MATHEMATICAL ENGINEERING TECHNICAL REPORTS

A Network Flow Approach to Cost Allocation for Rooted Trees

Satoru Iwata Nozomu Zuiki

METR 2003–14 April 2003

DEPARTMENT OF MATHEMATICAL INFORMATICS
GRADUATE SCHOOL OF INFORMATION SCIENCE AND TECHNOLOGY
THE UNIVERSITY OF TOKYO
BUNKYO-KU, TOKYO 113-8656, JAPAN

WWW page: http://www.i.u-tokyo.ac.jp/mi/mi-e.htm

A Network Flow Approach to Cost Allocation for Rooted Trees

Satoru Iwata * Nozomu Zuiki

April 2003

Abstract

In the game theory approach to cost allocation, the main computational issue is an algorithm for finding solutions such as the Shapley value and the nucleolus. In this paper, we consider the problem of allocating construction cost of a tree network that connects the supply source at the root to the users in the leaves. The Shapley value of this game can be computed in $O(n)$ time, where n is the number of leaves. We show that the core of the game can be expressed in terms of network flows. Based on this observation, we present $O(n \log n)$ algorithms for computing the nucleolus and the egalitarian allocation.

*Department of Mathematical Informatics, Graduate School of Information Science and Technology, University of Tokyo, Tokyo 113-8656, Japan.

1 Introduction.

Suppose we plan to construct a pipeline network to supply gas from a source to each house in a village. What is a reasonable way for the residents to share the construction cost? This situation can be naturally formulated as a cooperative game.

Let $T = (V, E)$ be a rooted directed tree that models the pipeline network. The root r corresponds to the source, and the leaves $N = \{1, \cdots, n\}$ to the residents. For each arc $e \in E$, we denote by $c(e)$ the cost to construct the corresponding part of the network. For a nonempty subset $S \subseteq N$, let E_S be the set of arcs in the paths from the root r to the leaves in S. Namely, E_S is the minimum arc set that is required to construct so that the residents in S should be able to obtain gas from the source. The cost to construct E_S is given by

$$v(S) = \sum_{e \in E_S} c(e). \tag{1}$$

If S is the set of residents who want to construct the pipeline, the residents in S have to pay $v(S)$ in total.

Regarding the residents N as players and v as a characteristic function, we consider a cooperative game (N, v).The game (N, v) is a convex game, whose characteristic function is submodular, i.e.,

$$v(S) - v(S - \{i\}) \geq v(T) - v(T - \{i\}), \quad \forall i \in S \subset T \subseteq N. \tag{2}$$

Cost allocation problems on trees have been considered by several authors. Megiddo [8] studied the case in which each resident exists in each vertex except for the root in a tree. He described an $O(n)$ algorithm for computing the Shapley value and an $O(n^3)$ algorithm for finding the nucleolus, where n is the number of vertices of the tree. Galil [4] improved the latter $O(n^3)$ bound to $O(n \log n)$ by using efficient mergeable heaps.

Given an instance of Megiddo's model, attach a new arc with zero construction cost to each internal node, and move the resident to the end of the new arc. The resulting equivalent instance has each resident in each leave. Thus Megiddo's model is a special case of our framework.

Granot, Mashler, Owen, and Zhu [5] considered another extension of Megiddo's model and called it a standard tree game. The standard tree game allows each vertex except for the root to have an arbitrary number of residents. They presented an algorithm for computing the nucleolus.

Apparently, the standard tree games include our framework. On the other hand, any standard tree game can be transformed to our game by attaching to each node the same number of arcs as the residents therein and moving them to the distinct end vertices. Thus our framework is as wide as the standard tree games. Since our model

is simpler to describe, it seems more suitable as a canonical model of these types of problems.

In this paper, we show that the core of our game can be expressed in terms of network flows. Based on this observation, we devise an algorithm for computing the nucleolus in $O(n \log n)$ time with the aid of efficient mergeable heaps. Moreover, we show that the egalitarian allocation can be found in $O(n \log n)$ time by a quadratic optimization algorithm of Hochbaum and Hong [6].

The concept of egalitarian allocation was proposed by Dutta and Ray [2]. In a convex game, the core forms a base polyhedron and the egalitarian allocation is the lexicographically optimal base introduced by Fujishige [3]. The lexicographically optimal base generalizes the lexicographically optimal flow in a network with multiple sources and sinks investigated by Megiddo [7]. Fujishige [3] showed that the lexicographically optimal base minimizes among all bases a separable convex quadratic objective function. Hochbaum and Hong [6] presented an efficient algorithm for solving this problem when the base polyhedron comes from the flows in tree network. We apply this algorithm to the egalitarian allocation of our game.

2 The Shapley value.

The Shapley value of a cooperative game (N, v) is defined by

$$y_i = \sum_{S: i \in S \subset N} \frac{|S|!(n - |S| - 1)!}{n!} [v(S) - v(S \backslash \{i\})]$$

In our game with $T = (V, E)$, let P_i denote the set of arcs in the path from r to $i \in N$ and $A(e)$ the set of leaves that use $e \in E$. Namely, $A(e) = \{i \mid e \in P_i\}$ for each $e \in E$.

The Shapley value of our game can be characterized as an allocation such that the cost of an arc is equally paid by its users. This is a straightforward generalization of the result of Megiddo [8]. It suggests a simple linear time algorithm for computing the Shapley value.

Theorem 1 *The Shapley value y of our game on the tree $T = (V, E)$ is given by*

$$y_i = \sum_{e \in P_i} \frac{c(e)}{|A(e)|}.$$

Proof. Suppose the users join the construction plan one by one in a random order and each user is charged with the additional cost when he joins. The Shapley value is the expectation of the cost that each user has to pay in such a situation.

For $e \in P_i$, if resident i is at the head in the order among the users in $A(e)$, then i must be charged with $c(e)$. The probability of this event is $1/|A(e)|$. Therefore, the expectation of the cost that i has to pay is the sum of $c(e)/|A(e)|$ for $e \in P_i$. Thus we obtain the formula for the Shapley value. ■

3 The core.

In this section, we show that the core of the game (N, v) can be expressed in terms of network flows.

For any vector $x \in \mathbf{R}^N$ and $S \subseteq N$, we denote $x(S) = \sum_{i \in S} x_i$. The *core* is the set of allocation defined by

$$C(v) = \{x \in \mathbf{R}^N \mid x(S) \leq v(S), \forall S \subset N, x(N) = v(N)\}.$$

Let $u : 2^N \to \mathbf{R}$ be defined by $u(S) = v(N) - v(N \backslash S)$. Then we have

$$C(v) = \{x \in \mathbf{R}^N \mid x(N) = u(N), x(S) \geq u(S), \forall S \subset N\}. \tag{3}$$

If $N \backslash S$ construct the arcs to obtain the gas, they must be charged with $v(N \backslash S)$. Then $u(S)$ is the rest of the total cost. Thus, $u(S)$ is the minimum cost S must be charged with.

Though the core is expressed in (3) by $2^n - 1$ inequality constraints, many of them are redundant. In fact, only $|E| = n - 1$ constraints that correspond to the arcs are needed to describe the core. This will be shown in Theorem 2.

For $S \subseteq N$, let F_S denote the set of arcs that are used only by S. Then F_S is a forest that consists of rooted directed subtrees. Let D_S be the roots of those subtrees. For each $p \in V$, we denote by T_p the subtree of T rooted at p containing every vertex below p. We also denote by V_p and N_p, respectively, the set of vertices and leaves of T_p. Then we have the following lemma.

Lemma 1 For an arbitrary $S \subset N$, we have

$$u(S) = \sum_{p \in D_S} u(N_p).$$

Proof. By the definition of u, we have

$$u(S) = \sum_{e \in F_S} c(e),$$

which together with $S = \bigcup_{p \in D_S} N_p$ implies

$$\sum_{p \in D_S} u(N_p) = \sum_{p \in D_S} \sum_{e \in F_{N_p}} c(e).$$

Since the arc sets F_{N_p} for $p \in D_S$ are disjoint, we have

$$\sum_{p \in D_S} u(N_p) = \sum_{e \in F_S} c(e).$$

Thus we obtain

$$u(S) = \sum_{p \in D_S} u(N_p).$$

■

For each $p \in V \backslash \{r\}$, we denote by e_p the arc that enters p in T. Namely, we have $N_p = A(e_p)$.

Theorem 2 *In the game (N, v), the core $C(v)$ is given by*

$$C(v) = \{x \in \mathbf{R}^N \mid x(N) = v(N),\ x(A(e)) \geq u(A(e)),\ \forall e \in E\}. \tag{4}$$

Proof. Let $C'(v)$ denote the polyhedron described in the right-hand side of (4). Since $C(v)$ has all inequalities in $C'(v)$, it is clear that $C'(v) \subseteq C(v)$. To prove $C(v) \subseteq C'(v)$, we derive $x(S) - u(S) \geq 0$ for an arbitrary $S \subseteq N$ from the inequality constraints in (4). If $x \in C'(v)$, we have

$$x(A(e_p)) \geq u(A(e_p))$$

for each $p \in D_S$. Summing up the both sides of these inequalities over D_S, we have

$$\sum_{p \in D_S} x(A(e_p)) \geq \sum_{p \in D_S} u(A(e_p)).$$

Because of $S = \bigcup_{p \in D_S} A(e_p)$ and Lemma 1, we have

$$x(S) \geq u(S).$$

This leads to $C'(v) \supseteq C(v)$. Thus we obtain $C(v) = C'(v)$. ■

We now consider how to express the core in terms of network flows. A vertex $t \in V$ is called a child of $s \in V$, if there is an arc from s to t. For each $s \in V$, let $\Gamma(s)$ denote the set of children of s. A flow is a function $f : E \to \mathbf{R}$ that satisfies the conservation law $f(e_p) = \sum_{q \in \Gamma(p)} f(e_q)$ for each internal vertex p. The value of a flow f is defined by $\mathrm{val}\, f = \sum_{q \in \Gamma(r)} f(e_q)$. Let $\ell : E \to \mathbf{R}$ be the lower bound defined by $l(e) = u(A(e))$. A flow f is called *feasible* if it satisfies $f(e) \geq \ell(e)$ for each arc $e \in E$ and $\mathrm{val}\, f = u(N)$. Then we have the following theorem.

Theorem 3 *The core of the game (N, v) can be expressed by*

$$C(v) = \{x \in \mathbf{R}^N \mid x_i = f(e_i), \forall i \in N,\ f\text{: feasible flow}\} \tag{5}$$

Theorem 3 shows that the core in the game (N, v) can be expressed by the set of feasible flows in T.

4 The nucleolus.

In this section, we present an efficient algorithm to compute the nucleolus in the game (N, v) based on the connection between the core and the feasible flows. The resulting algorithm is quite close to that of Megiddo when applied to the special case discussed in [8].

For the definition of the nucleolus, first we define $h(S, x) \equiv x(S) - u(S)$ for each proper nonempty subset S of N. Let $\eta(x)$ be the $(2^n - 2)$-tuple of the numbers $h(S, x)$ arranged in order of increasing magnitude. Then the nucleolus $z = (z_1, \cdots, z_n)$ is defined to be the unique vector that lexicographically maximizes $\eta(z)$ in the core.

Similarly to Lemma 1, we have the following.

Lemma 2 *For an arbitrary $S \subseteq N$, we have*

$$h(S, x) = \sum_{p \in D_S} h(A(e_p), x).$$

Proof. By the definition of h, we have $h(A(e_p), x) = x(A(e_p)) - u(A(e_p))$ for $p \in D_S$. Summing up the both sides of these equations for $p \in D_S$, we have

$$\sum_{p \in D_S} h(A(e_p), x) = \sum_{e \in D_S} x(A(e_p)) - \sum_{p \in D_S} u(A(e_p)).$$

Then it follows from Lemma 1 that

$$h(S, x) = \sum_{p \in D_S} h(A(e_p), x)$$

holds. ■

Lemma 2 implies that the nucleolus is a vector $x \in C(v)$ that lexicographically maximizes $h(A(e), x)$ for $e \in E$. By the definitions of $f(e)$ and $\ell(e)$, we have $h(A(e), x) = f(e) - \ell(e)$. Therefore, to compute the nucleolus, we should lexicographically maximize the gap between the feasible flow and the lower bound.

For a feasible flow f, let $\eta(f)$ denote the numbers $f(e) - \ell(e)$ arranged in the increasing order of magnitude. We call a feasible flow f^* the ν-flow if it lexicographically maximizes $\eta(f^*)$.

We now define functions g_s for $s \in V \backslash \{r\}$ and $\tilde{g}_s$ for $s \in V$ recursively by

$$\begin{aligned} g_s(\epsilon) &:= \max\{\ell(e_s) + \epsilon, \tilde{g}_s(\epsilon)\}, \\ \tilde{g}_s(\epsilon) &:= \sum_{t \in \Gamma(s)} g_t(\epsilon). \end{aligned}$$

Note that, for $s \in N$, we have $\Gamma(s) = \emptyset$, and hence $\tilde{g}_s(\epsilon) = 0$ holds. Then $g_s(\epsilon)$ is the lower bound on the flow through e_s when ℓ is uniformly increased by ϵ. Furthermore,

$\tilde{g}_r(\epsilon)$ is the lower bound on the value of feasible flows in the same situation. The functions g_s and $\tilde{g}_s$ are piece-wise linear, monotone nondecreasing, and convex. Let ϵ_s be the maximum value of ϵ such that $g_s(\epsilon) = \ell(e_s) + \epsilon$. Then $g_s(\epsilon) = \ell(e) + \epsilon$ if $\epsilon \leq \epsilon_s$, and $g_s(\epsilon) = \tilde{g}_s(\epsilon)$ otherwise.

Let ϵ^* be the maximum value that satisfies $\tilde{g}_r(\epsilon^*) = u(N)$. Then ϵ^* is the maximum gap between a feasible flow and its lower bound. Let $T^* = (W, F)$ be the unique maximal subtree of T such that $\epsilon_s < \epsilon^*$ for $s \in W \backslash \{r\}$, and let R be the set of vertices in $V \backslash W$ adjacent to W. Then we have $\tilde{g}_r(\epsilon^*) = \sum_{s \in R} g_s(\epsilon^*) = \epsilon^* |R| + \sum_{s \in R} \ell(e_s)$. On the other hand, there is a feasible flow f such that $f(e_s) = g_s(\epsilon^*)$ for $s \in W \cup R$. Hence a ν-flow f^* must satisfy $f^*(e_s) \geq \ell(e_s) + \epsilon^*$ for $s \in R$. It also satisfies $\sum_{s \in R} f^*(e_s) = u(N)$. Therefore, we have $f^*(e_s) = \ell(e_s) + \epsilon^*$ for every $s \in R$. Furthermore, this implies $f^*(e_s) = g_s(\epsilon^*) + \epsilon^*$ for every $s \in W \backslash \{r\}$. Thus, we obtain the following recursive procedure to compute the ν-flow.

Procedure Nu-Flow(p, α)

Step 1: Find ϵ^* with $\tilde{g}_p(\epsilon^*) = \alpha$.

Step 2: Identify the unique maximal subtree $T^* = (W, F)$ of T_p such that $\epsilon_s < \epsilon^*$ for $s \in W \backslash \{p\}$. Let R be the set of vertices in $V_p \backslash W$, adjacent to W.

Step 3: For each $s \in W \backslash \{r\}$, assign $f(e_s) = g_s(\epsilon^*)$. For each $q \in R$, assign $f(e_q) = \ell(e_q) + \epsilon^*$.

Step 4: For each $q \in R$, apply Nu-Flow$(q, f(e_q))$.

In order to accomplish Nu-Flow, we need a preprocess Breaks(ℓ) that computes ϵ_s for each $s \in V \backslash \{r\}$. It detects all the break points of $\tilde{g}_s$ smaller than ϵ_s, and finally it finds all the break points of $\tilde{g}_r$. It also assigns the information on the shape of $\tilde{g}_s$ between 0 and ϵ_s to each $s \in V$.

A naive implementation of this in a bottom-up manner takes $O(n^2)$ time. However, it can be implemented to run in $O(n \log n)$ time with the aid of efficient mergeable heaps as follows.

Suppose all the break points of g_t are stored in a heap H_t for each $t \in \Gamma(s)$. To compute ϵ_s, we construct a heap H_s by merging all H_t and delete the smallest ϵ_q in H_s repeatedly unless $\ell(e) + \epsilon_q > \tilde{g}_s(\epsilon_q)$. Then we are able to compute the break point ϵ_s, which is added to H_s.

While deleting ϵ_q from H_s, the algorithm associates the related information of $\tilde{g}_s$ to $s \in V$. Once ϵ_q is deleted from H_s, it will not appear as a break point. Therefore the total number of deletions from the heap is $O(n)$. Using efficient mergeable heaps such as the 2-3 tree, we are able to perform all these operations in $O(n \log n)$ time [1].

The algorithm for finding the nucleolus x^* is now described as follows.

Algorithm Nucleolus

Step 1: Apply Breaks(ℓ).

Step 2: Apply Nu-Flow($r, u(N)$).

Step 3: For each $i \in N$, assign $x_i^* = f(e_i)$.

We now discuss the time complexity of this algorithm. The preprocess Breaks requires $O(n \log n)$ time. Step 1 of Nu-Flow(p, α) takes linear time in the number of break points of $\tilde{g}_p$ smaller than ϵ_p. As the total number of such break points are at most n, the overall time spent for Step 1 of Nu-Flow during the algorithm is $O(n)$. Step 2 of Nu-Flow uses the depth first search, which requires a linear time in the size of $W \cup R$. Hence the total time spent for Steps 2 and 3 during the algorithm is again $O(n)$. Thus the overall running time of the algorithm is $O(n \log n)$.

5 The egalitarian allocation

In this section, we show that the egalitarian allocation in the game (N, v) can be found in $O(n \log n)$ time.

For an allocation $x \in \mathbf{R}^N$, let $\theta(x)$ be the sequence of numbers x_i for $i \in N$. The egalitarian allocation with respect to a weight vector $w \in \mathbf{R}^N$ is defined to be the unique vector that lexicographically maximizes $\theta(x)$ in the core.

Similarly to Section 4, we investigate the egalitarian allocation of based on network flows. For an arbitrary arc $e \in E$, we define the flow of e and its lower bound as $f(e) = \sum_{i \in A(e)} x_i$ and $l(e) = u(A(e)) = v(N) - v(N \backslash A(e))$, respectively. From Section 3, the core in the game (N, v) is denoted by (5). To have the egalitarian allocation in (N, v), we should lexicographically maximize the flow into N.

The egalitarian allocation with respect to the weight vector w_i is the unique optimal solution to the following separable convex quadratic minimization problem on a flow f in T.

$$\begin{aligned} \text{Minimize} \quad & \sum_{i \in N} \frac{x_i^2}{w_i} \\ \text{subject to} \quad & x(A(e)) = f(e) \quad (e \in E), \\ & \operatorname{val} f = v(N), \\ & f(e) \geq \ell(e) \quad (e \in E). \end{aligned}$$

This is equivalent to the following problem with $\beta = v(N)$ and $\tilde{f}(e) = \beta - f(e)$ for $e \in E$.

$$\text{Minimize} \quad \sum_{i \in N} \frac{(\beta - \tilde{x}_i)^2}{w_i}$$

$$\begin{aligned} \text{subject to} \quad & \tilde{x}(A(e)) = f(e) \quad (e \in E), \\ & \operatorname{val} \tilde{f} = (n-1)\beta, \\ & 0 \le \tilde{f}(e) \le \beta |A(e)| - \ell(e) \quad (e \in E). \end{aligned}$$

Hochbaum and Hong [6] presented an algorithm to solve this latter type of problems in $O(n \log n)$ time. Thus the egalitarian allocation for a tree can be found in $O(n \log n)$ time.

References

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman: *The Design and Analysis of Computer Algorithms*, Addison-Wesley, 1974.

[2] B. Dutta and D. Ray: A concept of egalitarianism under participation constraints. *Econometrica*, vol. 57 (1989), pp. 615–635.

[3] S. Fujishige: Lexicographically optimal base of a polymatroid with respect to a weight vector. *Mathematics of Operations Research*, vol. 5 (1980), pp. 186–196.

[4] Z. Galil: Applications of efficient mergeable heaps for optimization problems on trees. *Acta Informatica*, vol. 13 (1980), pp. 53–58.

[5] D. Granot, M. Mashler, G. Owen, and W. R. Zhu: The kernel/nucleolus of a standard tree game. *International Journal of Game Theory*, vol. 25 (1996), pp. 219–244.

[6] D. Hochbaum and S.-P. Hong: About strongly polynomial time algorithms for quadratic optimization over submodular constraints. *Mathematical Programming*, vol. 69 (1995), pp. 269–309.

[7] N. Megiddo: Optimal flows in networks with multiple sources and sinks. *Mathematical Programming*, vol. 7 (1974), pp. 97–107.

[8] N. Megiddo: Computational complexity of the game theory approach to cost allocation for a tree. *Mathematics of Operations Research*, vol. 3 (1978), pp. 189–196.

[9] D. Schmeidler: The nucleolus of a characteristic function game. *SIAM Journal on Applied Mathematics*, vol. 17 (1969), pp. 1163–1170.

[10] L. S. Shapley: A value for n-person games. *Contributions to the Theory of Games*, H. W. Kuhn and A. W. Tucker, eds., 1953, pp. 305–317.

[11] L. S. Shapley: Cores of convex games. *International Journal of Game Theory*, vol. 1 (1971), pp. 11–26.