrow \{0, +\infty\}$ for a tree $T_v \in \mathcal{T}$ and a real number $x \in \mathbb{R}$, as follows:

$$g_0(T_v; x) = \begin{cases} 0 & \text{if } T_v \text{ has an isolated partition } P \text{ such that } f(P) \geq x; \\ +\infty & \text{otherwise.} \end{cases} \quad (4)$$

The function g_{out} takes a value in $\mathbb{R}^+ \cup \{-\infty\}$, g_{in} takes a value in $\mathbb{R}^+ \cup \{+\infty\}$, and g_0 takes a value 0 or $+\infty$. Clearly, g_{out} is non-increasing, and both g_{in} and g_0 are non-decreasing. For any negative real number $x < 0$, $g_{\text{out}}(T_v; x) = g_{\text{out}}(T_v; 0)$, $g_{\text{in}}(T_v; x) = g_{\text{in}}(T_v; 0)$, and $g_0(T_v; x) = g_0(T_v; 0)$.

Our algorithm computes $g_{\text{out}}(T_v; x)$, $g_{\text{in}}(T_v; x)$ and $g_0(T_v; x)$ for each vertex v of T from the leaves to the root r of T by means of dynamic programming.

3.2 algorithm

We first show how to compute the maximum fulfillment $f(T)$ of a given tree T from $g_{\text{out}}(T; x)$, $g_{\text{in}}(T; x)$ and $g_0(T; x)$.

[How to compute $f(T)$]

Let $f_{\text{out}}(T_v)$ be the maximum fulfillment $f(P)$ taken over all j -out partitions P of T_v with $j \in \mathbb{R}^+$. Thus

$$f_{\text{out}}(T_v) = \max\{x \in \mathbb{R}^+ \mid g_{\text{out}}(T_v; x) \neq -\infty\}. \quad (5)$$

If $g_{\text{out}}(T_v; x) = -\infty$ for any number $x \in \mathbb{R}$, then let $f_{\text{out}}(T_v) = -\infty$. On the other hand, let $f_0(T_v)$ be the maximum fulfillment $f(P)$ taken over all isolated partitions P of T_v . Thus

$$f_0(T_v) = \max\{x \in \mathbb{R}^+ \mid g_0(T_v; x) \neq +\infty\}. \quad (6)$$

If $g_0(T_v; x) = +\infty$ for any number $x \in \mathbb{R}$, that is, v is a supply vertex, then let $f_0(T_v) = -\infty$. One can easily observe that

$$f(T_v) = \max\{f_{\text{out}}(T_v), f_0(T_v)\},$$

and hence

$$f(T) = f(T_r) = \max\{f_{\text{out}}(T_r), f_0(T_r)\}$$

for the root r of T . Note that a partition of $T = T_r$ with the maximum fulfillment $f(T)$ is either an isolated partition or a j -out partition for some number $j \in \mathbb{R}^+$.

We then explain how to compute $g_{\text{out}}(T_v; x)$, $g_{\text{in}}(T_v; x)$ and $g_0(T_v; x)$ for each vertex v of T .

[How to compute $g_{\text{out}}(T_v; x)$, $g_{\text{in}}(T_v; x)$ and $g_0(T_v; x)$]

We first compute $g_{\text{out}}(T_v^0; x)$, $g_{\text{in}}(T_v^0; x)$ and $g_0(T_v^0; x)$ for each vertex v of T as follows. Since T_v^0 consists of a single vertex v , $T_v = T_v^0$ if v is a leaf. If v is a demand vertex, then for any number $x \in \mathbb{R}$

$$g_{\text{out}}(T_v^0; x) = -\infty, \quad (7)$$

$$g_{\text{in}}(T_v^0; x) = \begin{cases} \text{dem}(v) & \text{if } x \leq \text{dem}(v); \\ +\infty & \text{otherwise,} \end{cases} \quad (8)$$

and

$$g_0(T_v^0; x) = \begin{cases} 0 & \text{if } x \leq 0; \\ +\infty & \text{otherwise.} \end{cases} \quad (9)$$

If v is a supply vertex, then for any number $x \in \mathbb{R}$

$$g_{\text{out}}(T_v^0; x) = \begin{cases} \text{sup}(v) & \text{if } x \leq 0; \\ -\infty & \text{otherwise,} \end{cases} \quad (10)$$

$$g_{\text{in}}(T_v^0; x) = +\infty, \quad (11)$$

and

$$g_0(T_v^0; x) = +\infty. \quad (12)$$

We next compute $g_{\text{out}}(T_v^i; x)$, $g_{\text{in}}(T_v^i; x)$ and $g_0(T_v^i; x)$, $1 \leq i \leq l$, for each internal vertex v of T from the counterparts of T_v^{i-1} and T_{v_i} , where l is the number of the children of v . (See Fig. 3.) Remember that $T_v = T_v^l$, and note that T_v^i is obtained from T_v^{i-1} and T_{v_i} by joining v and v_i as illustrated in Fig. 7 where T_v^{i-1} is indicated by a thin dotted line.

We first explain how to compute $g_{\text{out}}(T_v^i; x)$. Let P be a j -out partition of T_v^i such that $f(P) \geq x$ and $j = g_{\text{out}}(T_v^i; x) \neq -\infty$. Then there are the following four Cases (a)–(d):

Case (a): v_i is supplied power from a vertex in T_v^{i-1} ;

Case (b): v_i is not supplied power;

Case (c): v_i is supplied power from a vertex in T_{v_i} , and either v is a supply vertex or v is supplied power from a vertex in T_v^{i-1} ; and

Case (d): v is supplied power from a vertex in T_{v_i} .

Figures 7(a)–(d) illustrate the power flows in T_v^i for the four cases, where an arrow represents the direction of power supply, and a thick dotted line represents a deleted edge. For $x \in \mathbb{R}$ and $y \in \mathbb{R}$, we define $g_{\text{out}}^a(T_v^i; x, y)$, $g_{\text{out}}^b(T_v^i; x, y)$, $g_{\text{out}}^c(T_v^i; x, y)$ and $g_{\text{out}}^d(T_v^i; x, y)$ for Cases (a), (b), (c) and (d), respectively. Intuitively, x and y are the fulfillments of T_v^i and T_v^{i-1} , respectively.

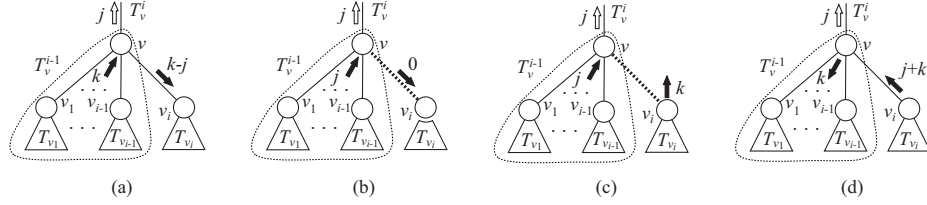


Fig. 7. Power flow in a j -out partition of T_v^i .

Case (a): v_i is supplied power from a vertex in T_v^{i-1} .

In this case, for some number $k \in \mathbb{R}^+$ with $k \geq j$, the j -out partition P of T_v^i can be obtained by merging a k -out partition P_1 of T_v^{i-1} and a $(k-j)$ -in partition P_2 of T_{v_i} such that $f(P) = f(P_1) + f^*(P_2)$. (See Fig. 7(a). The component of P containing v consists of the vertices in $C(P_1) \cup C(P_2)$, and each of the other components of P corresponds to a component of P_1 or P_2 .) Since $f(P) = f(P_1) + f^*(P_2) \geq x$, we have $f(P_1) \geq y$ and $f^*(P_2) \geq x - y$ for some number $y \in \mathbb{R}$. Since P_1 is a k -out partition of T_v^{i-1} with $f(P_1) \geq y$, one may assume by Eq. (2) that $k = g_{\text{out}}(T_v^{i-1}; y)$. Similarly one may assume that $k - j = g_{\text{in}}(T_{v_i}; x - y)$. Since $g_{\text{out}}(T_v^i; x) = j = k - (k - j)$, we define

$$g_{\text{out}}^a(T_v^i; x, y) = g_{\text{out}}(T_v^{i-1}; y) - g_{\text{in}}(T_{v_i}; x - y) \quad (13)$$

for Case (a).

Case (b): v_i is not supplied power.

In this case, P can be obtained by merging a j -out partition P_1 of T_v^{i-1} and an isolated partition P_2 of T_{v_i} such that $f(P_1) \geq y$ and $f(P_2) \geq x - y$ for some number $y \in \mathbb{R}$. (See Fig. 7(b). The family of sets of vertices corresponding to P is a union of two families corresponding to P_1 and P_2 .) Then $j = j - 0$, and hence let

$$g_{\text{out}}^b(T_v^i; x, y) = g_{\text{out}}(T_v^{i-1}; y) - g_0(T_{v_i}; x - y). \quad (14)$$

Case (c): v_i is supplied power from a vertex in T_{v_i} , and either v is a supply vertex or v is supplied power from a vertex in T_v^{i-1} .

In this case, for some number $k \in \mathbb{R}^+$, P can be obtained by merging a j -out partition P_1 of T_v^{i-1} and a k -out partition P_2 of T_{v_i} such that $f(P_1) \geq y$ and $f(P_2) \geq x - y$ for some numbers $k \in \mathbb{R}^+$ and $y \in \mathbb{R}$. (See Fig. 7(c). The family

of sets of vertices corresponding to P is a union of two families corresponding to P_1 and P_2 .) Then let

$$g_{\text{out}}^c(T_v^i; x, y) = \begin{cases} g_{\text{out}}(T_v^{i-1}; y) & \text{if } g_{\text{out}}(T_{v_i}; x - y) \neq -\infty; \\ -\infty & \text{if } g_{\text{out}}(T_{v_i}; x - y) = -\infty. \end{cases} \quad (15)$$

Case (d): v is supplied power from a vertex in T_{v_i} .

In this case, either v_i is a supply vertex or both v and v_i are supplied power from the same supply vertex in T_{v_i} . For some number $k \in \mathbb{R}^+$, P can be obtained by merging a k -in partition P_1 of T_v^{i-1} and a $(j+k)$ -out partition P_2 of T_{v_i} such that $f^*(P_1) \geq y$ and $f(P_2) \geq x - y$ for some numbers $k \in \mathbb{R}^+$ and $y \in \mathbb{R}$. (See Fig. 7(d). The component of P containing v consists of the vertices in $C(P_1) \cup C(P_2)$, and each of the other components of P corresponds to a component of P_1 or P_2 .) Then $j = (j+k) - k$, and hence let

$$g_{\text{out}}^d(T_v^i; x, y) = g_{\text{out}}(T_{v_i}; x - y) - g_{\text{in}}(T_v^{i-1}; y). \quad (16)$$

From g_{out}^a , g_{out}^b , g_{out}^c and g_{out}^d above one can compute $g_{\text{out}}(T_v^i; x)$ as follows:

$$g_{\text{out}}(T_v^i; x) = \max\{g_{\text{out}}^a(T_v^i; x, y), g_{\text{out}}^b(T_v^i; x, y), g_{\text{out}}^c(T_v^i; x, y), g_{\text{out}}^d(T_v^i; x, y) \mid y \in (\mathbb{R})\}$$

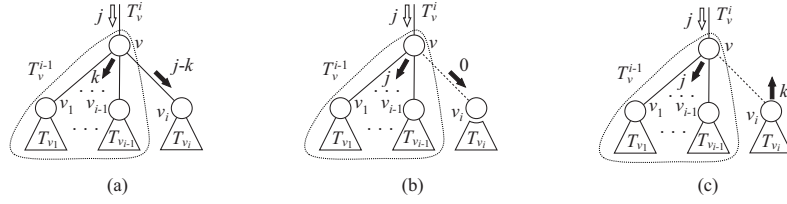


Fig. 8. Power flow in a j -in partition of T_v^i .

We next explain how to compute $g_{\text{in}}(T_v^i; x)$. Let P be a j -in partition such that $f^*(P) \geq x$ and $j = g_{\text{in}}(T_v^i; x) \neq +\infty$. Then there are the following three Cases (a), (b) and (c):

Case (a): $v_i \in C(P)$;

Case (b): $v_i \notin C(P)$, and v_i is not supplied power; and

Case (c): $v_i \notin C(P)$, and v_i is supplied power from a vertex in T_{v_i} .

Figures 8(a)–(c) illustrate the power flows in T_v^i for the three cases. We define $g_{\text{in}}^a(T_v^i; x, y)$, $g_{\text{in}}^b(T_v^i; x, y)$ and $g_{\text{in}}^c(T_v^i; x, y)$ for Cases (a), (b) and (c), respectively, as follows.

Case (a): $v_i \in C(P)$.

In this case, the j -in partition P can be obtained by merging a k -in partition P_1 of T_v^{i-1} and a $(j-k)$ -in partition P_2 of T_{v_i} such that $f^*(P_1) \geq y$ and

$f^*(P_2) \geq x - y$ for some numbers $k \in \mathbb{R}^+$ and $y \in \mathbb{R}$. (See Fig. 8(a).) Then $j = k + (j - k)$, and hence let

$$g_{\text{in}}^a(T_v^i; x, y) = g_{\text{in}}(T_v^{i-1}; y) + g_{\text{in}}(T_{v_i}; x - y). \quad (18)$$

Case (b): $v_i \notin C(P)$, and v_i is not supplied power.

In this case, P can be obtained by merging a j -in partition P_1 of T_v^{i-1} and an isolated partition P_2 of T_{v_i} such that $f^*(P_1) \geq y$ and $f(P_2) \geq x - y$ for some number $y \in \mathbb{R}$. (See Fig. 8(b).) Then $j = j + 0$, and hence let

$$g_{\text{in}}^b(T_v^i; x, y) = g_{\text{in}}(T_v^{i-1}; y) + g_0(T_{v_i}; x - y). \quad (19)$$

Case (c): $v_i \notin C(P)$, and v_i is supplied power from a vertex in T_{v_i} .

In this case, P can be obtained by merging a j -in partition P_1 of T_v^{i-1} and a k -out partition P_2 of T_{v_i} such that $f^*(P_1) \geq y$ and $f(P_2) \geq x - y$ for some numbers $k \in \mathbb{R}^+$ and $y \in \mathbb{R}$. (See Fig. 8(c).) Then let

$$g_{\text{in}}^c(T_v^i; x, y) = \begin{cases} g_{\text{in}}(T_v^{i-1}; y) & \text{if } g_{\text{out}}(T_{v_i}; x - y) \neq -\infty; \\ +\infty & \text{if } g_{\text{out}}(T_{v_i}; x - y) = -\infty. \end{cases} \quad (20)$$

From g_{in}^a , g_{in}^b and g_{in}^c above one can compute $g_{\text{in}}(T_v^i; x)$ as follows:

$$g_{\text{in}}(T_v^i; x) = \min\{g_{\text{in}}^a(T_v^i; x, y), g_{\text{in}}^b(T_v^i; x, y), g_{\text{in}}^c(T_v^i; x, y) \mid y \in \mathbb{R}\}. \quad (21)$$

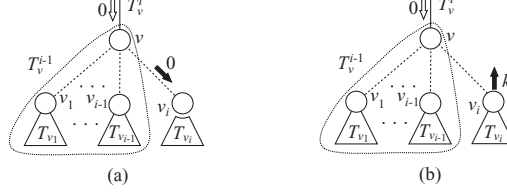


Fig. 9. Power flow in an isolated partition of T_v^i .

We finally explain how to compute $g_0(T_v^i; x)$. Let P be an isolated partition such that $f(P) \geq x$. There are the following two Cases (a) and (b), and we define $g_0^a(T_v^i; x, y)$ and $g_0^b(T_v^i; x, y)$ for Cases (a) and (b), respectively.

Case (a): v_i is a demand vertex and is not supplied power.

In this case, the isolated partition P can be obtained by merging an isolated partition P_1 of T_v^{i-1} and an isolated partition P_2 of T_{v_i} such that $f(P_1) \geq y$ and $f(P_2) \geq x - y$ for some number $y \in \mathbb{R}$. (See Fig. 9(a).) Then let

$$g_0^a(T_v^i; x, y) = g_0(T_v^{i-1}; y) + g_0(T_{v_i}; x - y). \quad (22)$$

Case (b): *either v_i is a supply vertex or v_i is supplied power (from a vertex in T_{v_i}).*

In this case P can be obtained by merging an isolated partition P_1 of T_v^{i-1} and a k -out partition P_2 of T_{v_i} such that $f(P_1) \geq y$ and $f(P_2) \geq x - y$ for some numbers $k \in \mathbb{R}^+$ and $y \in \mathbb{R}$. (See Fig. 9(b).) Then let

$$g_0^b(T_v^i; x, y) = \begin{cases} g_0(T_v^{i-1}; y) & \text{if } g_{\text{out}}(T_{v_i}; x - y) \neq -\infty; \\ +\infty & \text{if } g_{\text{out}}(T_{v_i}; x - y) = -\infty. \end{cases} \quad (23)$$

From g_0^a and g_0^b above one can compute $g_0(T_v^i; x)$ as follows:

$$g_0(T_v^i; x) = \min\{g_0^a(T_v^i; x, y), g_0^b(T_v^i; x, y) \mid y \in \mathbb{R}\}. \quad (24)$$

3.3 Proof of Theorem 2

We now show that our algorithm takes time $O(F^2n)$ as a proof of Theorem 2.

Since all the supplies and demands in T are integers, $f(P)$ and $f^*(P)$ are integers for any partition P of T_v . We denote by \mathbb{Z} the set of all integers. Let $\mathbb{Z}^+ = \{x \in \mathbb{Z} \mid x \geq 0\}$ and $\mathbb{Z}_F^+ = \{x \in \mathbb{Z} \mid 0 \leq x \leq F\}$. Define a function $\hat{g}_{\text{out}} : (\mathcal{T}, \mathbb{Z}) \rightarrow \mathbb{Z}^+ \cup \{-\infty\}$ for a tree $T_v \in \mathcal{T}$ and an integer $x \in \mathbb{Z}$ similarly as $g_{\text{out}} : (\mathcal{T}, \mathbb{R}) \rightarrow \mathbb{R}^+ \cup \{-\infty\}$ in Eq. (2):

$$\hat{g}_{\text{out}}(T_v; x) = \max\{j \in \mathbb{Z}^+ \mid T_v \text{ has a } j\text{-out partition } P \text{ such that } f(P) \geq x\}.$$

Define functions $\hat{g}_{\text{in}} : (\mathcal{T}, \mathbb{Z}) \rightarrow \mathbb{Z}^+ \cup \{+\infty\}$ and $\hat{g}_0 : (\mathcal{T}, \mathbb{Z}) \rightarrow \{0, +\infty\}$ similarly as g_{in} and g_0 in Eqs. (3) and (4). Define integral values $\hat{f}_{\text{out}}(T_v)$ and $\hat{f}_0(T_v)$ similarly as $f_{\text{out}}(T_v)$ and $f_0(T_v)$ in Eqs. (5) and (6):

$$\hat{f}_{\text{out}}(T_v) = \max\{x \in \mathbb{Z}^+ \mid \hat{g}_{\text{out}}(T_v; x) \neq -\infty\}; \quad (25)$$

and

$$\hat{f}_0(T_v) = \max\{x \in \mathbb{Z}^+ \mid \hat{g}_0(T_v; x) \neq +\infty\}. \quad (26)$$

Then $\hat{f}_{\text{out}}(T_v) = f_{\text{out}}(T_v)$ and $\hat{f}_0(T_v) = f_0(T_v)$, and hence

$$f(T_v) = \max\{\hat{f}_{\text{out}}(T_v), \hat{f}_0(T_v)\}. \quad (27)$$

We shall thus compute values $\hat{g}_{\text{out}}(T_v; x)$, $\hat{g}_{\text{in}}(T_v; x)$ and $\hat{g}_0(T_v; x)$ for all integers $x \in \mathbb{Z}$. However, one can easily observe that it suffices to compute them only for integers $x \in \mathbb{Z}_F^+$.

One can compute values $\hat{g}_{\text{out}}(T_v^0; x)$, $\hat{g}_{\text{in}}(T_v^0; x)$ and $\hat{g}_0(T_v^0; x)$ for a vertex v of T and all integers $x \in \mathbb{Z}_F^+$ in time $O(F)$ by the counterparts of Eqs. (7)–(12). If $i \geq 1$, then one can recursively compute values $\hat{g}_{\text{out}}(T_v^i; x)$, $\hat{g}_{\text{in}}(T_v^i; x)$ and $\hat{g}_0(T_v^i; x)$ for an internal vertex v of T and all integers $x \in \mathbb{Z}_F^+$ in time $O(|\mathbb{Z}_F^+|^2) = O(F^2)$ by the counterparts of Eqs. (13)–(24). Since $T_v = T_v^l$, one can compute $\hat{f}_{\text{out}}(T_v)$, $\hat{f}_0(T_v)$ and $f(T_v)$ by Eqs. (25)–(27) in time $O(F)$. Since

$T = T_r$ for the root r of T , the number of vertices in T is n and the number of edges is $n - 1$, one can compute $f(T) = f(T_r)$ in time $O(F^2n)$. This completes a proof of Theorem 2. $\mathcal{Q.E.D.}$

Note that if the supplies and demands are not always integers then $f(P)$ and $f^*(P)$ are not always integers and one cannot compute $f(T)$ in time $O(F^2n)$.

4 FPTAS for trees

Assume in this section that the supplies and demands of all vertices in a tree T are positive real numbers which are not always integers. The main result of this section is the following theorem.

Theorem 3. *There is a fully polynomial-time approximation scheme for the maximum partition problem on trees.*

In the remainder of this section, as a proof of Theorem 3, we give an algorithm to find a partition P of a tree T with $f(P) \geq (1 - \varepsilon)f(T)$ in time polynomial in n and $1/\varepsilon$ for any real number ε , $0 < \varepsilon < 1$. Thus our approximate maximum fulfillment $\bar{f}(T)$ of T is $f(P)$, and hence the error is bounded by $\varepsilon f(T)$, that is,

$$f(T) - \bar{f}(T) = f(T) - f(P) \leq \varepsilon f(T). \quad (28)$$

We first outline our algorithm and the analysis. We extend the ordinary “scaling and rounding” technique for the knapsack and maximum subset sum problems [2, 3, 6] and apply it to the maximum partition problem. For some scaling factor t , we consider the set $\{\dots, -2t, -t, 0, t, 2t, \dots\}$ as the range of functions g_{out} , g_{in} and g_0 , and find the approximate solution $\bar{f}(T)$ by using the pseudo-polynomial-time algorithm in Section 3. As we will show later in Lemma 2(c), we have

$$f(T) - \bar{f}(T) < 2nt. \quad (29)$$

Intuitively, Eq. (29) holds because the merge operation is executed no more than $2n$ times and each merge operation adds at most t to the error $f(T) - \bar{f}(T)$. Let m_d be the maximum demand, that is, $m_d = \max\{\text{dem}(v) \mid v \in V_d\}$. Taking $t = \varepsilon m_d / (2n)$ and claiming $f(T) \geq m_d$, we have Eq. (28).

We now give the details of our algorithm and the proof of its correctness. For a positive real number t , let $\mathbb{R}^t = \{\dots, -2t, -t, 0, t, 2t, \dots\}$ and $\mathbb{R}_F^{t+} = \{x \in \mathbb{R}^t \mid 0 \leq x \leq F\}$. The functions g_{out} , g_{in} and g_0 have range \mathbb{R} . In this section we define new functions \bar{g}_{out} , \bar{g}_{in} and \bar{g}_0 which have a sampled range \mathbb{R}^t and approximate g_{out} , g_{in} and g_0 , respectively. It should be noted that \bar{g}_{out} , \bar{g}_{in} and \bar{g}_0 do not always take the same value as g_{out} , g_{in} and g_0 , respectively, even for $x \in \mathbb{R}^t$. More precisely, we

- (i) recursively define and compute $\bar{g}_{\text{out}}(T_v; x)$, $\bar{g}_{\text{in}}(T_v; x)$ and $\bar{g}_0(T_v; x)$ for $x \in \mathbb{R}^t$ by the counterparts of Eqs. (7)–(24);
- (ii) define and compute values $\bar{f}_{\text{out}}(T_v)$ and $\bar{f}_0(T_v)$ by the counterparts of Eqs. (5) and (6), that is,

$$\bar{f}_{\text{out}}(T_v) = \max\{x \in \mathbb{R}_F^{t+} \mid \bar{g}_{\text{out}}(T_v; x) \neq -\infty\}$$

and

$$\bar{f}_0(T_v) = \max\{x \in \mathbb{R}_F^{t+} \mid \bar{g}_0(T_v; x) \neq +\infty\};$$

- (iii) define and compute

$$\bar{f}(T_v) = \max\{\bar{f}_{\text{out}}(T_v), \bar{f}_0(T_v)\}; \text{ and}$$

- (iv) define and compute $\bar{f}(T) = \bar{f}(T_r)$ where r is the root of T .

We will show later in Lemma 2(c) that $\bar{f}(T)$ is an approximate value of $f(T)$ satisfying Eq. (29). It should be noted that the supplies and demands are never scaled and rounded when we compute the functions \bar{g}_{out} , \bar{g}_{in} and \bar{g}_0 as above, and hence these functions take real values which are not necessarily in \mathbb{R}_F^t , although \bar{f}_{out} , \bar{f}_0 and \bar{f} take values in \mathbb{R}_F^{t+} .

The functions \bar{g}_{out} , \bar{g}_{in} and \bar{g}_0 approximate the original functions g_{out} , g_{in} and g_0 as in the following lemma, whose proof is omitted due to the page limitation. Note that $\bar{g}_{\text{out}}(T_v; x) = \bar{g}_{\text{out}}(T_v; 0)$, $\bar{g}_{\text{in}}(T_v; x) = \bar{g}_{\text{in}}(T_v; 0)$ and $\bar{g}_0(T_v; x) = \bar{g}_0(T_v; 0)$ for any negative number $x \in \mathbb{R}^t$.

Lemma 1. *Let $s(T_v^i)$ be the size of T_v^i , that is, the number of vertices and edges in T_v^i . Then the following (a), (b) and (c) hold:*

- (a)
 - (i) $\bar{g}_{\text{out}}(T_v^i; x) \leq g_{\text{out}}(T_v^i; x)$ for any number $x \in \mathbb{R}^t$;
 - (ii) $\bar{g}_{\text{out}}(T_v^i; x)$ is non-increasing; and
 - (iii) for any number $x \in \mathbb{R}$, there is an integer α such that

$$0 \leq \alpha \leq s(T_v^i) - 1$$

and

$$\bar{g}_{\text{out}}(T_v^i; \lfloor x/t \rfloor t - \alpha t) \geq g_{\text{out}}(T_v^i; x),$$

- (b)
 - (i) $\bar{g}_{\text{in}}(T_v^i; x) \geq g_{\text{in}}(T_v^i; x)$ for any number $x \in \mathbb{R}^t$;
 - (ii) $\bar{g}_{\text{in}}(T_v^i; x)$ is non-decreasing; and
 - (iii) for any number $x \in \mathbb{R}$, there is an integer β such that

$$0 \leq \beta \leq s(T_v^i)$$

and

$$\bar{g}_{\text{in}}(T_v^i; \lceil x/t \rceil t - \beta t) \leq g_{\text{in}}(T_v^i; x),$$

and

- (c) (i) $\bar{g}_0(T_v^i; x) \geq g_0(T_v^i; x)$ for any number $x \in \mathbb{R}^t$;
 (ii) $\bar{g}_0(T_v^i; x)$ is non-decreasing; and
 (iii) for any number $x \in \mathbb{R}$, there is an integer γ such that

$$0 \leq \gamma \leq s(T_v^i)$$

and

$$\bar{g}_0(T_v^i; \lceil x/t \rceil t - \gamma t) \leq g_0(T_v^i; x).$$

We then have the following lemma, whose proof is omitted from this extended abstract.

Lemma 2. *The following (a), (b) and (c) hold:*

- (a) $f_{\text{out}}(T_v^i) - s(T_v^i)t \leq \bar{f}_{\text{out}}(T_v^i)$;
 (b) $f_0(T_v^i) - s(T_v^i)t \leq \bar{f}_0(T_v^i)$;
 and
 (c) $f(T) - 2nt < \bar{f}(T) \leq f(T)$.

We are now ready to prove Theorem 3.

Proof of Theorem 3.

Let v be a demand vertex with the maximum demand $\text{dem}(v) = m_d = \max\{\text{dem}(v) \mid v \in V_d\}$ in T . Then one may assume that T has a path Q going from v to some supply vertex u such that

- (i) Q passes through only demand vertices except vertex u , and
 (ii) $\text{sup}(u)$ is no less than the sum of demands on Q ,

because, otherwise, v cannot be supplied power from any supply vertex and hence it suffices to solve the problem for each tree in a forest obtained from T by deleting v . Thus one may assume that

$$f(T) \geq m_d. \quad (30)$$

Let

$$t = \frac{\varepsilon m_d}{2n}. \quad (31)$$

Then by Lemma 2(c) and Eqs. (30) and (31) we have

$$f(T) < \bar{f}(T) + 2n \frac{\varepsilon m_d}{2n} \leq \bar{f}(T) + \varepsilon f(T),$$

and hence $(1 - \varepsilon)f(T) < \bar{f}(T) \leq f(T)$.

One can observe that the algorithm takes time

$$O\left(|\mathbb{R}_F^{t+}|^2 n\right) = O\left(\frac{n^5}{\varepsilon^2}\right),$$

because $|\mathbb{R}_F^{t+}| = \lfloor F/t \rfloor + 1$, $F \leq nm_d$ and hence by Eq. (31) we have $F/t \leq 2n^2/\varepsilon$.
Q.E.D.

5 Conclusions

In this paper, we studied the approximability of the maximum partition problem. We first showed that the maximum partition problem is MAXSNP-hard. We then gave an FPTAS for trees. It is easy to modify the FPTAS so that it actually finds a partition of a tree. The FPTAS for trees can be extended to that for series-parallel graphs and partial k -trees if there is exactly one supply vertex in a graph [4].

In the ordinary knapsack problem, each “item” is assigned a “size” and “value,” and one wishes to choose a subset of items that maximizes the sum of values of items such that their total size does not exceed the size of a bag [3, 6]. Consider a slightly modified version of the maximum partition problem on graphs in which each demand vertex is assigned not only a demand but also a “value,” and one wishes to find a partition which maximizes the sum of values of all demand vertices in components with supply vertices. This problem is indeed a generalization of the ordinary knapsack problem, and can be solved for trees using techniques similar to those for the maximum partition problem. Note that the standard approximation methods for the knapsack problem in [3, 6] cannot be applied to the modified maximum partition problem.

References

1. N. G. Boulaxis and M. P. Papadopoulos, Optimal feeder routing in distribution system planning using dynamic programming technique and GIS facilities, *IEEE Trans. on Power Delivery*, Vol. 17, pp. 242–247, 2002.
2. M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, CA, 1979.
3. O. H. Ibarra and C. E. Kim, Fast approximation algorithms for the knapsack and sum of subset problems, *J. Asso. Comput. Mach.*, Vol. 22, pp. 463–468, 1975.
4. T. Ito, E. D. Demaine, X. Zhou and T. Nishizeki, Approximability of partitioning graphs with supply and demand, in *Proc. of the 17th Annual International Symposium on Algorithms and Computation (ISAAC2006)*, *Lecture Notes in Computer Science*, Vol. 4288, pp. 121–130, 2006.
5. T. Ito, X. Zhou and T. Nishizeki, Partitioning trees of supply and demand, *International J. of Foundations of Computer Science*, Vol. 16, pp. 803–827, 2005.
6. P. N. Klein and N. E. Young, Approximation algorithms for NP-hard optimization problems, Chap. 34 in (Ed. M. J. Atallah) *Algorithms and Theory of Computation Handbook*, CRC Press, Boca Raton, Florida, 1999.

7. A. B. Morton and I. M. Y. Mareels, An efficient brute-force solution to the network reconfiguration problem, *IEEE Trans. on Power Delivery*, Vol. 15, pp. 996–1000, 2000.
8. C. H. Papadimitriou, *Computational Complexity*, Addison-Wesley, 1994.
9. C. H. Papadimitriou and M. Yannakakis, Optimization, approximation, and complexity classes, *J. Computer and System Sciences*, Vol. 43, pp. 425–440, 1991.