A Parametric Flow in Envy-free Cake Cutting

Takao Asano[1,a)]

Abstract: For the cake-cutting problem, Alijani, et al. [2], [8] and Asano and Umeda [3], [4] gave envy-free and truthful mechanisms with a small number of cuts, where the valuation function of each player is a single interval on the given cake. In this paper, we give a much simpler envy-free and truthful mechanism with a small number of cuts. Furthermore, we show that this approach can be applied to the envy-free and truthful mechanism proposed by Chen, et al. [6] where the valuation function of each player is piecewise uniform. Thus, we can obtain an envy-free and truthful mechanism with a small number of cuts, even if the valuation function of each player is piecewise uniform.

Keywords: cake-cutting problem, envy-freeness, fairness, truthfulness, mechanism design, parametric flow

1. Introduction

The problem of dividing a cake among players in a fair manner was first considered by Steinhaus [9]. Formally, the cake-cutting problem is stated as follows: Given a divisible heterogeneous cake C and n strategic players $N = \{1, 2, \ldots, n\}$, where each player $i \in N$ has a valuation function v_i over C, find an allocation of C to the players N that satisfies one or several fairness criteria. In the cake cutting literature, one of the most important criteria is *envy-freeness* [5]. In an envy-free allocation, each player considers his/her own allocation at least as good as any other player's allocation. In recent papers, some restricted classes of valuation functions have been studied. Piecewise constant and piecewise uniform valuation functions are two special classes of valuation functions [2], [5], [6], [8]. For a valuation function v on cake C, let $D(v) = \{x \in C \mid v(x) > 0\}$ (thus, $D(v)$ consists of several disjoint maximal contiguous intervals). Then the valuation function v is called *piecewise constant* if, for each contiguous interval I in $D(v)$, $v(x') = v(x'')$ holds for all $x', x'' \in I$. In a piecewise constant valuation v, if $v(x) = v(y)$ holds for all $x, y \in D(v)$, then v is called a *piecewise uniform* function.

Chen, Lai, Parkes, and Procaccia [6] presented an envy-free and truthful mechanism (i.e., polynomial-time algorithm) for the cake-cutting problem when the valuation functions are piecewise uniform. Aziz and Ye [5] considered the problem when valuation functions are piecewise constant and piecewise uniform. They designed three algorithms called CCEA, MEA and CDA for piecewise constant valuations. They showed that CCEA becomes essentially the same as the envy-free and truthful mechanism proposed by Chen, et al. [6], if it is restricted for piecewise uniform valuations. However, CCEA and the mechanism in [6] uses $\Omega(n \sum_{i \in N} m_i)$ cuts [2], [8], where m_i is the number of maximal contiguous subintervals in $D(v_i) = \{x \in C \mid v_i(x) > 0\}$ in piecewise uniform valuations v_i.

1 Chuo University
a) asano@ise.chuo-u.ac.jp

Alijani, Farhadi, Ghodsi, Seddighin, and Tajik [2], [8] considered that the number of cuts is important and considered the following cake-cutting problem by restricting each piecewise uniform valuation v_i such that $D(v_i) = \{x \in C \mid v_i(x) > 0\}$ is a single contiguous interval C_i in cake C: Given a divisible heterogeneous cake C, n strategic players $N = \{1, 2, \ldots, n\}$ with valuation interval $C_i \subseteq C$ of each player $i \in N$, find a mechanism for dividing C into pieces and allocating pieces of C to n players N to meet the following conditions: (i) the mechanism is envy-free; (ii) the mechanism is truthful; and (iii) the number of cuts made on cake C is small. And they gave an envy-free and truthful mechanism with at most $2n - 2$ cuts based on the expansion process with unlocking [2], [8]. By pointing out that their mechanism is not actually envy free, Asano and Umeda [3], [4] gave an alternative envy-free and truthful mechanism with at most $2n - 2$ cuts.

In this paper, we give a much simpler envy-free and truthful mechanism with a small number of cuts for the above cake-cutting problem. Furthermore, we show that this approach can be applied to the envy-free and truthful mechanism proposed by Chen, Lai, Parkes, and Procaccia [6] for the more general cake-cutting problem where the valuation function of each player is piecewise uniform. Thus, we can obtain an envy-free and truthful mechanism with a small number of cuts, even if the valuation function of each player is piecewise uniform.

2. Preliminaries

We are given a divisible heterogeneous cake $C = [\,0, 1) = \{x \mid 0 \leq x < 1\}$ [*1], n strategic players $N = \{1, 2, \ldots, n\}$ with valuation interval $C_i = [\,\alpha_i, \beta_i) = \{x \mid 0 \leq \alpha_i \leq x < \beta_i \leq 1\} \subseteq C$ of each player $i \in N$. We denote by $\mathcal{C}_N$ the (multi-)set of valuation intervals of all the players N, i.e., $\mathcal{C}_N = (C_1, C_2, \ldots, C_n)$. We also

*1 To guarantee that the pieces allocated to the players by a mechanism are mutually disjoint, we represent a given cake C to be $C = [\,0, 1) = \{x \mid 0 \leq x < 1\}$ in this paper and we assume that if a subinterval $X = [\,x', x'') = \{x \mid x' \leq x < x''\}$ of $C = [\,0, 1)$ is cut at $y \in X$ with $x' < y < x''$ then X is divided into two subintervals $X' = [\,x', y)$ and $X'' = [\,y, x'')$.

write $\mathcal{C}_N = (C_i : i \in N)$.

The valuation intervals $\mathcal{C}_N$ is called *solid*, if, for every point $x \in C$, there is a player $i \in N$ whose valuation interval $C_i \in \mathcal{C}_N$ contains x. As assumed in [2], [5], [8], we will also assume that $\mathcal{C}_N$ is solid throughout this paper, i.e., $\bigcup_{C_i \in \mathcal{C}_N} C_i = C$.

A union X of mutual disjoint sets $X_1, X_2, \ldots, X_k$ is denoted by $X = X_1 + X_2 + \cdots + X_k = \sum_{\ell=1}^{k} X_\ell$. A *piece* A_i of cake C is a union of mutually disjoint subintervals $A_{i_1}, A_{i_2}, \ldots, A_{i_{k_i}}$ of C. Thus, $A_i = A_{i_1} + A_{i_2} + \cdots + A_{i_{k_i}} = \sum_{\ell=1}^{k_i} A_{i_\ell}$. A partition $A_N = (A_1, A_2, \ldots, A_n)$ of cake C into n disjoint pieces $A_1, A_2, \ldots, A_n$ is called an *allocation* of C to n players N if each piece $A_i = \sum_{\ell=1}^{k_i} A_{i_\ell}$ is allocated to player i. We also write $A_N = (A_i : i \in N)$. Thus, $\sum_{i \in N} A_i = C$ in allocation $A_N = (A_i : i \in N)$ of C to n players N, and $A_i = \sum_{\ell=1}^{k_i} A_{i_\ell}$ is called an *allocated piece* of C to player i.

For an interval $X = [\, x', x'')$ of C, the *length* of X, denoted by $len(X)$, is defined by $x'' - x'$. For a piece $A = \sum_{\ell=1}^{k} X_\ell$ of cake C, the *length* of A, denoted by $len(A)$, is defined by the total sum of $len(X_\ell)$, i.e., $len(A) = \sum_{\ell=1}^{k} len(X_\ell)$. For each $i \in N$ and valuation interval C_i of player i, the *value* of piece $A = \sum_{\ell=1}^{k} X_\ell$ for player i, denoted by $V_i(A)$, is the total sum of $len(X_\ell \cap C_i)$, i.e., $V_i(A) = \sum_{\ell=1}^{k} len(X_\ell \cap C_i)$.

For an allocation $A_N = (A_i : i \in N)$ of cake C to n players N, if $V_i(A_i) \geq V_i(A_j)$ for all $j \in N$, then the allocated piece A_i to player i is called *envy-free* for player i. If, for every player $i \in N$, the allocated piece A_i to player i is envy-free for player i, then the allocation $A_N = (A_i : i \in N)$ to n players N is called *envy-free*.

Let $\mathcal{M}$ be a mechanism for the cake-cutting problem. Let $\mathcal{C}_N = (C_i : i \in N)$ be an arbitrary input to $\mathcal{M}$ and $A_N = (A_i : i \in N)$ be an allocation of cake C to n players N obtained by $\mathcal{M}$. If $A_N = (A_i : i \in N)$ with $A_i = \sum_{\ell=1}^{k_i} A_{i_\ell}$ for every input $\mathcal{C}_N = (C_i : i \in N)$ to $\mathcal{M}$ is envy-free then $\mathcal{M}$ is called *envy-free*.

Now, assume that only player i gives a false valuation interval C'_i and let $\mathcal{C}'_N(i) = (C'_j : j \in N)$ (all the other players $j \neq i$ give true valuation intervals C_j and thus $C'_j = C_j$ for each $j \neq i$) be an input to $\mathcal{M}$ and let an allocation of cake C to n players N obtained by $\mathcal{M}$ be $A'_N(i) = (A'_j : j \in N)$ with $A'_j = \sum_{\ell=1}^{k'_j} A'_{j_\ell}$ for each $j \in N$. The values of $A_i = \sum_{\ell=1}^{k_i} A_{i_\ell}$ and $A'_i = \sum_{\ell=1}^{k'_i} A'_{i_\ell}$ for player i are $V_i(A_i) = \sum_{\ell=1}^{k_i} len(A_{i_\ell} \cap C_i)$ and $V_i(A'_i) = \sum_{\ell=1}^{k'_i} len(A'_{i_\ell} \cap C_i)$ (note that $V_i(A'_i) \neq \sum_{\ell=1}^{k'_i} len(A'_{i_\ell} \cap C'_i)$). If $V_i(A_i) \geq V_i(A'_i)$, then player i does not want to give false valuation interval C'_i and player i will report true valuation interval C_i to $\mathcal{M}$ (thus, to report true valuation interval C_i is a *dominant strategy* of player i). For each player $i \in N$, if this holds, then $\mathcal{M}$ is called *truthful* (allocation $A_N = (A_i : i \in N)$ obtained by $\mathcal{M}$ is also called *truthful*).

For given solid valuation intervals $\mathcal{C}_N = (C_i : i \in N)$ and an interval $X = [\, x', x'')$ of cake C, let $N(X)$ be the set of players i in N whose valuation intervals C_i are entirely contained in X and let $\mathcal{C}_{N(X)}$ be the (multi-)set of valuation intervals in $\mathcal{C}_N$ which are entirely contained in X. Let n_X be the cardinality of $N(X)$. Thus, $N(X) = \{i \in N \mid C_i \subseteq X, C_i \in \mathcal{C}_N\}$, $\mathcal{C}_{N(X)} = (C_i \in \mathcal{C}_N : i \in N(X))$, and $n_X = |N(X)|$. The *density* of interval $X = [\, x', x'')$ of C, denoted by $\rho(X)$, is defined by $\rho(X) = \frac{len(X)}{|N(X)|} = \frac{x''-x'}{n_X}$. The density $\rho(X)$ is the average length of pieces of the players in $N(X)$ when the part X of cake C is divided among the players in $N(X)$. Note that, if $X \neq \emptyset$ (i.e., $len(X) \neq 0$) and $n_X = 0$ then $\rho(X) = \infty$. Let $\mathcal{X}$ be the set of all nonempty intervals in C. Let $\rho_{\min}$ be the minimum density among the densities of all nonempty intervals in C, i.e., $\rho_{\min} = \min_{X \in \mathcal{X}} \rho(X)$. Let $\mathcal{X}_{\min} = \{X \in \mathcal{X} \mid \rho(X) = \rho_{\min}\}$. Thus, $\mathcal{X}_{\min}$ is the set of all intervals of minimum density in C. An interval $X \in \mathcal{X}_{\min}$ is called a *maximal interval of minimum density* if no other interval of $\mathcal{X}_{\min}$ contains X properly.

The mechanisms by proposed in [2], [8] and [3], [4] were quite complicated. In this paper, for a given input of cake $C = [\, 0, 1)$, n players $N = \{1, 2, \ldots, n\}$, and solid valuation intervals $\mathcal{C}_N = (C_i : i \in N)$ with valuation interval $C_i = [\, \alpha_i, \beta_i) = \{x \mid 0 \leq \alpha_i \leq x < \beta_i \leq 1\} \subseteq C$ of each player $i \in N$, we give simple envy-free and truthful mechanisms with a small number of cuts. That is, each simple mechanism $\mathcal{M}$ finds an allocation $A_N = (A_i : i \in N)$ to players N satisfying the following properties: (a) $\mathcal{M}$ is envy-free; (b) $\mathcal{M}$ is truthful; (c) $A_i \subseteq C_i$ for each $i \in N$; and (d) $\sum_{i \in N} A_i = C$.

3. Core Mechanism $\mathcal{M}_1$

We first give a core mechanism $\mathcal{M}_1$ which assumes that a cake $C = [0, 1)$ is an interval of minimum density $\rho_{\min}$.

Mechanism 3.1 Core Mechanism $\mathcal{M}_1$**.**

Input: A cake $C = [\, 0, 1)$, n players $N = \{1, 2, \ldots, n\}$ and solid valuation intervals $\mathcal{C}_N = (C_i : i \in N)$ with valuation interval $C_i = [\, \alpha_i, \beta_i)$ of each player $i \in N$ and $\bigcup_{C_i \in \mathcal{C}_N} C_i = C$, where $C = [0, 1)$ is an interval of minimum density $\rho_{\min}$ in cake $C = [0, 1)$.

Output: Allocation $A_N = (A_i : i \in N)$ with $A_i \subseteq C_i$ and $len(A_i) = \rho_{\min}$ for each $i \in N$ and $\sum_{i \in N} A_i = C$.

Algorithm {
 sort the valuation intervals $\mathcal{C}_N = (C_i : i \in N)$ in a lexicographic order with respect to (β_i, α_i) and assume $C_1 \leq C_2 \leq \cdots \leq C_n$ in this lexicographic order;
 set $A_0 = \emptyset$;
 for $i = 1$ **to** n **do**
  set $A_i = [a_i, b_i) \setminus \sum_{i'=0}^{i-1} A_{i'} \subseteq C_i \setminus \sum_{i'=0}^{i-1} A_{i'}$ with length $\rho_{\min}$, where $[a_i, b_i) \subseteq C_i$ and a_i is the leftmost endpoint in $C_i \setminus \sum_{i'=0}^{i-1} A_{i'}$;
}

Fig.1 shows an example of solid valuation intervals $\mathcal{C}_N = (C_i : i \in N)$ ($N = \{1, 2, 3, 4, 5\}$) with $\rho(C) = \rho_{\min} = 0.2$ and an allocation $A_N = (A_i : i \in N)$ obtained by $\mathcal{M}_1$.

Theorem 3.1 For cake $C = [\, 0, 1)$, n players $N = \{1, 2, \ldots, n\}$, and solid valuation intervals $\mathcal{C}_N = (C_i : i \in N)$ with valuation interval $C_i = [\, \alpha_i, \beta_i)$ of each player $i \in N$, let $[0, 1)$ be an interval of minimum density $\rho_{\min}$ in cake $C = [\, 0, 1)$. Then, $\mathcal{M}_1$ finds an allocation $A_N = (A_i : i \in N)$ with $A_i \subseteq C_i$ and $len(A_i) = \rho_{\min}$ for each $i \in N$ and $\sum_{i \in N} A_i = C$. Furthermore, the number of cuts made on cake C is at most $2n - 2$.

Proof: It is clear that the number of cuts made on cake C is at most $2n - 2$, since $\mathcal{M}_1$ uses at most two cuts at a_i and b_i to obtain $A_i = [a_i, b_i) \setminus \sum_{i'=0}^{i-1} A_{i'}$ and no cut is required at $0, 1$, the endpoints of cake $C = [0, 1)$.

We next prove that $\mathcal{M}_1$ correctly finds an allocation $A_N = (A_i : i \in N)$ with $A_i \subseteq C_i$, $len(A_i) = \rho_{\min}$ and $\sum_{i \in N} A_i = C$.

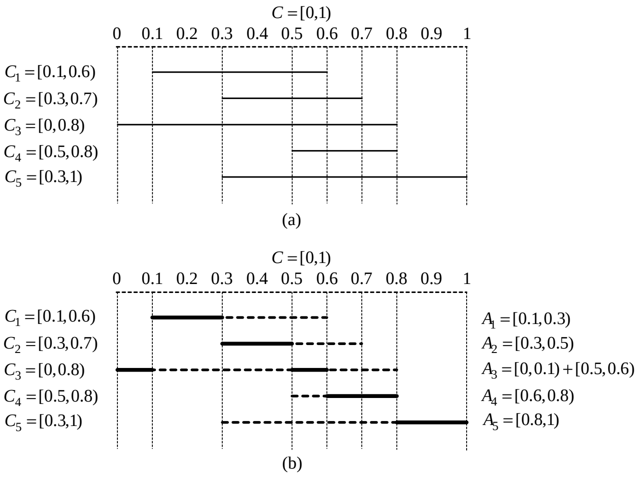

Fig. 1 (a) Solid valuation intervals $\mathcal{C}_N = (C_i : i \in N)$ with $\rho(C) = \rho_{\min} = 0.2$. (b) Allocation $A_N = (A_i : i \in N)$ obtained by $\mathcal{M}_1$.

Suppose contrarily that we could not set $A_i = [a_i, b_i) \setminus \sum_{i'=0}^{i-1} A_{i'} \subseteq C_i$ with length $\rho_{\min}$ for some $i \in N$. Let j be the minimum among such i's and let $J = \{1, 2, \ldots, j\}$. Thus, we could set $A_i = [a_i, b_i) \setminus \sum_{i'=0}^{i-1} A_{i'} \subseteq C_i = [\alpha_i, \beta_i)$ with length $\rho_{\min}$ for each $i \in J \setminus \{j\}$ but could not set $A_j = [a_j, b_j) \setminus \sum_{i'=0}^{j-1} A_{i'} \subseteq C_j = [\alpha_j, \beta_j)$ with length $\rho_{\min}$. This implies that $C_j \setminus \sum_{i'=0}^{j-1} A_{i'} = [a_j, \beta_j)$ is of length ρ_j less than $\rho_{\min}$. Now we consider valuation intervals $\mathcal{C}_J = (C_i : i \in J)$. Note that each $C_i = [\alpha_i, \beta_i) \in \mathcal{C}_J$ satisfies $\beta_i \leq \beta_j$, since the valuation intervals in $\mathcal{C}_N = (C_i : i \in N)$ were sorted in the lexicographic order with respect to (β_i, α_i). Let

$$A'_i = A_i \quad (i \in J \setminus \{j\}) \quad \text{and} \quad A'_j = C_j \setminus \textstyle\sum_{i'=0}^{j-1} A_{i'} = [a_j, \beta_j).$$

Allocation $(A'_i : i \in J)$ (i.e., $\sum_{i' \in J} A'_i$) consists of several maximal contiguous intervals. Let $I = [a, b)$ be the rightmost maximal contiguous interval among the maximal contiguous intervals in allocation $(A'_i : i \in J)$. Thus, $b = \beta_j$. Let $K \subseteq J$ be the set of all $i \in J$ with $A'_i \cap I \neq \emptyset$, i.e., $K = \{i \in J \mid A'_i \cap I \neq \emptyset\}$. If $A'_j = C_j \setminus \sum_{i'=0}^{j-1} A_{i'} = [a_j, \beta_j) = \emptyset$, then we modify $K = K \cup \{j\}$. Thus, in any case (i.e., $A'_j = \emptyset$ or not),

$$K = \{j\} \cup \{i \in J \mid A'_i \cap I \neq \emptyset\}.$$

Now we consider valuation intervals $\mathcal{C}_K = (C_i : i \in K)$. Then each $C_i \in \mathcal{C}_K$ is contained in I. This can be obtained as follows.

Of course, $C_j = [\alpha_j, \beta_j)$ is contained in I. Actually, since $C_j \setminus \sum_{i'=0}^{j-1} A_{i'} = [a_j, \beta_j)$ is of length $< \rho_{\min}$ and $A'_j = C_j \setminus \sum_{i'=0}^{j-1} A_{i'} = [a_j, \beta_j)$, if $A'_j = \emptyset$ then $C_j \subseteq \sum_{i'=0}^{j-1} A_{i'}$ and a single contiguous interval C_j is contained in the rightmost maximal contiguous interval I in $\sum_{i'=0}^{j-1} A_{i'} = A'_j \cup \sum_{i'=0}^{j-1} A_{i'}$ (i.e., $C_j \subseteq I$), and otherwise (i.e., if $A'_j \neq \emptyset$), $C_j \subseteq A'_j \cup \sum_{i'=0}^{j-1} A_{i'}$ and a single contiguous interval C_j is contained in the rightmost maximal contiguous interval I in $A'_j \cup \sum_{i'=0}^{j-1} A_{i'}$.

Now suppose that there were some $C_i \in \mathcal{C}_K \setminus \{C_j\}$ not contained in I. Thus, $C_i = [\alpha_i, \beta_i) \in \mathcal{C}_K$ $(i \in K \setminus \{j\})$ would contain a point x in $[0, b) \setminus I = [0, \beta_j) \setminus I = [0, a)$. Let $k \in K$ be the minimum among such i's and let $C_k = [\alpha_k, \beta_k) \in \mathcal{C}_K$ contains a point x_k in $[0, b) \setminus I = [0, \beta_j) \setminus I = [0, a)$. Thus, $\beta_k \leq \beta_j$ and $\alpha_k \leq x_k < a \leq a'_k < \beta_k$ for some $a'_k \in A'_k \cap I \neq \emptyset$ since $k \in K \setminus \{j\} \subseteq J \setminus \{j\}$. Furthermore, since we chose $I = [a, b)$ as the rightmost maximal contiguous interval among the maximal contiguous intervals in allocation $(A'_i : i \in J)$, there is the maximal contiguous interval $I' = [a', a)$ which is not contained in allocation $(A'_i : i \in J)$. Since $C_k = [\alpha_k, \beta_k)$ is a contiguous interval and satisfies $\alpha_k \leq x_k < a \leq a'_k < \beta_k$, we can assume $x_k \in I' \cap C_k \neq \emptyset$. Thus, $x_k \notin A'_k$. Then, however, $\mathcal{M}_1$ would have included x_k into A'_k in place of some $a''_k \in A'_k \cap I \neq \emptyset$, because $\mathcal{M}_1$ sets $A'_k = A_k = [a_k, b_k) \setminus \sum_{i'=0}^{k-1} A_{i'} \subseteq C_k \setminus \sum_{i'=0}^{k-1} A_{i'}$ with length $\rho_{\min}$ where $[a_k, b_k) \subseteq C_k$ and a_k is the leftmost endpoint in $C_k \setminus \sum_{i'=0}^{k-1} A_{i'}$. This is a contradiction. Thus, we have each $C_i \in \mathcal{C}_K$ is contained in I and $\bigcup_{i \in K} C_i \subseteq I$.

By the argument above, we have $\bigcup_{i \in K} C_i = I = \sum_{i \in K} A'_i$, since $\bigcup_{i \in K} C_i \subseteq I$ and $I = \sum_{j \in J} A'_j \cap I = \sum_{i \in K} A'_i \cap I \subseteq \sum_{i \in K} A'_i \subseteq \sum_{i \in K} C_i$ by the definitions of I and K and $A'_i \subseteq C_i$ for each $i \in K$. Thus, $K = N(I)$ and $n_I = |N(I)|$. Furthermore, by noting that $len(A'_j) = \rho_j < \rho_{\min}$ for $j \in K$ and $len(A'_i) = \rho_{\min}$ for each $i \in K \setminus \{j\}$, we have

$$\rho_j + \sum_{i \in K \setminus \{j\}} \rho_{\min} = len(I) = b - a < \rho_{\min} + \sum_{i \in K \setminus \{j\}} \rho_{\min} = |K| \rho_{\min}.$$

Thus, $\rho(I) = \frac{len(I)}{|K|} < \rho_{\min}$. However, this is a contradiction, since $C = [0, 1)$ is the maximal interval of minimum density $\rho_{\min}$.

Thus, $\mathcal{M}_1$ correctly finds an allocation $A_N = (A_i : i \in N)$ with $A_i \subseteq C_i$, $len(A_i) = \rho_{\min}$ and $\sum_{i \in N} A_i = C$. □

$\mathcal{M}_1$ can be used as a procedure in the mechanism proposed for the cake-cutting problem in Asano and Umeda [4], where cake $C = [0, 1)$ is not necessarily an interval of minimum density $\rho_{\min}$. Actually, $\mathcal{M}_1$ is CutMaxInterval$(N, C, \mathcal{C}_N)$ when C is a maximal interval of minimum density $\rho_{\min}$ in cake C. Thus, $\mathcal{M}_1$ can be modified a little and used in place of Procedure CutMaxInterval$(R, H, \mathcal{D}_R)$ as follows, where H is a maximal interval of minimum density $\rho_{\min}$, $R = N(H)$, and valuation interval $D_i \in \mathcal{D}_R$ of each player $i \in R$ is $D_i = C_i \in \mathcal{C}_R$ (thus, $\mathcal{D}_R = \mathcal{C}_R$).

Procedure 3.1 CutMaxInterval($R, H, \mathcal{D}_R$) {
  sort the valuation intervals $\mathcal{D}_R = (D_i = (\alpha_i, \beta_i) : i \in R)$ in a lexicographic order with respect to (β_i, α_i) and assume $D_{R_1} \leq D_{R_2} \leq \cdots \leq D_{R_r}$ in this lexicographic order where $r = |R|$;
  set $A_{R_0} = \emptyset$;
  for $i = 1$ **to** r **do**
    set $A_{R_i} = [a_{R_i}, b_{R_i}) \setminus \sum_{i'=0}^{i-1} A_{R_{i'}} \subseteq D_{R_i} \setminus \sum_{i'=0}^{i-1} A_{R_{i'}}$ with length $\rho_{\min}$, where $[a_{R_i}, b_{R_i}) \subseteq D_{R_i}$ and a_{R_i} is the leftmost endpoint in $D_{R_i} \setminus \sum_{i'=0}^{i-1} A_{R_{i'}}$;
}

Thus, their mechanism in Asano and Umeda [4] can be written as follows, although we omit the details.

Mechanism 3.2 Their cake-cutting mechanism in [4].
Input: A cake $C = [0, 1)$, n players $N = \{1, 2, \ldots, n\}$ and solid valuation intervals $\mathcal{C}_N$ with valuation interval $C_i = [\alpha_i, \beta_i)$ of each player $i \in N$ and $\bigcup_{C_i \in \mathcal{C}_N} C_i = C$.
Output: Allocation $A_N = (A_i : i \in N)$ to players N.
Algorithm { CutCake$(N, C, \mathcal{C}_N)$; }

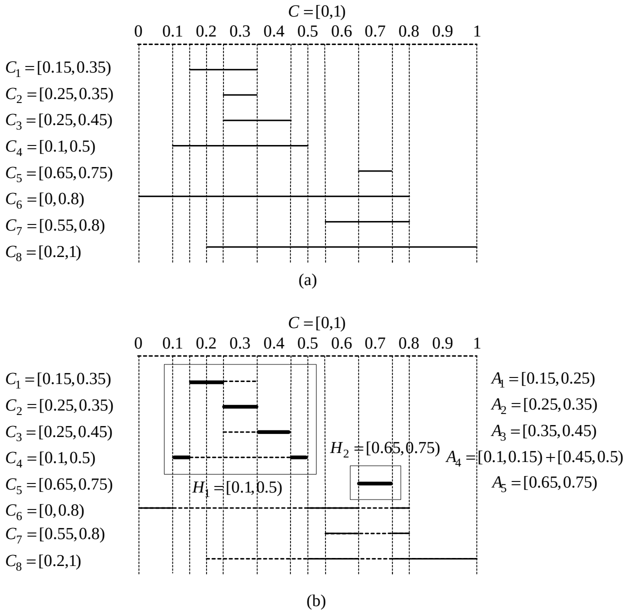

Fig. 2 (a) Example of solid valuation intervals $\mathcal{C}_N = (C_i : i \in N)$ $(N = \{1, 2, \ldots, 8\})$. (b) In first iteration, maximal intervals H_1, H_2 of minimum density $\rho_{\min} = 0.1$ with $N(H_1) = \{1, 2, 3, 4\}$ and $N(H_2) = \{5\}$, and allocation $A_{N(H_1)} = (A_i : i \in N(H_1))$ and allocation $A_{N(H_2)} = (A_5)$.

Procedure 3.2 CutCake$(P, D, \mathcal{D}_P)$ {

Find all the maximal intervals of minimum density $\rho_{\min}$ in the cake-cutting problem with cake D, players P and solid valuation intervals $\mathcal{D}_P$;

Let $H_1 = [\,h'_1, h''_1),\ H_2 = [\,h'_2, h''_2), \ldots,\ H_L = [\,h'_L, h''_L)$ be all the maximal intervals of minimum density $\rho_{\min}$;

for $\ell = 1$ **to** L **do**

cut cake D at both endpoints h'_ℓ, h''_ℓ of H_ℓ;

$R_\ell = \{k \in P \mid D_k \subseteq H_\ell, D_k \in \mathcal{D}_P\}$; $\mathcal{D}_{R_\ell} = (D_k \in \mathcal{D}_P :\ k \in R_\ell)$;

CutMaxInterval$(R_\ell, H_\ell, \mathcal{D}_{R_\ell})$;

$P' = P$; $D' = D$;

for $\ell = 1$ **to** L **do** $P' = P' \setminus R_\ell$; $D' = D' \setminus H_\ell$;

if $P' \neq \emptyset$ **then** // $P' = P \setminus \sum_{\ell=1}^{L} R_\ell$ and $D' = D \setminus \sum_{\ell=1}^{L} H_\ell$

$\mathcal{D}'_{P'} = \emptyset$;

for each $D_k \in \mathcal{D}_P$ with $k \in P'$ **do**

$D'_k = D_k \setminus \sum_{\ell=1}^{L} H_\ell$; $\mathcal{D}'_{P'} = \mathcal{D}'_{P'} + \{D'_k\}$;

Perform virtually shrinking of all $H_1, H_2, \ldots, H_L$;

Let $D^{(S)}, D^{(S)}_k \in \mathcal{D}^{(S)}_{P'}, \mathcal{D}^{(S)}_{P'}$ be obtained from $D', D'_k \in \mathcal{D}'_{P'}$, $\mathcal{D}'_{P'}$ by virtually shrinking of all $H_1, H_2, \ldots, H_L$;

CutCake$(P', D^{(S)}, \mathcal{D}^{(S)}_{P'})$;

}

For an input example in Fig.2(a), their cake-cutting mechanism given above works as shown in Fig.2(b) and Fig.3. Note that the original CutMaxInterval$(R, H, \mathcal{D}_R)$ in Asano and Umeda [4] was complicated because it was based on the quite complicated core method for solving the cake-cutting problem where cake X is a minimal interval of minimum density $\rho_{\min}$ in maximal interval H of minimum density $\rho_{\min}$. The following theorem holds.

Theorem 3.2 [4] Asano and Umeda's mechanism correctly finds, in $O(n^3)$ time, an envy-free and truthful allocation $A_N = (A_i : i \in N)$ of cake C to n players N with $A_i \subseteq C_i$ for each player $i \in N$ and $\sum_{i \in N} A_i = C$. Furthermore, the number of cuts made over C by Mechanism 3.2 is at most $2n - 2$.

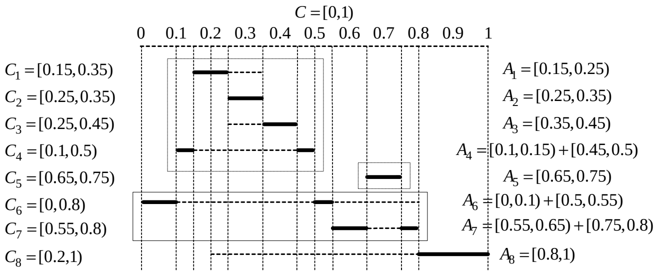

Fig. 3 The second and third iterations for the example in Fig.2. In the second iteration, the minimum density is $\rho_{\min} = 0.15$ and $N(H_1) = \{6, 7\}$, $A_6 = [0, 0.1) + [0.5, 0.55)$ and $A_7 = [0.55, 0.65) + [0.75, 0.8)$. In the third (last) iteration, the minimum density is $\rho_{\min} = 0.2$ and $N(H_1) = \{8\}$ and $A_8 = [0.8, 1)$.

We can improve the time complexity from $O(n^3)$ to $O(n^2 \log n)$ using a parametric flow [7] in the later sections.

4. Second Version $\mathcal{M}_2$

In this section, we give the second version $\mathcal{M}_2$ which can be applied to the envy-free and truthful mechanism proposed by Chen, et al. [6] when the valuation function of each player is piecewise uniform. We are given a cake $C = [\,0, 1)$, n players $N = \{1, 2, \ldots, n\}$, and solid valuation intervals $\mathcal{C}_N = (C_i : i \in N)$ with valuation interval $C_i = [\,\alpha_i, \beta_i) \subseteq C$ of each player $i \in N$. We are also given $(s_i : i \in N)$ such that there is an allocation $A'_N = (A'_i : i \in N)$ to players N with $A'_i \subseteq C_i$ and $s_i = len(A'_i) > 0$ for each $i \in N$ and $\sum_{i \in N} A'_i = C$ (thus $\sum_{i \in N} s_i = 1$). Then $\mathcal{M}_2$ is almost the same as $\mathcal{M}_1$ and can be written as follows.

Mechanism 4.1 Second Version $\mathcal{M}_2$**.**

Input: A cake $C = [\,0, 1)$, n players $N = \{1, 2, \ldots, n\}$ and solid valuation intervals $\mathcal{C}_N = (C_i : i \in N)$ with valuation interval $C_i = [\,\alpha_i, \beta_i)$ of each player $i \in N$ and $\bigcup_{C_i \in \mathcal{C}_N} C_i = C$ and $(s_i : i \in N)$ for players N such that there is an allocation $A'_N = (A'_i : i \in N)$ to players N with $A'_i \subseteq C_i$ and $len(A'_i) = s_i > 0$ for each $i \in N$ and $\sum_{i \in N} A'_i = C$ (thus $\sum_{i \in N} s_i = 1$).

Output: Allocation $A_N = (A_i : i \in N)$ with $A_i \subseteq C_i$ and $len(A_i) = s_i$ for each $i \in N$ and $\sum_{i \in N} A_i = C$.

Algorithm {

sort the valuation intervals $\mathcal{C}_N = (C_i : i \in N)$ in a lexicographic order with respect to (β_i, α_i) and assume $C_1 \leq C_2 \leq \cdots \leq C_n$ in this lexicographic order;

set $A_0 = \emptyset$;

for $i = 1$ **to** n **do**

set $A_i = [a_i, b_i) \setminus \sum_{i'=0}^{i-1} A_{i'} \subseteq C_i \setminus \sum_{i'=0}^{i-1} A_{i'}$ with length s_i, where $[a_i, b_i) \subseteq C_i$ and a_i is the leftmost endpoint in $C_i \setminus \sum_{i'=0}^{i-1} A_{i'}$;

}

Fig.4 shows an input example of solid valuation intervals $\mathcal{C}_N = (C_i : i \in N)$ and $(s_i : i \in N)$ with $\sum_{i \in N} s_i = 1$ and an allocation $A_N = (A_i : i \in N)$ obtained by $\mathcal{M}_2$. By an argument similar to one in Theorem 3.1 we have the following theorem.

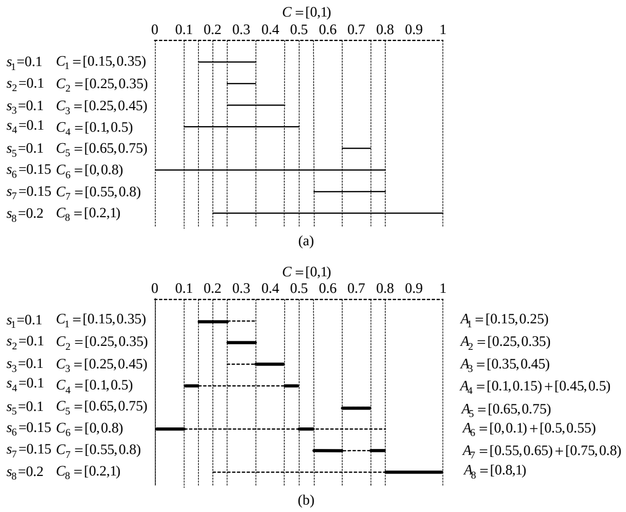

Fig. 4 (a) Example of solid valuation intervals $\mathcal{C}_N = (C_i : i \in N)$ ($N = \{1, 2, \ldots, 8\}$) and ($s_i : i \in N$) with $\sum_{i \in N} s_i = 1$. (b) Allocation $A_N = (A_i : i \in N)$ obtained by $\mathcal{M}_2$.

Theorem 4.1 $\mathcal{M}_2$ correctly finds an allocation $A_N = (A_i : i \in N)$ with $A_i \subseteq C_i$ and $len(A_i) = s_i$ for each $i \in N$ and $\sum_{i \in N} A_i = C$. Furthermore, the number of cuts made on cake C is at most $2n-2$.

By Theorem 4.1, in order to obtain an envy-free allocation $A_N = (A_i : i \in N)$ with $A_i \subseteq C_i$ and $len(A_i) = s_i$ for each $i \in N$ and $\sum_{i \in N} A_i = C$, we only need ($s_i : i \in N$) such that there is an envy-free allocation $A'_N = (A'_i : i \in N)$ to players N with $A'_i \subseteq C_i$ and $len(A'_i) = s_i$ for each $i \in N$ and $\sum_{i \in N} A'_i = C$. Furthermore, Theorem 4.1 can be applied to Chen, et al.'s mechanism for the cake-cutting problem when the valuation function v_i of each player $i \in N$ is piecewise uniform [6]: Given a cake $C = [\,0, 1)$, n players $N = \{1, 2, \ldots, n\}$ and solid pricewise-uniform valuation functions ($v_i : i \in N$) such that $D(v_i) = \{x \in C \mid v_i(x) > 0\}$ of each valuation function v_i consists of $i_{m_i} \geq 1$ disjoints intervals in C (i.e., $D(v_i) = \sum_{j=1}^{m_i} C_{i_j}$ where C_{i_j} is a single maximal interval in C) and $\bigcup_{i \in N} D(v_i) = C$. Chen, et al.'s mechanism finds an envy-free allocation $A'_N = (A'_i : i \in N)$ such that $A'_i = \sum_{j=1}^{m_i} A'_{i_j}$ with $A'_{i_j} \subseteq C_{i_j}$ for each $i \in N$, each $j = 1, 2, \ldots, m_i$ and $\sum_{i \in N} A'_i = C$. Thus, we can set $s_{i_j} = len(A'_{i_j})$ and apply Theorem 4.1 to obtain an envy-free allocation $A_N = (A_i : i \in N)$ such that $A_i = \sum_{j=1}^{m_i} A_{i_j}$ with $A_{i_j} \subseteq C_{i_j}$ and $len(A_{i_j}) = s_{i_j}$ with at most $2(\sum_{i \in N} m_i) - 2$ cuts.

5. Flow Network on Valuation Intervals

In this section, we consider a flow network arising from valuation intervals to find such ($s_i : i \in N$) when $D(v_i) = \{x \in C \mid v_i(x) > 0\}$ of valuation function v_i is a single interval C_i in C.

Let X_N be the set of all endpoints α_i, β_i of $C_i = [\,\alpha_i, \beta_i)$ of $\mathcal{C}_N = (C_i : i \in N)$ and we assume the elements in X_N are sorted $x_0 < x_1 < \cdots < x_{n'}$ where $x_0 = 0$, $x_{n'} = 1$ and $n' \leq 2n - 1$. For each j with $1 \leq j \leq n'$, let $I_j = [x_{j-1}, x_j)$ and let $\mathcal{I}_N = (I_j : 1 \leq j \leq n')$. Let $G_N = (\mathcal{C}_N, \mathcal{I}_N, E_N)$ be a bipartite graph with vertex set $V_N = \mathcal{C}_N + \mathcal{I}_N$ and edge set E_N where $(C_i, I_j) \in E_N$ if and only if $I_j \subseteq C_i$. $G_N = (\mathcal{C}_N, \mathcal{I}_N, E_N)$ is called a *convex bipartite graph* since it has a property that if $(C_i, I_j), (C_i, I_{j'}) \in E_N$ with $j < j'$ then $(C_i, I_{j''}) \in E_N$ for each j'' with $j < j'' < j'$ (Fig.5).

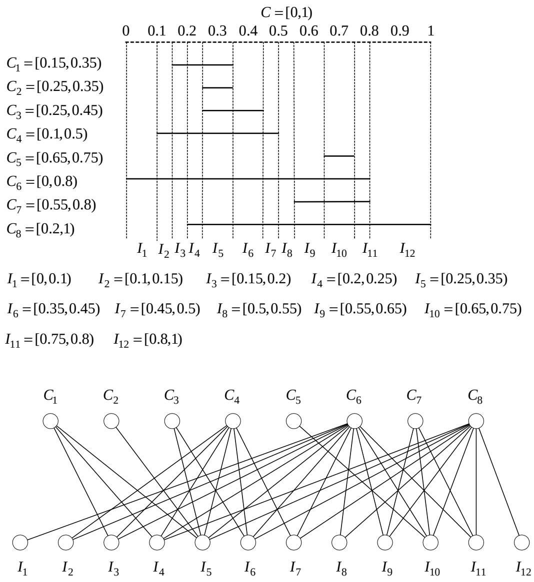

Fig. 5 Example of valuation intervals $\mathcal{C}_N = (C_i : i \in N)$ ($N = \{1, 2, \ldots, 8\}$) $\mathcal{I}_N = (I_j : 1 \leq j \leq n')$ ($j = 1, 2, \ldots, 12$) and the convex bipartite graph $G_N = (\mathcal{C}_N, \mathcal{I}_N, E_N)$.

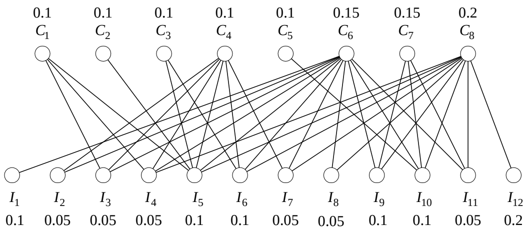

Fig. 6 Network $H_N = (G_N, S_N, T_N)$ corresponding to example in Fig.5 with $s_1 + s_2 + \cdots + s_n = 1$.

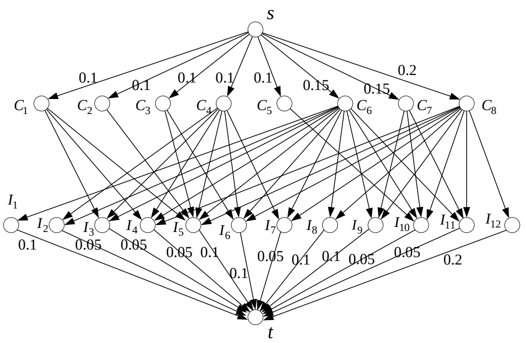

Fig. 7 Network $H_N(s, t) = (G_N, \mathrm{capa}_N, s, t)$ corresponding to $H_N = (G_N, S_N, T_N)$ in Fig.6.

Now we assume that we are given a positive number s_i for each valuation interval $C_i \in \mathcal{C}_N$ and a positive number $t_j = len(I_j)$ for each $I_j \in \mathcal{I}_N$. Let $S_N = (s_1, s_2, \ldots, s_n)$ and $T_N = (t_1, t_2, \ldots, t_{n'})$. Note that $t_1 + t_2 + \cdots + t_{n'} = 1$. Let $H_N = (G_N, S_N, T_N)$ be a network on convex bipartite graph G_N with supply s_i of each vertex C_i and demand t_j of each vertex I_j (Fig.6). A function $f : E_N \to \mathbf{R}_+$ is called an *flow* in H_N and a flow f in H_N is called *feasible*, if $s_i = \sum_{e \in \delta(C_i)} f(e)$ for each $C_i \in \mathcal{C}_N$ and $t_j = \sum_{e \in \delta(I_j)} f(e)$ for each $I_j \in \mathcal{I}_N$, where $\delta(v)$ is the set of edges in E_N incident to vertex v in G_N. It is clear that if H_N has a feasible flow then $s_1 + s_2 + \cdots + s_n = 1$.

We also consider a network $H_N(s, t) = (G_N, \mathrm{capa}_N, s, t)$ which is obtained $H_N = (G_N, S_N, T_N)$ by adding two new vertices s, t and adding a directed edge (s, C_i) with capacity $\text{capa}_N(s, C_i) = s_i$ for each $C_i \in \mathcal{C}_N$ and a directed edge (I_j, t) with capacity $\text{capa}_N(I_j, t) = t_j$ for each $I_j \in \mathcal{I}_N$. We assume that each edge $(C_i, I_j) \in E_N$ is directed from C_i to I_j and has an infinite capacity $\text{capa}_N(C_i, I_j) = \infty$ (Fig.7). We denote by $V_N(s, t)$ and $E_N(s, t)$ the set of all vertices and the set of all directed edges in $H_N(s, t)$, respectively. Thus, $V_N(s, t) = V_N + \{s, t\} = \mathcal{C}_N + \mathcal{I}_N + \{s, t\}$ and $E_N(s, t) = E_N + \{(s, C_i) \mid C_i \in \mathcal{C}_N\} + \{(I_j, t) \mid I_j \in \mathcal{I}_N\}$. A function $f : E_N(s, t) \to \mathbf{R}_+$ is an *s-t flow* in $H_N(s, t)$ if (i) and (ii) hold:

(i) $0 \leq f(s, C_i) \leq s_i$ for each edge (s, C_i) and $0 \leq f(I_j, t) \leq t_j$ for each edge (I_j, t), and

(ii) $f(s, C_i) = \sum_{e=(C_i, I_j) \in \delta^+(C_i)} f(e)$ for each $C_i \in \mathcal{C}_N$ and $f(I_j, t) = \sum_{e=(C_i, I_j) \in \delta^-(I_j)} f(e)$ for each $I_j \in \mathcal{I}_N$, where $\delta^+(C_i)$ is the set of directed edges in $H_N(s, t)$ leaving from C_i and $\delta^-(I_j)$ is the set of directed edges in $H_N(s, t)$ entering into I_j.

The *value* of an *s-t* flow f in $H_N(s, t)$, denoted by $\text{val}(f)$, is defined by $\text{val}(f) = \sum_{C_i \in \mathcal{C}_N} f(s, C_i)$. Clearly, $\text{val}(f) = \sum_{I_j \in \mathcal{I}_N} f(I_j, t))$ by the above condition (ii). An *s-t* flow f in $H_N(s, t)$ is called *maximum* if $\text{val}(f) \geq \text{val}(f')$ for all *s-t* flow f' in $H_N(s, t)$. A partition $(Y, \overline{Y})$ of vertex set $V_N(s, t) = \mathcal{C}_N + \mathcal{I}_N + \{s, t\}$ is called an *s-t cut* in $H_N(s, t)$ if $s \in Y$ and $t \in \overline{Y}$. We also call $E(Y, \overline{Y}) = \{e = (y, y') \in E_N(s, t) \mid y \in Y, y' \in \overline{Y}\}$ the *s-t* cut in $H_N(s, t)$ defined by *s-t* cut $(Y, \overline{Y})$. The *capacity* of an *s-t* cut $(Y, \overline{Y})$ in $H_N(s, t)$, denoted by $\text{capa}_N(Y, \overline{Y})$, is defined by the sum of the capacities $\text{capa}_N(e)$ of all edges $e = (y, y') \in E_N(s, t)$ with $y \in Y$ and $y' \in \overline{Y}$, i.e., $\text{capa}_N(Y, \overline{Y}) = \sum_{e=(y,y') \in E_N(s,t):\, y \in Y,\, y' \in \overline{Y}} \text{capa}_N(e)$. An *s-t* cut $(Y, \overline{Y})$ in $H_N(s, t)$ is called *minimum* if $\text{capa}_N(Y, \overline{Y}) \leq \text{capa}_N(Y', \overline{Y'})$ for all *s-t* cuts $(Y', \overline{Y'})$ in $H_N(s, t)$. For any *s-t* flow f and any *s-t* cut $(Y, \overline{Y})$ in $H_N(s, t)$, $\text{val}(f) \leq \text{capa}_N(Y, \overline{Y})$ holds. Furthermore, $\text{val}(f) = \text{capa}_N(Y, \overline{Y})$ holds if and only if f is a minimum *s-t* flow and $(Y, \overline{Y})$ is a minimum *s-t* cut in $H_N(s, t)$ (the well-known maximum-flow and minimum-cut theorem).

For a flow f in $H_N(s, t)$, a residual network with respect to f, denoted by $H_N(s, t)(f)$, is defined as follows. The vertex set $V_N(s, t)(f)$ of $H_N(s, t)(f)$ is the vertex set $V_N(s, t)$ of $H_N(s, t)$. The edge set $E_N(s, t)(f)$ of $H_N(s, t)(f)$ is defined as follows. For an edge $e = (u, v)$ of $H_N(s, t)$, let $e^{\text{rev}} = (v, u)$ (i.e., $e^{\text{rev}} = (v, u)$ is the reverse edge of $e = (u, v) \in E_N(s, t)$). Let $E_N^{\text{rev}}(s, t) = \{e^{\text{rev}} \mid e \in E_N(s, t)\}$. The residual capacity $\text{capa}_f(a)$ of an edge $a = (u, v) \in E_N(s, t) + E_N^{\text{rev}}(s, t)$, is defined as follows:

$$\text{capa}_f(a) = \begin{cases} \text{capa}(a) - f(a) & (a \in E_N(s, t)) \\ f(e) & (a = e^{\text{rev}},\ e \in E_N(s, t)). \end{cases} \tag{1}$$

Then the edge set $E_N(s, t)(f)$ of $H_N(s, t)(f)$ is defined by

$$E_N(s, t)(f) = \{a \in E_N(s, t) + E_N^{\text{rev}}(s, t) \mid \text{capa}_f(a) > 0\}.$$

Thus, the capacity of each edge a of $H_N(s, t)(f)$ is $\text{capa}_f(a) > 0$. It is well known that an *s-t* flow f in $H_N(s, t)$ is maximum if and only if there is no *s-t* path in the residual network $H_N(s, t)(f)$. Furthermore, suppose that there is no *s-t* path in $H_N(s, t)(f)$, and let $\overline{Y}$ be the set of vertices v such that there is a v-t path in $H_N(s, t)(f)$ and $Y = V_N(s, t) \setminus \overline{Y}$. Then $(Y, \overline{Y})$ in $H_N(s, t)$ is a minimum *s-t* cut and $Y' \subseteq Y$ holds for each minimum *s-t* cut $(Y', \overline{Y'})$ in $H_N(s, t)$.

5.1 A Parametric Flow on Valuation Intervals

For a parameter λ with $0 \leq \lambda \leq 1$, let $s_i = \lambda$ for each $i \in N$. We denote by $S_{N(\lambda)}$ this special $S_N = (s_1, s_2, \ldots, s_n)$ with $s_i = \lambda$ for each $i \in N$, i.e., $S_{N(\lambda)} = (\lambda, \lambda, \ldots, \lambda) = (s_1, s_2, \ldots, s_n) = S_N$. Let $T_N = (t_1, t_2, \ldots, t_{n'})$ with $t_1 + t_2 + \cdots + t_{n'} = 1$ as before. We use $H_{N(\lambda)} = (G_N, S_{N(\lambda)}, T_N)$ and $H_{N(\lambda)}(s, t)$ when we emphasize $S_{N(\lambda)} = (\lambda, \lambda, \ldots, \lambda) = (s_1, s_2, \ldots, s_n) = S_N$ (this network $H_{N(\lambda)}(s, t)$ is proposed by Chen, et al. [6]). An *s-t* flow f in $H_{N(\lambda)}(s, t)$ is called a *parametric flow* in $H_{N(\lambda)}(s, t)$. Parametric flows and parametric searching were considered in [1], [7], [10].

The density $\rho(X)$ of interval $X = [x', x'')$ of cake $C = [0, 1)$ is closely related to this parameter λ. For a maximum flow f_λ in $H_{N(\lambda)}(s, t)$ found in this paper, $\overline{Y_\lambda}$ throughout this paper is the set of vertices v such that there is a v-t path in $H_{N(\lambda)}(s, t)(f_\lambda)$ and let $Y_\lambda = V_N(s, t) \setminus \overline{Y_\lambda}$. Then $(Y_\lambda, \overline{Y_\lambda})$ is a minimum *s-t* cut in $H_{N(\lambda)}(s, t)$ and $Y'_\lambda \subseteq Y_\lambda$ (thus, $\overline{Y_\lambda} \subseteq \overline{Y'_\lambda}$) holds for each minimum *s-t* cut $(Y'_\lambda, \overline{Y'_\lambda})$ in $H_{N(\lambda)}(s, t)$. That is, $\overline{Y_\lambda}$ is a minimal set among the minimum *s-t* cuts $(Y'_\lambda, \overline{Y'_\lambda})$ in $H_{N(\lambda)}(s, t)$. Furthermore, for two distinct parameters $\lambda' < \lambda$, $\overline{Y_{\lambda'}} \supseteq \overline{Y_\lambda}$ (i.e., $Y_{\lambda'} \subseteq Y_\lambda$) holds.

Specifically, for $\lambda = \rho_{\min}$ and the minimum *s-t* cut $(Y_\lambda, \overline{Y_\lambda})$ in $H_{N(\lambda)}(s, t)$ defined above, Y_λ is the disjoint union of all maximal intervals of minimum density $\rho_{\min}$ and its capacity $\text{capa}(Y_\lambda, \overline{Y_\lambda})$ is $\text{capa}(Y_\lambda, \overline{Y_\lambda}) = \lambda|\overline{Y_\lambda} \cap \mathcal{C}_N| + \sum_{v \in Y_\lambda \cap \mathcal{I}_N} \text{capa}(v, t)$. Of course, $\lambda|\overline{Y_\lambda} \cap \mathcal{C}_N| = \sum_{v \in \overline{Y_\lambda} \cap \mathcal{C}_N} \text{capa}(s, v)$, since $\text{capa}(s, v) = \lambda$ for each $v \in \mathcal{C}_N$. There are at most n distinct minimum *s-t* cuts $(Y_\lambda, \overline{Y_\lambda})$ in $H_{N(\lambda)}(s, t)$ for parameters λ with $0 \leq \lambda \leq 1$, since $\overline{Y_{\lambda'}} \supseteq \overline{Y_\lambda}$ (i.e., $Y_{\lambda'} \subseteq Y_\lambda$) holds for two distinct parameters $\lambda' < \lambda$ as described above. Suppose that there are exactly K distinct minimum *s-t* cuts $(Y_\lambda, \overline{Y_\lambda})$ in $H_{N(\lambda)}(s, t)$ for parameters λ with $0 \leq \lambda \leq 1$, and let

$$\lambda_0 = 0 \leq \lambda_1 < \lambda_2 < \cdots < \lambda_K \leq \lambda_\infty = 1, \tag{2}$$

where we consider $\lambda_0 = 0$ and $\lambda_\infty = 1$, for convenience.

Fig.8 shows an example of network $H_{N(\lambda)}(s, t)$ corresponding to valuation intervals $\mathcal{C}_N = (C_i : i \in N)$ (and $\mathcal{I}_N = (I_j : 1 \leq j \leq n')$) in Fig.5 and the minimum *s-t* cut $(Y_\lambda, \overline{Y_\lambda})$ in $H_{N(\lambda)}(s, t)$ with $\lambda = \rho_{\min} = 0.1$. Fig.9 shows that the minimum *s-t* cuts $(Y_\lambda, \overline{Y_\lambda})$ in $H_{N(\lambda)}(s, t)$ in Fig.8 for parameters λ with $0 \leq \lambda \leq 1$ form a lower envelope of the arrangement of lines generated by $y = \text{capa}(Y_\lambda, \overline{Y_\lambda}) = \lambda|\overline{Y_\lambda} \cap \mathcal{C}_N| + \sum_{v \in Y_\lambda \cap \mathcal{I}_N} \text{capa}(v, t)$ ($K = 3$ and $\lambda_1 = 0.1 < \lambda_2 = 0.15 < \lambda_3 = 0.2$). Note that there are more minimum *s-t* cuts in $H_{N(\lambda)}(s, t)$, for example, $(Z, \overline{Z})$ with $Z = \{s, C_2, C_5, I_5, I_{10}\}$ and $\overline{Z} = \{t\} + (\mathcal{C}_N \setminus \{C_2, C_5\}) + (\mathcal{I}_N \setminus \{I_5, I_{10}\})$ is a minimum *s-t* cut $H_{N(\lambda)}(s, t)$ with $\lambda = 0.1$ and the corresponding line is $y = \text{capa}(Z, \overline{Z}) = 6\lambda + \text{capa}(I_5, t) + \text{capa}(I_{10}, t) = 6\lambda + 0.2$. Note also that, for finding a lower envelope of the arrangement of lines generated by all the minimum *s-t* cuts in $H_{N(\lambda)}(s, t)$ for parameters λ with $0 \leq \lambda \leq 1$, it is sufficient to consider only all the minimum *s-t* cuts $(Y_\lambda, \overline{Y_\lambda})$ in $H_{N(\lambda)}(s, t)$ defined above.

Suppose that Procedure CutCake($P, D, \mathcal{D}_P$) is called exactly K' times in Mechanism 3.2 for the cake-cutting problem with a cake $C = [0, 1)$, n players $N = \{1, 2, \ldots, n\}$, and solid valuation intervals $\mathcal{C}_N = (C_i : i \in N)$ where $C_i = [\alpha_i, \beta_i) \subseteq C$ is a valuation interval of player $i \in N$, and let $\rho^{(k)}_{\min}$ be the minimum density of interval in the k-th call of CutCake($P, D, \mathcal{D}_P$). Clearly, $\rho^{(1)}_{\min} = \rho_{\min}$ in CutCake($N, C, \mathcal{C}_N$). Furthermore, by Lemma 6 in [4], we have $\rho^{(1)}_{\min} < \rho^{(2)}_{\min} < \cdots < \rho^{(K')}_{\min}$. Thus, we have the following lemma.

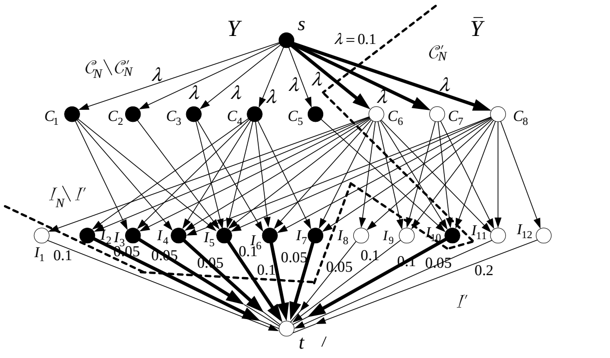

Fig. 8 Example of network $H_{N(\lambda)}(s,t)$ corresponding to the valuation intervals $\mathcal{C}_N = (C_i : i \in N)$ ($N = \{1, 2, \ldots, 8\}$) (and $\mathcal{I}_N = (I_j : 1 \le j \le n')$ ($j = 1, 2, \ldots, 12$)) in Fig.5 and a minimum s-t cut $(Y, \overline{Y}) = (Y_\lambda, \overline{Y_\lambda})$ in $H_{N(\lambda)}(s,t)$ with $\lambda = \rho_{\min} = 0.1$.

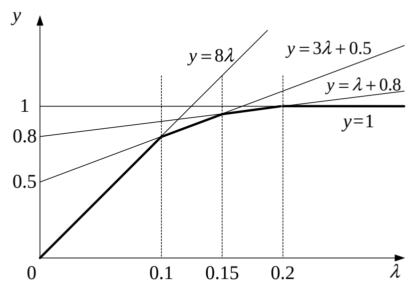

Fig. 9 Minimum s-t cuts $(Y_\lambda, \overline{Y_\lambda})$ in $H_{N(\lambda)}(s,t)$ in Fig.8 for parameters λ with $0 \le \lambda \le 1$ form a lower envelope of the arrangement of lines generated by $y = \mathrm{capa}(Y_\lambda, \overline{Y_\lambda}) = \lambda|\overline{Y_\lambda} \cap \mathcal{C}_N| + \sum_{v \in Y_\lambda \cap \mathcal{I}_N} \mathrm{capa}(v, t)$ ($K = 3$ and $\lambda_1 = 0.1 < \lambda_2 = 0.15 < \lambda_3 = 0.2$).

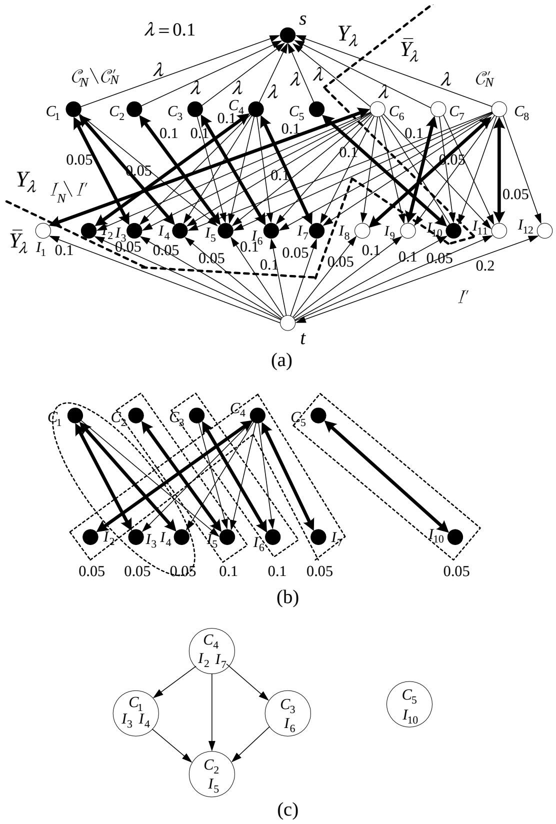

Fig. 10 (a) The residual network $H_{N(\lambda_1)}(s,t)(f_1)$ with respect to a maximum flow f_1 in $H_{N(\lambda_1)}(s,t)$ where $\lambda_1 = 0.1$, (b) the subnetwork $H_{N(\lambda_1)}(f_1)$ of $H_{N(\lambda_1)}(s,t)(f_1)$ induced by $Y_{\lambda_1} \setminus \{s\}$ and (c) the condensed network $H'_{N(\lambda_1)}(f_1)$ of $H_{N(\lambda_1)}(f_1)$.

Lemma 5.1 For the cake-cutting problem with cake $C = [\,0, 1)$, n players $N = \{1, 2, \ldots, n\}$ and solid valuation intervals $\mathcal{C}_N = (C_i : i \in N)$ where $C_i = [\,\alpha_i, \beta_i) \subseteq C$ is a valuation interval of player $i \in N$, and the corresponding network $H_{N(\lambda)}(s,t)$, we have

$$K = K' \quad \text{and} \quad \rho^{(k)}_{\min} = \lambda_k \quad (k = 1, 2, \ldots, K). \tag{3}$$

Proof: To prove $\rho_{\min} = \rho^{(1)}_{\min} = \lambda_1$, we consider a maximum s-t flow f_1 in $H_{N(\lambda_1)}(s,t)$ and let $\overline{Y_{\lambda_1}}$ be the set of vertices v of $H_N(s,t)$ such that there is a path from v to t in the residual network $H_{N(\lambda_1)}(s,t)(f_1)$ with respect to f_1 and let $Y_{\lambda_1} = V_N(s,t) \setminus \overline{Y_{\lambda_1}}$. Then $(Y_{\lambda_1}, \overline{Y_{\lambda_1}})$ is a minimum s-t cut in $H_{N(\lambda_1)}(s,t)$. Let $H_{N(\lambda_1)}(f_1)$ be the subnetwork of $H_{N(\lambda_1)}(s,t)(f_1)$ induced by $Y_{\lambda_1} \setminus \{s\}$, that is, $H_{N(\lambda_1)}(f_1)$ is the network obtained from $H_{N(\lambda_1)}(s,t)(f_1)$ by deleting all the vertices in $\overline{Y_{\lambda_1}} + \{s\}$. We next compute all the strongly connected components $Z_1, Z_2, \ldots, Z_Q$ of $H_{N(\lambda_1)}(f_1)$. Of course, two distinct strongly connected components $Z_q, Z_{q'}$ are vertex-disjoint, i.e., $V(Z_q) \cap V(Z_{q'}) = \emptyset$ for $1 \le q < q' \le Q$ and the set $Y_{\lambda_1} \setminus \{s\}$ is partitioned into $V(Z_1) + V(Z_2) + \cdots + V(Z_Q)$, where $V(Z_q)$ is the vertex set of Z_q for each $q = 1, 2, \ldots, Q$. Then we contract each strongly connected component Z_q into one vertex z_q. The resulting network $H'_{N(\lambda_1)}(f_1)$ is called a *condensed network* of $H_{N(\lambda_1)}(f_1)$. This condensed network is acyclic, that is, it has no directed cycle. Thus, there is a vertex z_q with indegree 0. By symmetry, there is also a vertex $z_{q'}$ with outdegree 0. We denote by $V(z_q)$ the set of vertices $z_{q'}$ such that there is a directed path from z_q to $z_{q'}$ in $H'_{N(\lambda_1)}(f_1)$, and we denote by $\mathcal{V}(z_q)$ be the set of vertices in the strongly connected components $Z_{q'}$ of $H_{N(\lambda_1)}(f_1)$ corresponding to the vertices $z_{q'} \in V(z_q)$. Then each $\mathcal{V}(z_q)$ corresponding to vertex z_q with indegree 0 forms a maximal interval of minimum density $\rho_{\min}$. Similarly, each $\mathcal{V}(z_q)$ corresponding to vertex z_q with outdegree 0 forms a minimal interval of minimum density $\rho_{\min}$. Furthermore, for a vertex subset U of $H'_{N(\lambda_1)}(f_1)$, each connected component in $\bigcup_{z_q \in U} \mathcal{V}(z_q)$ corresponding to vertices $z_q \in U$ forms an interval of minimum density. Thus, $\bigcup_{z_q \in U} \mathcal{V}(z_q)$ forms a disjoint union of intervals of minimum density. Furthermore, let $Z = \{s\} + \bigcup_{z_q \in U} \mathcal{V}(z_q)$. Then $(Z, \overline{Z})$ is a minimum s-t cut in $H_{N(\lambda_1)}(s,t)$.

Since $(\{s\}, V(H_N(s,t)) \setminus \{s\})$ and $(Y_{\lambda_1}, \overline{Y_{\lambda_1}})$ are both minimum s-t cuts in $H_{N(\lambda_1)}(s,t)$, we have $\mathrm{capa}(\{s\}, V(H_N(s,t)) \setminus \{s\}) = n\lambda_1 = \mathrm{capa}(Y_{\lambda_1}, \overline{Y_{\lambda_1}}) = \lambda_1|\overline{Y_{\lambda_1}} \cap \mathcal{C}_N| + \rho_{\min}|Y_{\lambda_1} \cap \mathcal{C}_N|$ and $\rho_{\min} = \lambda_1$.

Thus, if $K = 1$ then the lemma clearly holds. If $K > 1$, then we use an induction on K. Suppose that lemma holds for all $K \le K'' - 1$ ($K'' \ge 2$) and consider when $K = K''$. Since all the maximal intervals $H_1, H_2, \ldots, H_L$ of minimum density $\rho_{\min}$ are deleted and the resulting hollow intervals $H_1, H_2, \ldots, H_L$ are virtually shrunken in the first call CutCake$(N, C, \mathcal{C}_N)$, the second call CutCake$(P, D, \mathcal{D}_P)$ is for cake $D = C'^{(S)}$, players $P = N \setminus \sum_{\ell=1}^{L} N(H_\ell)$ and solid valuation intervals $\mathcal{D}_P$, where D, $D_k \in \mathcal{D}_P$ and $\mathcal{D}_P$ are obtained from C', $C'_k = C_k \setminus \sum_{\ell=1}^{L} H_\ell \in \mathcal{C}'_{N'}$ and $\mathcal{C}'_{N'}$ (which consists of valuations $C'_k = C_k \setminus \sum_{\ell=1}^{L} H_\ell \ne \emptyset$ for all $k \in N'$) by virtually shrinking of all $H_1, H_2, \ldots, H_L$. Due to the shortage of space, we omit details, but we can use the induction hypothesis on the second call CutCake$(P, D, \mathcal{D}_P)$ including all the other calls and obtain that the lemma holds. □

5.2 Finding $\lambda_1, \lambda_2, \ldots, \lambda_K$ in Parametric Flow

To find all $\lambda_1, \lambda_2, \ldots, \lambda_K$, we use a binary search on interval (λ^-, λ^+) to find λ_k with $\lambda^- < \lambda_k < \lambda^+$ based on the method in [7]. We initially set $\lambda^- = 0$, $\lambda^+ = 1$ and $H_{N(\lambda)(\lambda^-,\lambda^+)}(s,t) = H_{N(\lambda)}(s,t)$. Then we find the minimum s-t cut $(Y_{\lambda^-}, \overline{Y_{\lambda^-}})$ in $H_{N(\lambda^-)(\lambda^-,\lambda^+)}(s,t)$ and the minimum s-t cut $(Y_{\lambda^+}, \overline{Y_{\lambda^+}})$ in $H_{N(\lambda^+)(\lambda^-,\lambda^+)}(s,t)$. Let

$y_{\lambda^-}(\lambda) = \lambda|\overline{Y_{\lambda^-}} \cap \mathcal{C}_N| + \sum_{v \in Y_{\lambda^-} \cap \mathcal{I}_N} \mathrm{capa}(v,t)$,

$y_{\lambda^+}(\lambda) = \lambda|\overline{Y_{\lambda^+}} \cap \mathcal{C}_N| + \sum_{v \in Y_{\lambda^+} \cap \mathcal{I}_N} \mathrm{capa}(v,t)$.

Initially, $y_{\lambda^-}(\lambda) = y_0(\lambda) = n\lambda$ and $y_{\lambda^+}(\lambda) = y_1(\lambda) = 1$. Note that,
$y_{\lambda^-}(\lambda^-) = \mathrm{capa}(Y_{\lambda^-}, \overline{Y_{\lambda^-}}) = \lambda^-|\overline{Y_{\lambda^-}} \cap \mathcal{C}_N| + \sum_{v \in Y_{\lambda^-} \cap \mathcal{I}_N} \mathrm{capa}(v,t)$,
$y_{\lambda^+}(\lambda^+) = \mathrm{capa}(Y_{\lambda^+}, \overline{Y_{\lambda^+}}) = \lambda^+|\overline{Y_{\lambda^+}} \cap \mathcal{C}_N| + \sum_{v \in Y_{\lambda^+} \cap \mathcal{I}_N} \mathrm{capa}(v,t)$.
In each iteration, we find λ^* such that $y_{\lambda^-}(\lambda^*) = y_{\lambda^+}(\lambda^*)$ and find the minimum s-t cut $(Y_{\lambda^*}, \overline{Y_{\lambda^*}})$ in $H_{N(\lambda^*)(\lambda^-,\lambda^+)}(s,t)$ and $y_{\lambda^*}(\lambda) = \lambda|\overline{Y_{\lambda^*}} \cap \mathcal{C}_N| + \sum_{v \in Y_{\lambda^*} \cap \mathcal{I}_N} \mathrm{capa}(v,t)$. If $y_{\lambda^*}(\lambda^*) = y_{\lambda^-}(\lambda^*) = y_{\lambda^+}(\lambda^*)$ then $\lambda^* = \lambda_k$ and we stop the binary search on interval (λ^-, λ^+). Otherwise, we continue the binary search on interval (λ^-, λ^*) in the network $H_{N(\lambda)(\lambda^-,\lambda^*)}(s,t)$ obtained from $H_{N(\lambda)(\lambda^-,\lambda^+)}(s,t)$ by deleting $\overline{Y_{\lambda^*}} \setminus \{t\}$ and the binary search on interval (λ^*, λ^+) in the network $H_{N(\lambda)(\lambda^*,\lambda^+)}(s,t)$ obtained from $H_{N(\lambda)(\lambda^-,\lambda^+)}(s,t)$ by deleting $Y_{\lambda^*} \setminus \{s\}$. We denote by $\mathcal{C}_{N(\lambda^-,\lambda^+)}$ the set of vertices $C_i \in \mathcal{C}_N$ which are contained in $H_{N(\lambda)(\lambda^-,\lambda^+)}(s,t)$. Similarly, we denote by $\mathcal{I}_{N(\lambda^-,\lambda^+)}$ the set of vertices $I_j \in \mathcal{I}_N$ which are contained in $H_{N(\lambda)(\lambda^-,\lambda^+)}(s,t)$. We also use $\overline{\mathcal{C}_{N(\lambda^-,\lambda^+)}} = \mathcal{C}_N \setminus \mathcal{C}_{N(\lambda^-,\lambda^+)}$ and $\overline{\mathcal{I}_{N(\lambda^-,\lambda^+)}} = \mathcal{I}_N \setminus \mathcal{I}_{N(\lambda^-,\lambda^+)}$. To find the minimum s-t cut $(Y_{\lambda^*}, \overline{Y_{\lambda^*}})$ in $H_{N(\lambda^*)(\lambda^-,\lambda^+)}(s,t)$, we use the following mechanism.

Mechanism 5.1 Parametric Flow Mechanism.

Input: A cake $C = [\,0,1)$, n players $N = \{1,2,\ldots,n\}$ and solid valuation intervals $\mathcal{C}_N = (C_i : i \in N)$ with valuation interval $C_i = [\,\alpha_i, \beta_i)$ of each player $i \in N$ and $\bigcup_{C_i \in \mathcal{C}_N} C_i = C$.

Output: $(s_i : i \in N)$ such that there is an envy-free allocation $A'_N = (A'_i : i \in N)$ to players N with $A'_i \subseteq C_i$ and $len(A'_i) = s_i$ for each $i \in N$ and $\sum_{i \in N} A'_i = C$.

Algorithm {
Let $X_N = \{x_0, x_1, \ldots, x_{n'}\}$ be the set of all endpoints α_i, β_i of
  $C_i = [\,\alpha_i, \beta_i)$ of $\mathcal{C}_N = (C_i : i \in N)$;
assume $x_0 < x_1 < \cdots < x_{n'}$ by sorting the elements in X_N
  where $x_0 = 0$, $x_{n'} = 1$;
let $I_j = [x_{j-1}, x_j)$ and $t_j = x_j - x_{j-1}$ for each j with $1 \le j \le n'$;
let $\mathcal{I}_N = (I_j : 1 \le j \le n')$;
let $C_1 \le C_2 \le \cdots \le C_n$ in a lexicographic order with respect
  to (β_i, α_i) by sorting valuation intervals $C_i = (\alpha_i, \beta_i) \in \mathcal{C}_N$;
set $\lambda^- = 0$ and $\lambda^+ = 1$;
Consider $H_{N(\lambda)(\lambda^-,\lambda^+)}(s,t)$ with $\lambda^- < \lambda < \lambda^+$;
$K = 0$;
FindMaxFlow($H_{N(\lambda)(\lambda^-,\lambda^*)}(s,t)$);
}

Note that, Parametric Flow Mechanism 5.1 not only correctly finds such $(s_i : i \in N)$ but also finds an envy-free allocation $A_N = (A_i : i \in N)$ with $A_i \subseteq C_i$ and $len(A_i) = s_i$ for each $i \in N$ and $\sum_{i \in N} A_i = C$. We can also show that Mechanism 5.1 is envy-free and truthful by the argument in [3]. It can be implemented to run in $O(n^2 \log n)$ time with union-split-find data structures by considering $H_{N(\lambda)(\lambda^-,\lambda^+)}(s,t)$ implicitly. We omit the details.

Procedure 5.1 FindMaxFlow($H_{N(\lambda)(\lambda^-,\lambda^*)}(s,t)$) {
find λ^* such that $y_{\lambda^-}(\lambda^*) = y_{\lambda^+}(\lambda^*)$;
let $\mathcal{C}_{N(\lambda^-,\lambda^+)} = (C_{i_1}, C_{i_2}, \cdots, C_{i_p})$ with $i_1 < i_2 < \ldots < i_p$;
$A_{i_0} = \emptyset$;
for $j = 1$ **to** p **do**
  let $Z = \sum_{j'=0}^{j-1} A_{i_{j'}} + \sum_{I \in \overline{\mathcal{I}_{N(\lambda^-,\lambda^+)}}} I$;
  set $A_{i_j} = [a_{i_j}, b_{i_j}) \setminus Z \subseteq C_{i_j} \setminus Z$ of length $\min\{\lambda^*, len(C_{i_j} \setminus Z)\}$
    where a_{i_j} is the minimum endpoint in $C_{i_j} \setminus Z$;
  $f(s, C_{i_j}) = len(A_{i_j})$;
let $A = \sum_{j=1}^{p} A_{i_j}$;
for each $I \in \mathcal{I}_{N(\lambda^-,\lambda^+)}$ **do** $f(I,t) = len(A \cap I)$;
for each edge (C_{i_j}, I) in $H_{N(\lambda)(\lambda^-,\lambda^*)}(s,t)$ **do**
  $f(C_{i_j}, I) = len(A_{i_j} \cap I)$;
let $H_{N(\lambda^*)(\lambda^-,\lambda^+)}(s,t)(f)$ be residual network with respect to f;
let $\overline{Y_{\lambda^*}}$ be the set of vertices v of $H_{N(\lambda^*)(\lambda^-,\lambda^+)}(s,t)(f)$ such
  that there is a path from v to t in $H_{N(\lambda^*)(\lambda^-,\lambda^+)}(s,t)(f)$;
let Y_{λ^*} be the set of vertices v of $H_{N(\lambda^*)(\lambda^-,\lambda^+)}(s,t)(f)$
  not contained in $\overline{Y_{\lambda^*}}$;
let $y_{\lambda^*}(\lambda) = \lambda|\overline{Y_{\lambda^*}} \cap \mathcal{C}_N| + \sum_{v \in Y_{\lambda^*} \cap \mathcal{I}_N} \mathrm{capa}(v,t)$;
if $y_{\lambda^*}(\lambda^*) = y_{\lambda^-}(\lambda^*) = y_{\lambda^+}(\lambda^*)$ **then**
  $K = K + 1$; $\lambda^* = \lambda_K$;
  for each C_{i_j} in $H_{N(\lambda)(\lambda^-,\lambda^*)}(s,t)$) **do** $s_{i_j} = \lambda_K$;
else
  FindMaxFlow($H_{N(\lambda)(\lambda^-,\lambda^*)}(s,t)$) where $H_{N(\lambda)(\lambda^-,\lambda^*)}(s,t)$ is
    obtained from $H_{N(\lambda)(\lambda^-,\lambda^+)}(s,t)$ by deleting $\overline{Y_{\lambda^*}} \setminus \{t\}$;
  FindMaxFlow($H_{N(\lambda)(\lambda^*,\lambda^+)}(s,t)$) where $H_{N(\lambda)(\lambda^*,\lambda^+)}(s,t)$ is
    obtained from $H_{N(\lambda)(\lambda^-,\lambda^+)}(s,t)$ by deleting $Y_{\lambda^*} \setminus \{s\}$.
}

References

[1] H. Aissi, S.T. McCormick, and M. Queyranne, Faster algorithms for next breakpoint and max value for parametric global minimum cuts, *Proc. of 21st International Conference on Integer Programming and Combinatorial Optimization*, pp.27–39, 2020.

[2] R. Alijani, M. Farhadi, M. Ghodsi, M. Seddighin, and A.S. Tajik, Envy-free mechanisms with minimum number of cuts, *Proc. of 31st AAAI Conference on Artificial Intelligence*, pp.312–318, 2017.

[3] T. Asano and H. Umeda, An envy-free and truthful mechanism for the cake-cutting problem, *RIMS Kôkyûroku 2154*, Kyoto University, April, pp.54–91, 2020.

[4] T. Asano and H. Umeda, Cake Cutting: An envy-free and truthful mechanism with a small number of cuts, *31st International Symposium on Algorithms and Computation (ISAAC 2020)*, pp.15.1–15.16, 2020.

[5] H. Aziz and C. Ye, Cake cutting algorithms for piecewise constant and piecewise uniform valuations, *Proc. of 10th International Conference on Web and Internet Economics*, pp.1–14, 2014.

[6] Y. Chen, J. K. Lai, D. C. Parkes, and A. D. Procaccia, Truth, justice, and cake cutting, *Games and Economic Behavior*, Vol.77, pp.284–297, 2013.

[7] G. Gallo, M.D. Grigoriadis, and R.E. Tarjan, A fast parametric maximum flow algorithm and applications, *SIAM Journal on Computing*, Vol. 18, pp.30–55, 1989.

[8] M. Seddighin, M. Farhadi, M. Ghodsi, R. Alijani, and A.S. Tajik, Expand the shares together: envy-free mechanisms with a small number of cuts, *Algorithmica*, Vol. 81, pp.1728–1755, 2019.

[9] H. Steinhaus, The problem of fair division, *Econometrica*, Vol. 16, pp.101–104, 1948.

[10] T. Tokuyama, Minimax parametric optimization problems and multidimensional parametric searching, *Proc. of 33rd ACM Symposium on Theory of Computing*, pp.75–83, 2001.