The Tenth International Symposium on Operations Research and Its Applications (ISORA 2011)
Dunhuang, China, August 28–31, 2011

Partitioning Trees with Supply, Demand and Edge-Capacity

Masaki Kawabata[1,*] Takao Nishizeki[1,†]

[1]School of Science and Technology
Kwansei Gakuin University
2-1 Gakuen, Sanda, 669-1337 Japan.

Abstract Let T be a tree. Each vertex of T is either a supply vertex or a demand vertex, and is assigned a positive number called the supply or demand. Each demand vertex can receive "power" from exactly one supply vertex through edges in T. Each edge is assigned a positive number called the capacity. One wishes to partition T into subtrees by deleting edges from T so that each subtree contains exactly one supply vertex whose supply is no less than the sum of all demands in the subtree and the flow through each edge is no more than capacity of the edge. The "partition problem" is a decision problem to ask whether T has such a partition. The "maximum partition problem" is an optimization version of the partition problem. In this paper, we give three algorithms for the problems. First is a linear-time algorithm for the partition problem. Second is a pseudo-polynomial-time algorithm for the maximum partition problem. Third is a fully polynomial-time approximation scheme (FPTAS) for the maximum partition problem.

Keywords tree; partition problem; maximum partition problem; supply; demand; edge-capacity; algorithm; approximation; FPTAS; pseudo polynomial-time

1 Introduction

The partition problem and the maximum partition problem for graphs have some applications in the power supply problem for power delivery networks and VLSI circuits [1, 4, 5, 6]. Ito *et al*. deal with the problems for trees having supply and demand vertices [3], but do not take account of edge-capacities. This paper deals with the problems with edge-capacities.

Let T be a tree. Each vertex of T is either a *supply vertex* or a *demand vertex*, and is assigned a positive number called the *supply* or *demand*. Each demand vertex can receive "power" from exactly one supply vertex through edges in T. Each edge is assigned a positive number, called the *capacity*. We thus wish to partition T into subtrees by deleting edges from T so that

(a) each subtree contains exactly one supply vertex whose supply is no less than the sum of all demands in the subtree; and

*masaki.kawabata7671@kwansei.ac.jp
†nishi@kwansei.ac.jp

(b) the flow through each edge is no more than the capacity of the edge.

Not every tree has such a partition. The *partition problem* is a decision problem to ask whether T has such a partition. If T has no such partition, then we wish to partition T into subtrees so that

(a) each subtree contains at most one supply vertex; and
(b) if a subtree contains a supply vertex, then the supply is no less than the sum of all demands in the subtree and the flow through each edge is no more than the capacity.

The *maximum partition problem* is an optimization problem to find one of these partitions that maximizes the "fulfillment," that is, the sum of demands in all subtrees with supply vertices. Clearly, the maximum partition problem is a generalization of the partition problem.

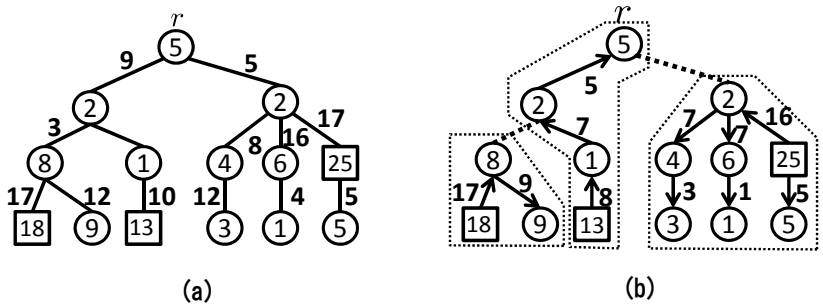

Figure 1: (a) Tree T, and (b) desired partition.

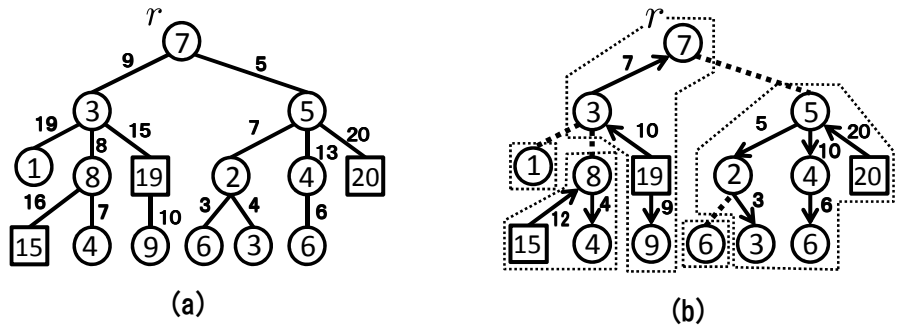

Figure 2: (a) Tree T, and (b) maximum partition.

Figure 1(a) depicts a tree T; each supply vertex is drawn by a rectangle and each demand vertex by a circle, the supply or demand is written inside; and the capacity is attached to each edge. The tree T has a desired partition as illustrated in Fig. 1(b); the

deleted edges are drawn by thick dotted lines; each subtree is surrounded by a thin dotted line; the flow value is attached to each edge, and the flow direction is indicated by an arrow. On the other hand, the tree T in Fig. 2(a) has no desired partition. Figure 2(b) illustrates the maximum partition of T, whose fulfillment is (7+3+9)+(4+8)+(5+2+4+3+6)=51.

These problems can be defined for graphs in general. However, the partition problem is NP-complete even for complete bipartite graphs [3]. On the other hand, the maximum partition problem is NP-hard even for trees [3]. Moreover, the maximum partition problem is MAXSNP-hard for graphs [2].

In this paper we give three algorithms for a tree T by extending the algorithms for the problems without edge-capacities in [3]. First is a linear-time algorithm for the partition problem. Second is a pseudo-polynomial-time algorithm to solve the maximum partition problem in time $O(F^2n)$, where n is the number of vertices in T and F is the sum of all demands in T. Third is a fully polynomial-time approximation scheme (FPTAS) for the maximum partition problem. (The detail of the FPTAS is omitted in this extended abstract.)

2 Linear Algorithm for the Partition Problem

In this section, we have the following theorem on the partition problem.

Theorem 1. *The partition problem can be solved for trees in linear time.*

In the remainder of this section, as a proof of Theorem 1, we give an algorithm to solve the partition problem for trees in linear time. Let T be a given tree. We choose an arbitrary vertex r as the root of T, and regard T as a "rooted" tree. We denote by $s(v)$ the supply of a supply vertex v, and by $d(v)$ the demand of a demand vertex v. We denote by $c(u,v)$ the capacity of an edge (u,v) joining vertices u and v. The algorithm repeatedly executes the following Steps 1–4 until exactly one vertex remains in T.

(Step 1) Choose any internal vertex u of T such that all children are leaves, and let D be the set of all children of u that are demand vertices.

(Step 2) If there is a demand vertex $v \in D$ such that $c(u,v) < d(v)$, then output "no desired partition."

(Step 3) If u is a demand vertex, then let $d(u) := d(u) + \sum_{v \in D} d(v)$ and delete all vertices in D from T. If the demand vertex u has a (supply) child, then execute the following (a)–(c).

(a) Let $s(v) := \min\{s(u), c(u,v)\}$ for each child v of u. Let w be the child whose supply is maximum among all children of u. Delete all the other children of u from T.

(b) If $d(u) \leq s(w)$, then delete w from T, regard u as a supply vertex with $s(u) := s(w) - d(u)$, and let T be the resulting tree.

(c) If $d(u) > s(w)$, then delete the supply vertex w from T and let T be the resulting tree.

(Step 4) If u is a supply vertex, then delete all supply vertices that are children of u and execute the following (a) and (b).

(a) If $s(u) \geq \sum_{v \in D} d(v)$, then delete all the children of u from T, let $s(u) := s(u) - \sum_{v \in D} d(v)$, and let T be the resulting tree.

(b) If $s(u) < \sum_{v \in D} d(v)$, then output "no desired partition."

When the algorithm terminates, T consists of a single vertex. If the vertex is a supply vertex, then there is a desired partition. Otherwise, there is no desired partition.

One can easily observe that the algorithm correctly solves the partition problem in linear time, and that the algorithm can be easily modified so that it actually finds a partition of a tree if there is.

3 Maximum Partition Problem

The main result of this section is the following theorem.

Theorem 2. *The maximum partition problem can be solved for a tree T in time $O(F^2n)$ if the demands and supplies are integers and $F = \min\{\sum_{v\in V_s} s(v), \sum_{v\in V_d} d(v)\}$, where V_s is the set of all supply vertices and V_d the set of all demand vertices in T.*

In the remainder of this section, as a proof of Theorem 2, we give an algorithm to solve the maximum partition problem in time $O(F^2n)$. In this section and the succeeding section, a *partition* P of a tree T is to partition T into subtrees by deleting edges from T so that

(a) each subtree contains at most one supply vertex; and
(b) if a subtree contains a supply vertex, then the supply is no less than the sum of all demands in the subtree and the flow through each edge is no more than the capacity.

The *fulfillment $f(P)$ of a partition P* is the sum of demands in all subtrees with supply vertices. The *maximum partition problem* is to find a partition of T with the *maximum fulfillment*, which is called the *maximum fulfillment $f(T)$ of T*. Clearly $f(T) \le F$.

One may assume that T is a rooted tree with root r. For a vertex v of T, we denote by T_v the maximum subtree of T rooted at v. Let $\mathbb{R}$ be the set of all real numbers. For a non-negative real number z, we denote by $\mathbb{R}_z$ the set of all real numbers $x \le z$, and denote by $\mathbb{R}_z^+$ the set of all real numbers x, $0 \le x \le z$. Our idea is to introduce two new functions $g_{\rm out}^{\rm par}(T_v;x)$ and $g_{\rm in}^{\rm par}(T_v;x)$, $x \in \mathbb{R}$, in addition to the three functions $g_{\rm out}(T_v;x)$, $g_{\rm in}(T_v;x)$ and $g_0(T_v;x)$ considered in [3]. We compute the five functions for each vertex v from leaves to the root of T by dynamic programming.

Let v be a vertex of T, let $v_1, v_2, \cdots, v_l$ be the children of v, and let $e_i, 1 \le i \le l$, be the edge joining v and v_i. Let $T_{v_i}, 1 \le i \le l$, be the maximum subtree of T rooted at v_i. We denote by T_v^i the subtree of T_v which consists of vertex v, edges $e_1, e_2, \cdots, e_i$ and subtrees $T_{v_1}, T_{v_2}, \cdots, T_{v_i}$. Clearly $T_v = T_v^l$. 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 $R(P)$ be the set of all vertices in the subtree in P containing the root v of T_v. We now define j-out, j-out-parent, j-in, j-in-parent and isolated partitions, as follows.

(a) A partition P of T_v is called a *j-out partition*, $j \in \mathbb{R}_F^+$, if $R(P)$ contains a supply vertex w and $s(w) \ge j + \sum_{u\in R(P)-\{w\}} d(u)$. A j-out partition of T_v corresponds to a partition of T in which v is supplied power from a supply vertex in T_v.
(b) A j-out partition P of T_v is called a *j-out-parent partition* if $j \le c(v, v^{\rm par})$, where $v^{\rm par}$ is the parent of v.

(c) A partition P of T_v is called a *j-in partition* if the following (i) and (ii) hold:
   (i) $R(P)$ contains no supply vertex; and
   (ii) add to T_v a virtual supply vertex v_s with $s(v_s) = j$ together with a virtual edge (v_s, v) with $c(v_s, v) = j$, then in the resulting tree T_v^* all (demand) vertices in $R(P)$ can be supplied power from v_s, that is, $\sum_{u \in R(P)} d(u) \leq j$ and the flow through each edge is no more than the capacity.

   A j-in partition of T_v corresponds to a partition of T in which v is supplied power from a supply vertex outside T_v. For a j-in partition P, let $f^*(P) = f(P) + \sum_{u \in R(P)} d(u)$. Clearly, a j-in partition P of T_v induces a partition P^* of T_v^* such that $f(P^*) = f^*(P)$.
(d) A j-in partition P of T_v is called a *j-in-parent partition* if $j \leq c(v, v^{\text{par}})$.
(e) A partition P of T_v is called an *isolated partition* if $R(P)$ consists of a single demand vertex v. An isolated partition of T_v corresponds to a partition P' of T in which v is not supplied power from any supply vertex in T.

We now give a formal definition of the five functions $g_{\text{out}}(T_v;x)$, $g_{\text{out}}^{\text{par}}(T_v;x)$, $g_{\text{in}}(T_v;x)$, $g_{\text{in}}^{\text{par}}(T_v;x)$, and $g_0(T_v;x)$, $x \in \mathbb{R}$:

$$g_{\text{out}}(T_v;x) = \max\{j \in \mathbb{R}_F^+ | T_v \text{ has a } j\text{-out partition } P \text{ such that } f(P) \geq x\} \tag{1}$$

where $g_{\text{out}}(T_v;x) = -\infty$ if T_v has no j-out partition P with $f(P) \geq x$ for any number $j \in \mathbb{R}_F^+$;

$$g_{\text{out}}^{\text{par}}(T_v;x) = \min\{g_{\text{out}}(T_v;x), c(v, v^{\text{par}})\}; \tag{2}$$

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

where $g_{\text{in}}(T_v;x) = +\infty$ if T_v has no j-in partition with $f^*(P) \geq x$ for any number $j \in \mathbb{R}_F^+$;

$$g_{\text{in}}^{\text{par}}(T_v;x) = \begin{cases} g_{\text{in}}(T_v;x) & \text{if } g_{\text{in}}(T_v;x) \leq c(v, v^{\text{par}}); \\ +\infty & \text{otherwise;} \end{cases} \tag{4}$$

and

$$g_0(T_v;x) = \begin{cases} 0 & \text{if } T_v \text{ has an isolated partition } P \text{ such that } f(P) \geq x; \\ +\infty & \text{otherwise.} \end{cases} \tag{5}$$

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

$$f_{\text{out}}(T_v) = \max\{x \in \mathbb{R} | g_{\text{out}}(T_v;x) \neq -\infty\}. \tag{6}$$

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

$$f_0(T_v) = \max\{x \in \mathbb{R} | g_0(T_v;x) = 0\}. \tag{7}$$

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

$$f(T_v) = \max\{f_{\text{out}}(T_v), f_0(T_v)\}, \tag{8}$$

and hence

$$f(T) = f(T_r) = \max\{f_{\text{out}}(T_r), f_0(T_r)\} \tag{9}$$

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 $j \in \mathbb{R}_F^+$.

Our algorithm computes $g_{\text{out}}(T_v;x)$, $g_{\text{out}}^{\text{par}}(T_v;x)$, $g_{\text{in}}(T_v;x)$, $g_{\text{in}}^{\text{par}}(T_v;x)$, and $g_0(T_v;x)$ for each vertex v of T from leaves to the root r of T by means of dynamic programming. One can compute the fulfillment $f(T)$ of T from $g_{\text{out}}(T_r;x)$ and $g_0(T_r;x)$ by Eqs. (6)–(8).

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

$$g_{\text{out}}(T_v^0;x) = -\infty, \tag{10}$$

$$g_{\text{out}}^{\text{par}}(T_v^0;x) = -\infty, \tag{11}$$

$$g_{\text{in}}(T_v^0;x) = \begin{cases} d(v) & \text{if } x \le d(v); \\ +\infty & \text{otherwise,} \end{cases} \tag{12}$$

$$g_{\text{in}}^{\text{par}}(T_v^0;x) = \begin{cases} g_{\text{in}}(T_v^0;x) & \text{if } g_{\text{in}}(T_v^0;x) \le c(v,v^{\text{par}}); \\ +\infty & \text{otherwise,} \end{cases} \tag{13}$$

and

$$g_0(T_v^0;x) = \begin{cases} 0 & \text{if } x \le 0; \\ +\infty & \text{otherwise.} \end{cases} \tag{14}$$

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

$$g_{\text{out}}(T_v^0;x) = \begin{cases} s(v) & \text{if } x \le 0; \\ -\infty & \text{otherwise,} \end{cases} \tag{15}$$

$$g_{\text{out}}^{\text{par}}(T_v^0;x) = \min\{g_{\text{out}}(T_v^0;x), c(v,v^{\text{par}})\}, \tag{16}$$

$$g_{\text{in}}(T_v^0;x) = +\infty, \tag{17}$$

$$g_{\text{in}}^{\text{par}}(T_v^0;x) = +\infty, \tag{18}$$

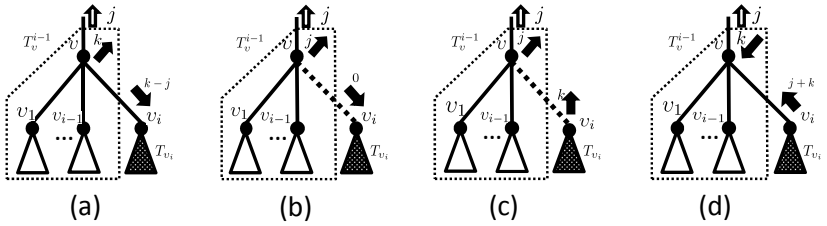

Figure 3: Flows in a j-out partition of T_v^i.

and

$$g_0(T_v^0;x) = +\infty. \tag{19}$$

We next compute $g_{\mathrm{out}}(T_v^i;x)$, $g_{\mathrm{out}}^{\mathrm{par}}(T_v^i;x)$, $g_{\mathrm{in}}(T_v^i;x)$, $g_{\mathrm{in}}^{\mathrm{par}}(T_v^i;x)$ and $g_0(T_v^i;x)$, $1 \le i \le l$, for each internal vertex v of T from the counterparts for T_v^{i-1} and T_{v_i}, where l is the number of the children of v. 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. 3 where T_v^{i-1} is surrounded by a thin dotted line.

We explain only how to compute $g_{\mathrm{out}}(T_v^i;x)$ and $g_{\mathrm{out}}^{\mathrm{par}}(T_v^i;x)$, because one can similarly compute $g_{\mathrm{in}}(T_v^i;x)$, $g_{\mathrm{in}}^{\mathrm{par}}(T_v^i;x)$ and $g_0(T_v^i;x)$. Let P be a j-out partition of T_v^i such that $f(P) \ge x$ and $j = g_{\mathrm{out}}(T_v^i;x) \ne -\infty$. Then there are the following four Cases (a)–(d). (In Figs. 3(a)–(d) an arrow represents the direction of power supply for Cases (a)–(d), respectively.)

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

In this case, 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-parent partition P_2 of T_{v_i} such that $f(P_1) \ge y$ and $f^*(P_2) \ge x-y$ for some numbers $k(\ge j)$ and $y \in \mathbb{R}_x^+$. (See Fig. 3 (a).) Intuitively, x and y are the fulfillments of T_v^i and T_v^{i-1}, respectively. One may assume that $k = g_{\mathrm{out}}(T_v^{i-1};y)$ and $k-j = g_{\mathrm{in}}^{\mathrm{par}}(T_{v_i};x-y)$. Since $g_{\mathrm{out}}(T_v^i;x) = j = k-(k-j)$, we define $g_{\mathrm{out}}^{\mathrm{a}}(T_v^i;x,y)$ as follows:

$$g_{\mathrm{out}}^{\mathrm{a}}(T_v^i;x,y) = g_{\mathrm{out}}(T_v^{i-1};y) - g_{\mathrm{in}}^{\mathrm{par}}(T_{v_i};x-y). \tag{20}$$

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 $f(P_1) \ge y$ and $f(P_2) \ge x-y$ for some $y \in \mathbb{R}_x^+$. (See Fig. 3 (b).) Then $j = j-0$, and hence let

$$g_{\mathrm{out}}^{\mathrm{b}}(T_v^i;x,y) = g_{\mathrm{out}}(T_v^{i-1};y) - g_0(T_{v_i};x-y). \tag{21}$$

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, 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 $k \in \mathbb{R}_F^+$ and $y \in \mathbb{R}_x^+$. (See Fig. 3 (c).) Then let

$$g_{\text{out}}^{\text{c}}(T_v^i;x,y) = \begin{cases} g_{\text{out}}(T_v^{i-1};y) & \text{if } g_{\text{out}}(T_{v_i};x-y) \geq 0; \\ -\infty & \text{if } g_{\text{out}}(T_{v_i};x-y) = -\infty. \end{cases} \tag{22}$$

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

In this case, P can be obtained by merging a k-in partition P_1 of T_v^{i-1} and a $(j+k)$-out-parent partition P_2 of T_{v_i} such that $f^*(P_1) \geq y$ and $f(P_2) \geq x-y$ for some number $k \in \mathbb{R}_F^+$ and $y \in \mathbb{R}_x^+$. (See Fig. 3 (d).) Then $j = (j+k)-k$, and hence let

$$g_{\text{out}}^{\text{d}}(T_v^i;x,y) = g_{\text{out}}^{\text{par}}(T_{v_i};x-y) - g_{\text{in}}(T_v^{i-1};y). \tag{23}$$

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

$$\begin{aligned} g_{\text{out}}(T_v^i;x) = \max \{\, & g_{\text{out}}^{\text{a}}(T_v^i;x,y), g_{\text{out}}^{\text{b}}(T_v^i;x,y), \\ & g_{\text{out}}^{\text{c}}(T_v^i;x,y), g_{\text{out}}^{\text{d}}(T_v^i;x,y) | y \in \mathbb{R}_x^+ \}; \end{aligned} \tag{24}$$

and

$$g_{\text{out}}^{\text{par}}(T_v^i;x) = \min\{g_{\text{out}}(T_v^i;x), c(v,v^{\text{par}})\}. \tag{25}$$

Suppose now that all the supplies and demands are integers. Then $f(P)$ and $f^*(P)$ are integer for any partition of T_v. Therefore, it suffices to compute values $g_{\text{out}}(T_v;x)$, $g_{\text{out}}^{\text{par}}(T_v;x)$, $g_{\text{in}}(T_v;x)$, $g_{\text{in}}^{\text{par}}(T_v;x)$ and $g_0(T_v;x)$ only for integers x, $0 \leq x \leq F$. One can compute $g_{\text{out}}(T_v^0;x)$, $g_{\text{out}}^{\text{par}}(T_v^0;x)$, $g_{\text{in}}(T_v^0;x)$, $g_{\text{in}}^{\text{par}}(T_v^0;x)$ and $g_0(T_v^0;x)$ for a vertex v of T and all integers x, $0 \leq x \leq F$, in time $O(F)$ by Eqs. (10)–(19). One can recursively compute $g_{\text{out}}(T_v^i;x)$, $g_{\text{out}}^{\text{par}}(T_v^i;x)$, $g_{\text{in}}(T_v^i;x)$, $g_{\text{in}}^{\text{par}}(T_v^i;x)$ and $g_0(T_v^i;x)$ for an internal vertex v of T and all integers x, $0 \leq x \leq F$, in time $O(F^2)$ by Eqs. (20)–(25) for g_{out} and by the similar equations for g_{in} and g_0. Since $T_v = T_v^l$, one can compute $f_{\text{out}}(T_v)$, $f_0(T_v)$ and $f(T_v)$ in time $O(F)$ by Eqs. (6) – (8). The number of vertices in T is n, and the number of edges is $n-1$. Therefore, one can compute $f(T) = f(T_r)$ in time $O(F^2 n)$. This completes a proof of Theorem 2.

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, No. 1(2002), pp. 242-247.

[2] T. Ito, E. D. Demaine, X. Zhou and T. Nishizeki, "Approximability of partitioning graphs with supply and demand," *Journal of Discrete Algorithms*, 6(2008), pp. 627-650.

[3] T. Ito, X. Zhou and T. Nishizeki, "Partitioning trees of supply and demand," *IJFCS*, Volume:16, Issue:4(2005), pp. 803-827.

[4] T. Lengauer, "Combinatorial Algorithms for Integrated Circuit Layout," *John Wiley and Sons*, Chichester(1990).

[5] 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, No. 3(2000), pp. 996-1000.

[6] J-H. Teng and C-N. Lu, "Feeder-switch relocation for customer interruption cost minimization," *IEEE Trans. on Power Delivery*, Vol. 17, No. 1(2002), pp. 254-259.