TITLE:

Envy-Free and Truthful Cake-Cuttings Based on Parametric Flows (New Trends in Algorithms and Theory of Computation)

AUTHOR(S):

Asano, Takao

CITATION:

Asano, Takao. Envy-Free and Truthful Cake-Cuttings Based on Parametric Flows (New Trends in Algorithms and Theory of Computation). 数理解析研究所講究録 2021, 2182: 1-39

ISSUE DATE:

2021-04

URL:

http://hdl.handle.net/2433/264881

RIGHT:

京都大学学術情報リポジトリ
KURENAI 紅
Kyoto University Research Information Repository

Envy-Free and Truthful Cake-Cuttings Based on Parametric Flows

Takao Asano [*]
Chuo University

Abstract

For the cake-cutting problem, Alijani, et al. [2, 30] and Asano and Umeda [3, 4] gave envy-free and truthful mechanisms with a small number of cuts, where the desired part of each player's valuation function is a single interval on the given cake. In this paper, based on parametric flows, we give efficient envy-free and truthful mechanisms with a small number of cuts, which are much simpler than those proposed by Alijani, et al. [2, 30] and Asano and Umeda [3, 4]. Furthermore, we show that this approach can be applied to the envy-free and truthful mechanism proposed by Chen, et al. [16], 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.

1 Introduction

The problem of dividing a cake among players in a fair manner has attracted the attention of mathematicians, economists, political scientists and computer scientists [6, 7, 14, 16, 17, 18, 19, 27, 28, 29] since it was first considered by Banach and Knaster [14] and Steinhaus [32, 33]. The cake-cutting problem is often used as a metaphor for prominent real-world problems that involve the division of a heterogeneous divisible good [12]. Some of examples include allocating staff to time-intensive tasks such as scheduling police patrol operations and allocating of cleaning tasks to maintenance crews [17, 37]. Territory-splitting applications are also discussed by Thomson and Sherstyuk based on fair cake-cutting approaches [31, 37].

Slightly more formally, the cake-cutting problem is stated as follows [16]: Given a divisible heterogeneous cake C represented by an interval $[0, 1)$ and n strategic players $N = \{1, 2, \ldots, n\}$, where each player i has a valuation function v_i over the cake C, divide the cake C and find an allocation of the cake C to the players that satisfies one or several fairness criteria. In the cake cutting literature, the most important criteria are *envy-freeness* and *proportionality* [6]. In an envy-free allocation, each player considers her/his allocation at least as good as any other player's allocation. In a proportional allocation, each player gets at least $\frac{1}{n}$ of the value she/he assigns to the cake C. An envy-free allocation is a proportional allocation when every portion of the cake that is desired at least one player is allocated to some player.

[*] The author would like to thank Professor Shigeo Tsujii of Research and Development Initiative, Chuo University. This work was supported by the Research Institute for Mathematical Sciences, an International Joint Usage/Research Center located in Kyoto University. E-mail: asano@ise.chuo-u.ac.jp

A *piece* A of cake C is a finite union of disjoint subintervals X of C. A piece A can also be viewed as a set of disjoint subintervals X of C. For a general valuation function v_i of player i which is integrable or piecewise continuous, the value $V_i(A)$ of a piece A of cake C for player i can be written by $\int_{x \in A} v_i(x)dx$. Thus, the value $V_i(A)$ of the piece A of disjoint subintervals X of C for player i is $V_i(A) = \sum_{X \in A} V_i(X)$.

Since general valuation functions may not have a finite discrete representation as an input to the cake-cutting problem, most algorithms and computational complexity analyses are based on oracle computation models. Among them a most popular computation model for general integrable valuation functions is the Robertson-Webb model based on two types of queries: evaluation and cut [29].

In the Roberson-Webb model, Even and Paz proposed a proportional cake cutting algorithm with $O(n \log n)$ queries that outputs a contiguous interval (a piece with a single interval) allocation to each player [20] and Edmonds and Pruhs proved that any proportional cake cutting algorithm, even if it is allowed to output an allocation consisting of several disjoint intervals to each player, requires $\Omega(n \log n)$ queries [18].

For envy-freeness, Stromquist showed that there is no finite envy-free cake cutting algorithm that outputs a contiguous allocation to each player for any $n \geq 3$ [28, 35], although an envy-free allocation with a contiguous interval allocation to each player is guaranteed to exist [34, 36]. Note that any cake cutting algorithm that outputs a contiguous allocation to each player uses $n-1$ cuts on the cake C. If a contiguous allocation to each player is not required, Aziz and Mackenzie showed that there is an envy-free cake cutting algorithm with $O(n^{n^{n^{n^{n^{n}}}}})$ queries [7]. Procaccia showed that any envy-free cake cutting algorithm requires $\Omega(n^2)$ queries in the Roberson-Webb model [26]. Furthermore, Deng, Qi and Saberi showed that finding an envy-free allocation using $n-1$ cuts on cake C is PPAD-complete when valuation functions are given explicitly by polynomial-time algorithms [17], although their result requires very general (e.g., non-additive, non monotone) valuation functions [22].

In recent papers, some restricted classes of valuation functions have been studied [6, 9, 12, 15, 16, 25]. Piecewise uniform and piecewise constant valuation functions are two special classes of valuation functions [2, 6, 16, 30]. For a nonnegative valuation function v on cake C, let $D(v) = \{x \in C \mid v(x) > 0\}$. Thus, we can consider that $D(v)$ consists of several disjoint maximal contiguous intervals. Then v is called *piecewise uniform* if $v(x) = v(y)$ holds for all $x, y \in D(v)$. Similarly, v is called *piecewise constant* if, for each contiguous interval I in $D(v)$, $v(x') = v(x'')$ holds for all $x', x'' \in I$. Note that $v(x) \neq v(y)$ may hold for $x \in I$ and $y \in J$ when I, J are two distinct maximal contiguous intervals in $D(v)$ of piecewise constant valuation v. Thus, a piecewise uniform valuation is always a piecewise constant valuation. One of the most important properties of these valuation functions is that they can be described concisely. Kurokawa, Lai, and Procaccia proved that finding an envy-free allocation in the Robertson-Webb model when the valuation functions are piecewise uniform is as hard as solving the problem without any restriction on the valuation functions [24].

The cake-cutting problem has been studied not only from the viewpoint of computational complexity but also from the game theoretical point of view [2, 6, 9, 16, 25, 30]. Chen, Lai, Parkes, and Procaccia considered a strong notion of truthfulness (denoted by strategy-proofness), in which the players' dominant strategies are to reveal their true valuations over the cake [16]. They presented an envy-free and truthful mechanism for the cake-cutting problem based on maximum flow and minimum cut techniques [39] when the valuation functions are piecewise uniform. Aziz and Ye considered the problem when valuation functions are piecewise constant and piecewise uniform [6]. Based on parametric network flows [21], random assignments and probabilistic serial algorithms [5, 11], they designed three algorithms called CCEA (Controlled Cake Eating Algorithm), MEA (Market Equilibrium Algorithm) and CSD (Constrained Serial Dictatorship Algorithm) with nice properties for piecewise constant valuations, which partially solve an open problem for piecewise constant valuations posed by Chen et al. in [16]. They showed that CCEA runs in $O(n^5M^2\log(\frac{n^2}{M}))$, where ($n$ is the number of players and) M is the number of subintervals defined by the union of discontinuity points of the players' piecewise constant valuations ($M \leq 2\sum_{i\in N} m_i$ where m_i is the number of maximal contiguous intervals in $D(v_i) = \{x \in C \mid v_i(x) > 0\}$ of piecewise constant valuation v_i). They also showed that, when CCEA and MEA are restricted for piecewise uniform valuations, CCEA and MEA become essentially the same as the mechanism in [16] (as mentioned above, a piecewise uniform valuation is always a piecewise constant valuation). Note that, however, CCEA, MEA and the mechanism in [16] for dividing the cake use $\Omega(nM)$ cuts [2, 30], where $M \leq 2\sum_{i\in N} m_i$ and m_i is the number of maximal contiguous intervals in $D(v_i) = \{x \in C \mid v_i(x) > 0\}$ of piecewise uniform valuation v_i as mentioned above.

Alijani, Farhadi, Ghodsi, Seddighin, and Tajik [2, 30] considered that the number of cuts is important and considered the following cake-cutting problem by requiring $D(v_i) = \{x \in C \mid v_i(x) > 0\}$ of piecewise uniform valuation v_i of each player i to be 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 [2, 30]. Asano and Umeda [3, 4] also gave an alternative envy-free and truthful mechanism with at most $2n-2$ cuts, by pointing out that their original mechanism in [2, 30] is not actually envy free.

In this paper, based on parametric flows, we give efficient envy-free and truthful mechanisms with a small number of cuts, which lead to a much simpler mechanism than those proposed by Alijani, et al. [2, 30] and Asano and Umeda [3, 4]. Thus, we can obtain a much simpler envy-free and truthful mechanism with at most $2n-2$ cuts which runs in $O(n^2\log n)$ time 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, et al. for the more general cake-cutting problem where the valuation function of each player is piecewise uniform [16]. Thus, this approach can make their envy-free and truthful mechanism use $2M-2$ cuts, where $M \leq 2\sum_{i\in N} m_i$ and m_i is the number of maximal contiguous intervals in $D(v_i) = \{x \in C \mid v_i(x) > 0\}$ of piecewise uniform valuation v_i as mentioned above.

2 Preliminaries

We are given a divisible heterogeneous cake $C = [0, 1) = \{x \mid 0 \leq x < 1\}$ [1], n players $N = \{1, 2, \ldots, n\}$ with valuation interval $C_i = [\alpha_i, \beta_i) = \{x \mid 0 \leq \alpha_i \leq x < \beta_i \leq 1\} \subseteq C$

[1] We assume, $C = [0, 1) = \{x \mid 0 \leq x < 1\}$, and, 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 $X'' = [y, x'')$.

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 write $\mathcal{C}_N = (C_i : i \in N)$. Valuation intervals $\mathcal{C}_N$ is called *solid*, if, for every $x \in C$, there is a player $i \in N$ whose valuation interval $C_i \in \mathcal{C}_N$ contains x. As in [2, 6, 4, 30], we will assume that $\mathcal{C}_N$ is solid, i.e., $\bigcup_{C_i \in \mathcal{C}_N} C_i = C$, throughout this paper.

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, in allocation $A_N = (A_i : i \in N)$ of C to n players N, $\sum_{i \in N} A_i = C$ holds 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 (i.e., a polynomial-time algorithm in this paper) 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 $A_N = (A_i : i \in N)$ for every input $\mathcal{C}_N = (C_i : i \in N)$ to $\mathcal{M}$ is always 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 $A'_N(i) = (A'_j : j \in N)$. 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

$$V_i(A_i) = \textstyle\sum_{\ell=1}^{k_i} len(A_{i_\ell} \cap C_i) \text{ and } V_i(A'_i) = \sum_{\ell=1}^{k'_i} len(A'_{i_\ell} \cap C_i)$$

(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 there is no merit for player i to give false C'_i and player i will report true valuation interval C_i to $\mathcal{M}$. For each player $i \in N$, if this holds for every input $\mathcal{C}_N = (C_i : i \in N)$ to $\mathcal{M}$, then $\mathcal{M}$ is called *truthful* (allocation $A_N = (A_i : i \in N)$ obtained by $\mathcal{M}$ is also called *truthful*).

For 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 with valuation interval C_i contained in X and let $\mathcal{C}_{N(X)}$ be the (multi-) set of valuation intervals in $\mathcal{C}_N$ which are contained in X. Thus,

$$N(X) = \{i \in N \mid C_i \subseteq X, C_i \in \mathcal{C}_N\} \text{ and } \mathcal{C}_{N(X)} = (C_i \in \mathcal{C}_N : i \in N(X)).$$

Let $n_X = |N(X)|$. The *density* of interval $X = [x', x'')$ of C, denoted by $\rho(X)$, is defined by

$$\rho(X) = \frac{len(X)}{|N(X)|} = \frac{x'' - x'}{n_X}. \tag{1}$$

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 $N(X)$. 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. A *minimal interval of minimum density* is similarly defined.

3 Core Mechanism $\mathcal{M}_1$

In this section, we give the core mechanism $\mathcal{M}_1$ which can be applied to the envy-free and truthful mechanism proposed by Chen, et al. [16] 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 $\sum_{i \in N} s_i = 1$). Note that there is no need to have such an allocation $A'_N = (A'_i : i \in N)$ in hand.

Then the core mechanism $\mathcal{M}_1$ can be written as follows.

ALGORITHM 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 $i \in N$
        (thus $\bigcup_{C_i \in \mathcal{C}_N} C_i = C$) and $(s_i : i \in N)$ such that there is an allocation
        $A'_N = (A'_i : i \in N)$ to players $N$ with $A'_i \subseteq C_i$ and $len(A'_i) = s_i$ 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 $\sum_{i \in N} A_i = C$.
sort $\mathcal{C}_N = (C_i : i \in N)$ in a lexicographic order with respect to $(\beta_i, \alpha_i)$ 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 $A_i = [a_i, b_i) \setminus \sum_{i'=0}^{i-1} A_{i'}$ with length $s_i$ such that $[a_i, b_i) \subseteq C_i$ and $a_i$ is the
        leftmost endpoint in $C_i \setminus \sum_{i'=0}^{i-1} A_{i'}$;
```

Figure 1 shows an 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 $\mathcal{M}_1$.

We have the following theorem.

Theorem 3.1 $\mathcal{M}_1$ 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 $\sum_{i \in N} A_i = C$ in $O(n \log n)$ time. Furthermore, the number of cuts made by $\mathcal{M}_1$ on cake C is at most $2n - 2$.

Proof: The number of cuts made on cake C is clearly 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 $0, 1$ of cake $C = [0, 1)$. Similalrly, it can be easily shown that $\mathcal{M}_1$ runs in $O(n \log n)$ time, since lexicographical sorting of $\mathcal{C}_N = (C_i : i \in N)$ requires $O(n \log n)$ time and a_i, b_i for each $i \in N$ can be found in $O(\log n)$ time based on appropriate data structures, for example, union-find-split data structures (Figure 1(c)).

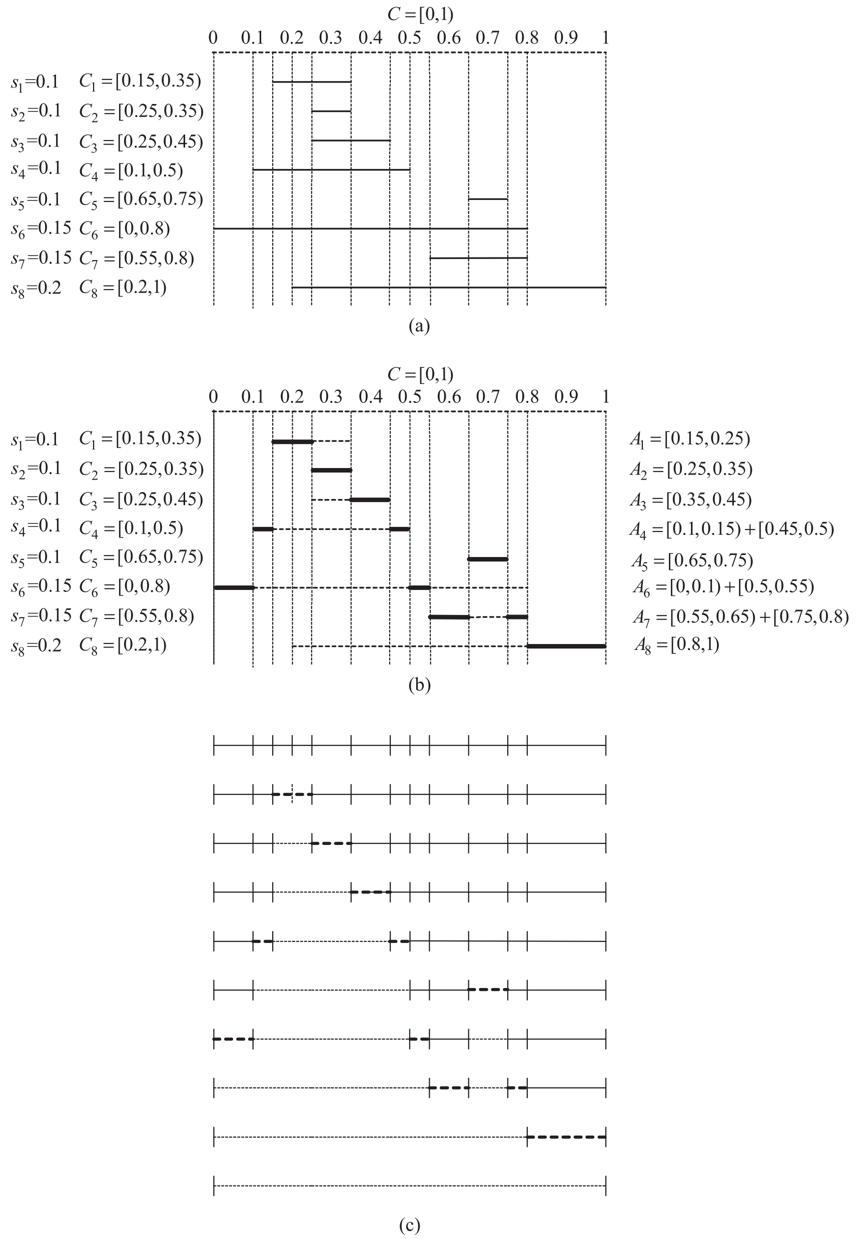

$C=[0,1)$
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
s_1=0.1 $C_1=[0.15,0.35)$
s_2=0.1 $C_2=[0.25,0.35)$
s_3=0.1 $C_3=[0.25,0.45)$
s_4=0.1 $C_4=[0.1,0.5)$
s_5=0.1 $C_5=[0.65,0.75)$
s_6=0.15 $C_6=[0,0.8)$
s_7=0.15 $C_7=[0.55,0.8)$
s_8=0.2 $C_8=[0.2,1)$
(a)
$C=[0,1)$
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
s_1=0.1 $C_1=[0.15,0.35)$
s_2=0.1 $C_2=[0.25,0.35)$
s_3=0.1 $C_3=[0.25,0.45)$
s_4=0.1 $C_4=[0.1,0.5)$
s_5=0.1 $C_5=[0.65,0.75)$
s_6=0.15 $C_6=[0,0.8)$
s_7=0.15 $C_7=[0.55,0.8)$
s_8=0.2 $C_8=[0.2,1)$
$A_1=[0.15,0.25)$
$A_2=[0.25,0.35)$
$A_3=[0.35,0.45)$
$A_4=[0.1,0.15)+[0.45,0.5)$
$A_5=[0.65,0.75)$
$A_6=[0,0.1)+[0.5,0.55)$
$A_7=[0.55,0.65)+[0.75,0.8)$
$A_8=[0.8,1)$
(b)
(c)

Figure 1: (a) Example of $\mathcal{C}_N = (C_i : i \in N)$ 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 $\mathcal{M}_1$. (c) Maintaining of intervals by union-find-split data structures (the set of thick dotted intervals is allocated to a player in the current iteration).

We next give a proof on the proposition that $\mathcal{M}_1$ correctly finds an allocation $A_N = (A_i : i \in N)$ with $A_i \subseteq C_i$, $s_i = len(A_i)$ and $\sum_{i\in N} A_i = C$.

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 s_i for some $i \in N$. Let j be the minimum among such is and let $J = \{1, 2, \ldots, j\}$. Of course, $j > 1$, since we assumed that there is an allocation $A'_N = (A'_i : i \in N)$ to players N with $A'_i \subseteq C_i$ and $len(A'_i) = s_i$ for each $i \in N$ and $\sum_{i\in N} A'_i = C$ (thus $C_1 = [\alpha_1, \beta_1)$ is of length at least s_1 and $A_1 = [a_1, b_1) = [\alpha_1, \alpha_1 + s_1) \subseteq C_1$). 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 $\beta_i \le \beta_j$, since $\mathcal{C}_N = (C_i : i \in N)$ was sorted in the lexicographic order with respect to (β_i, α_i). 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 s_i 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 s_j. This implies that

$$C_j \setminus \textstyle\sum_{i'=0}^{j-1} A_{i'} \text{ is of length } s'_j < s_j$$

and $C_j \setminus \sum_{i'=0}^{j-1} A_{i'} = [a_j, \beta_j) \setminus \sum_{i'=0}^{j-1} A_{i'}$, since $\beta_i \le \beta_j$ and if $\beta_i = \beta_j$ then $\alpha_i \le \alpha_j$ for each $i \in J$. Let

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

Thus, $\sum_{i\in J} A''_i$ of allocation $(A''_i : i \in J)$ consists of several maximal contiguous intervals. Let $I = [a, b)$ be the rightmost maximal contiguous interval among the maximal contiguous intervals in $\sum_{i\in J} A''_i$ (Figure 2). Thus, $b = \beta_j$. Define $K \subseteq J$ by

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

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, which can be obtained as follows.

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

Now suppose that there were $i \in K \setminus \{j\}$ such that $C_i \in \mathcal{C}_K$ is not contained in I. Thus, $I = [a, b)$ is a proper subinterval of $[0, b) = [0, \beta_j)$ (i.e., $a > 0$) and $C_i = [\alpha_i, \beta_i) \in \mathcal{C}_K \setminus \{C_j\}$ contains a point x in $[0, b) \setminus I = [0, a)$. Let $k \in K \setminus \{j\}$ be the minimum among such is and let x_k be a point of $C_k = [\alpha_k, \beta_k) \in \mathcal{C}_K$ contained in $[0, a) = [0, b) \setminus I$. Note that $C_k \cap I \supseteq A''_k \cap I \neq \emptyset$ since $k \in K \setminus \{j\} \subseteq J \setminus \{j\}$. Thus, $\beta_k \le \beta_j$ and $\alpha_k \le x_k < a \le a'_k < \beta_k$ for some $a'_k \in A''_k \cap I \neq \emptyset$. Furthermore, since we chose $I = [a, b) \neq [0, b)$ as the rightmost maximal contiguous interval among the maximal contiguous intervals in $\sum_{i\in J} A''_i$, we have $\sum_{i\in J} A''_i \neq [0, b) = [0, \beta_j)$. Let $I' = [a', a)$ be the rightmost maximal contiguous interval in $[0, b) \setminus \sum_{i\in J} A''_i$ (Figure 2).

Since $C_k = [\alpha_k, \beta_k)$ is a contiguous interval and satisfies $\alpha_k \le x_k < a \le a'_k < \beta_k$, we can assume $x_k \in I' \cap C_k \neq \emptyset$. Thus, $x_k \notin A''_k$ by $I' \cap A''_k \subseteq I' \cap \sum_{i\in J} A''_i = \emptyset$. 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 $A''_k = A_k = [a_k, b_k) \setminus \sum_{i=0}^{k-1} A''_i \subseteq C_k$ with length s_k such that a_k is the leftmost endpoint

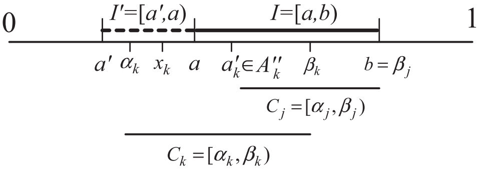


Figure 2: Illustration of $I = [a, b)$ and $I' = [a', a)$.

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 $A''_h \cap I = \emptyset$ for $h \in J \setminus K$ and

$$I = \sum_{i\in J} A''_i \cap I = \sum_{i\in K} A''_i \cap I \subseteq \sum_{i\in K} A''_i \subseteq \sum_{i\in K} C_i$$

by the definitions of I and K and $A''_i \subseteq C_i$ for each $i \in K$. Thus,

$$\sum_{i\in K} len(A''_i) = s'_j + \sum_{i\in K\setminus\{j\}} s_i = len(I) = b - a < s_j + \sum_{i\in K\setminus\{j\}} s_i$$

since $s'_j < s_j$. However, this is a contradiction, since we assumed that there is an allocation $A'_N = (A'_i : i \in N)$ to players N with $A'_i \subseteq C_i$ and $s_i = len(A'_i)$ for each $i \in N$ and $\sum_{i\in N} A'_i = C$, and thus

$$s_j + \sum_{i\in K\setminus\{j\}} s_i = \sum_{i\in K} len(A'_i) \leq len(\bigcup_{i\in K} C_i) = len(I) = b - a.$$

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

By Theorem 3.1, 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 $\sum_{i\in N} A'_i = C$.

4 Flow Network on Valuation Intervals

In this section, we consider a flow network arising from the cake-cutting problem with cake $C = [0, 1)$, n players $N = \{1, 2, \ldots, n\}$ and solid valuation intervals $\mathcal{C}_N = (C_i : i \in N)$ with valuation interval $C_i = [\alpha_i, \beta_i)$ of each player $i \in N$ and $\bigcup_{C_i\in\mathcal{C}_N} C_i = C$. Similar flow networks are given by Athanassoglou and Sethuraman [5] and Chen, et al. [16]. Actullay, the flow network in this section is the same as their flow networks when they are applied to the above cake-cutting problem. By this flow network, we will be able to obtain $(s_i : i \in N)$ such that there is an envy-free allocation $A'_N = (A'_i : i \in N)$ to players

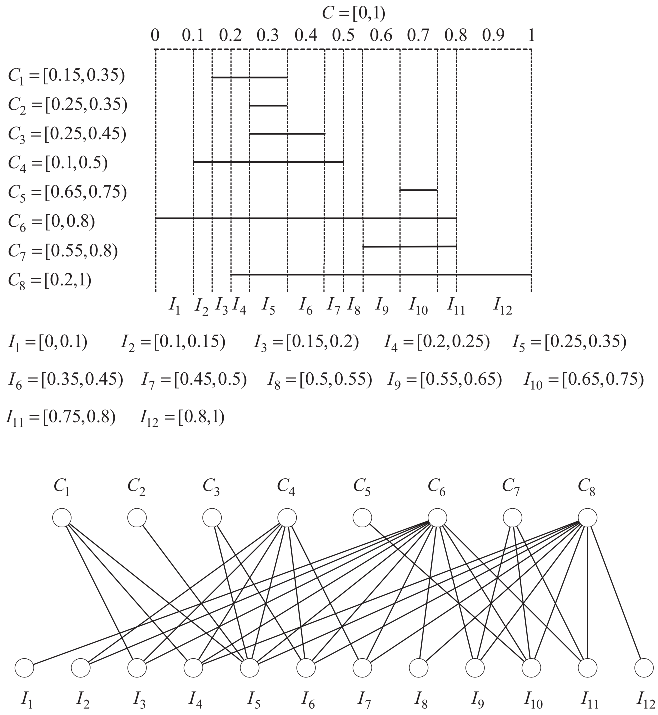

C = [0,1)
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
C1 = [0.15,0.35)
C2 = [0.25,0.35)
C3 = [0.25,0.45)
C4 = [0.1,0.5)
C5 = [0.65,0.75)
C6 = [0,0.8)
C7 = [0.55,0.8)
C8 = [0.2,1)
I1 I2 I3 I4 I5 I6 I7 I8 I9 I10 I11 I12
I1 = [0,0.1) I2 = [0.1,0.15) I3 = [0.15,0.2) I4 = [0.2,0.25) I5 = [0.25,0.35)
I6 = [0.35,0.45) I7 = [0.45,0.5) I8 = [0.5,0.55) I9 = [0.55,0.65) I10 = [0.65,0.75)
I11 = [0.75,0.8) I12 = [0.8,1)
C1 C2 C3 C4 C5 C6 C7 C8
I1 I2 I3 I4 I5 I6 I7 I8 I9 I10 I11 I12

Figure 3: Example of solid valuation intervals $\mathcal{C}_N = (C_i : i \in N)$ $(N = \{1, 2, \ldots, 8\})$, $\mathcal{I}_N = (I_\ell : 1 \leq \ell \leq m)$ $(\ell = 1, 2, \ldots, 12)$ and the convex bipartite graph $G_N = (\mathcal{C}_N, \mathcal{I}_N, E_N)$.

N with $A'_i \subseteq C_i$ and $len(A'_i) = s_i$ for each $i \in N$ and $\sum_{i \in N} A'_i = C$, and thus we can apply Theorem 3.1.

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

$$x_0 < x_1 < \cdots < x_m \tag{2}$$

where $x_0 = 0$, $x_m = 1$ and $m \leq 2n - 1$. For each ℓ with $1 \leq \ell \leq m$, let $I_\ell = [x_{\ell-1}, x_\ell)$ and let $\mathcal{I}_N = (I_\ell : 1 \leq \ell \leq m)$. 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 $(C_i, I_\ell) \in E_N$ if and only if $I_\ell \subseteq C_i$ (Figure 3). $G_N = (\mathcal{C}_N, \mathcal{I}_N, E_N)$ is called a *convex bipartite graph* since it has a property that if $(C_i, I_\ell), (C_i, I_{\ell'}) \in E_N$ with $\ell < \ell'$ then $(C_i, I_{\ell''}) \in E_N$ for each ℓ'' with $\ell < \ell'' < \ell'$.

Let $G_N(s,t)$ be the directed graph obtained from $G_N = (\mathcal{C}_N, \mathcal{I}_N, E_N)$ by adding new vertices s, t and directed edges (s, C_i) $(i \in N)$ and (I_ℓ, t) $(\ell = 1, \ldots, m)$. We consider each edge $(C_i, I_\ell) \in E_N$ is directed from C_i to I_ℓ. Then the flow network $H_{N(\lambda)}(s,t)$ is obtained from $G_N(s,t)$ by defining the capacity capa(e) of each directed edge e of $G_N(s,t)$ as follows. Each directed edge (s, C_i) $(i \in N)$ has capacity λ with parameter $0 \leq \lambda \leq 1$, each directed edge (I_ℓ, t) $(\ell = 1, \ldots, m)$ has capacity $len(I_\ell)$, and each directed edge $(C_i, I_\ell) \in E_N$ has capacity ∞ (Figure 4). We denote by $V_N(s,t)$ and $E_N(s,t)$ the set of all vertices and the

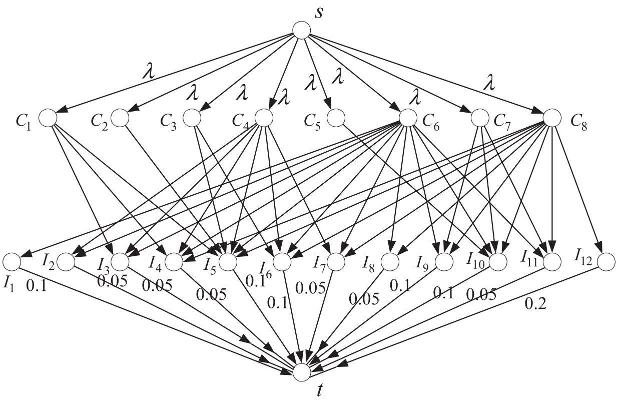


Figure 4: Example of network $H_{N(\lambda)}(s,t)$ corresponding to the valuation intervals $\mathcal{C}_N = (C_i : i \in N)$ $(N = \{1, 2, \ldots, 8\})$ (and $\mathcal{I}_N = (I_\ell : 1 \le \ell \le m)$ $(\ell = 1, 2, \ldots, 12)$) in Figure 3.

set of all directed edges in $H_{N(\lambda)}(s,t)$, respectively. Thus,

$$\begin{aligned} V_N(s,t) &= V_N + \{s,t\} = \mathcal{C}_N + \mathcal{I}_N + \{s,t\}, \\ E_N(s,t) &= E_N + \{(s, C_i) \mid C_i \in \mathcal{C}_N\} + \{(I_\ell, t) \mid I_\ell \in \mathcal{I}_N\}. \end{aligned} \tag{3}$$

A function $f : E_N(s,t) \to \mathbf{R}_+$ is called an *s-t flow* in $H_{N(\lambda)}(s,t)$ with parameter $0 \le \lambda \le 1$, (later f is also called a *parametric* s-t flow and denoted by f_λ since it is associated with parameter λ) if (i) and (ii) hold:

(i) $0 \le f(s, C_i) \le \mathrm{capa}(s, C_i) = \lambda$ for each edge (s, C_i) and $0 \le f(I_\ell, t) \le \mathrm{capa}(I_\ell, t) = len(I_\ell)$ for each edge (I_ℓ, t), and

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

The *value* of an s-t flow f in $H_{N(\lambda)}(s,t)$, denoted by $\mathrm{val}(f)$, is defined by

$$\mathrm{val}(f) = \sum_{C_i \in \mathcal{C}_N} f(s, C_i). \tag{4}$$

Clearly, $\mathrm{val}(f) = \sum_{I_\ell \in \mathcal{I}_N} f(I_\ell, t))$ by the above condition (ii). An s-t flow f in $H_{N(\lambda)}(s,t)$ is called *maximum* if $\mathrm{val}(f) \ge \mathrm{val}(f')$ for all s-t flows f' in $H_{N(\lambda)}(s,t)$. 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 $H_{N(\lambda)}(s,t)$ if $s \in Y$ and $t \in \overline{Y}$. We also call an edge set

$$E(Y, \overline{Y}) = \{e = (y, y') \in E_N(s,t) \mid \ y \in Y, \ y' \in \overline{Y}\}$$

the s-t cut in $H_{N(\lambda)}(s,t)$ defined by s-t cut $(Y, \overline{Y})$. The *capacity* of an s-t cut $(Y, \overline{Y})$ in $H_{N(\lambda)}(s,t)$, denoted by $\mathrm{capa}(Y, \overline{Y})$, is defined by the sum of the capacities $\mathrm{capa}(e)$ of all edges $e = (y, y') \in E_N(s,t)$ with $y \in Y$ and $y' \in \overline{Y}$. That is,

$$\mathrm{capa}(Y, \overline{Y}) = \sum_{e=(y,y') \in E_N(s,t):\ y \in Y,\ y' \in \overline{Y}} \mathrm{capa}(e) \tag{5}$$

(i.e., $\mathrm{capa}(Y,\overline{Y}) = \sum_{e\in E(Y,\overline{Y})} \mathrm{capa}(e)$). An s-t cut $(Y,\overline{Y})$ in $H_{N(\lambda)}(s,t)$ is called *minimum* if $\mathrm{capa}(Y,\overline{Y}) \leq \mathrm{capa}(Y',\overline{Y'})$ for all s-t cuts $(Y',\overline{Y'})$ in $H_{N(\lambda)}(s,t)$. For any s-t flow f and any s-t cut $(Y,\overline{Y})$ in $H_{N(\lambda)}(s,t)$, $\mathrm{val}(f) \leq \mathrm{capa}(Y,\overline{Y})$ holds. Furthermore, $\mathrm{val}(f) = \mathrm{capa}(Y,\overline{Y})$ holds if and only if f is a minimum s-t flow and $(Y,\overline{Y})$ is a minimum s-t cut in $H_{N(\lambda)}(s,t)$ (the well-known max-flow min-cut theorem [39]).

For an s-t flow f in $H_{N(\lambda)}(s,t)$, a residual network with respect to f, denoted by $H_{N(\lambda)}(s,t)(f)$, is defined as follows. The vertex set $V_N(s,t)(f)$ of $H_{N(\lambda)}(s,t)(f)$ is the vertex set $V_N(s,t)$ of $H_{N(\lambda)}(s,t)$. The edge set $E_N(s,t)(f)$ of $H_{N(\lambda)}(s,t)(f)$ is defined as follows. For an edge $e=(u,v)$ of $H_{N(\lambda)}(s,t)$, let $e^{\mathrm{rev}} = (v,u)$ (i.e., $e^{\mathrm{rev}} = (v,u)$ is the reverse edge of $e=(u,v) \in E_N(s,t)$). Let

$$E_N^{\mathrm{rev}}(s,t) = \{e^{\mathrm{rev}} \mid e \in E_N(s,t)\}.$$

The residual capacity $\mathrm{capa}_f(a)$ of an edge $a=(u,v) \in E_N(s,t) + E_N^{\mathrm{rev}}(s,t)$ is defined as follows:

$$\mathrm{capa}_f(a) = \begin{cases} \mathrm{capa}(a) - f(a) & (a \in E_N(s,t)) \\ f(e) & (a = e^{\mathrm{rev}} \in E_N^{\mathrm{rev}}(s,t),\ e \in E_N(s,t)). \end{cases} \tag{6}$$

Then the edge set $E_N(s,t)(f)$ of $H_{N(\lambda)}(s,t)(f)$ is defined by

$$E_N(s,t)(f) = \{a \in E_N(s,t) + E_N^{\mathrm{rev}}(s,t) \mid \mathrm{capa}_f(a) > 0\}. \tag{7}$$

Thus, the capacity $\mathrm{capa}_f(a)$ of each edge a in the residual network $H_{N(\lambda)}(s,t)(f)$ is positive. It is well known that an s-t flow f in $H_{N(\lambda)}(s,t)$ is maximum if and only if there is no s-t path in the residual network $H_{N(\lambda)}(s,t)(f)$ [39].

4.1 Finding a maximum flow f_λ in $H_{N(\lambda)}(s,t)$

A maximum s-t flow f_λ in $H_{N(\lambda)}(s,t)$ can be found by Procedure FindMaxFlow($H_{N(\lambda)}(s,t)$) below, which is almost the same as Core Mechanism $\mathcal{M}_1$ (ALGORITHM 1).

```
**Procedure** FindMaxFlow($H_{N(\lambda)}(s,t)$)
sort $\mathcal{C}_N = (C_i : i \in N)$ in a lexicographic order with respect to $(\beta_i, \alpha_i)$ 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 $A_i = [a_i, b_i) \setminus \sum_{i'=0}^{i-1} A_{i'}$ of length $\min\{\lambda, len(C_i \setminus \sum_{i'=0}^{i-1} A_{i'})\}$ such that
        $[a_i, b_i) \subseteq C_i$ and $a_i$ is the leftmost endpoint in $C_i \setminus \sum_{i'=0}^{i-1} A_{i'}$;
    $f(s, C_i) = len(A_i)$;
let $A = \sum_{i=1}^{n} A_i$;
for each $I_\ell \in \mathcal{I}_N$ do  $f(I_\ell, t) = len(A \cap I_\ell)$;
// for each edge $(C_i, I_\ell) \in E_N$ in $H_{N(\lambda)}(s,t)$ do  $f(C_i, I_\ell) = len(A_i \cap I_\ell)$ implicitly;
```

Figure 5 shows a maximum s-t flow $f = f_\lambda$ found by FindMaxFlow($H_{N(\lambda)}(s,t)$) for the example in Figure 3. Figure 6 shows the the residual network $H_{N(\lambda)}(s,t)(f_\lambda)$ with respect to f_λ in Figure 5. Figure 7 shows a minimum s-t cut $(Y_\lambda, \overline{Y_\lambda})$ in $H_{N(\lambda)}(s,t)$, where $\overline{Y_\lambda}$ is the set of vertices v such that there is a v-t path in $H_{N(\lambda)}(s,t)(f_\lambda)$ in Figure 6.

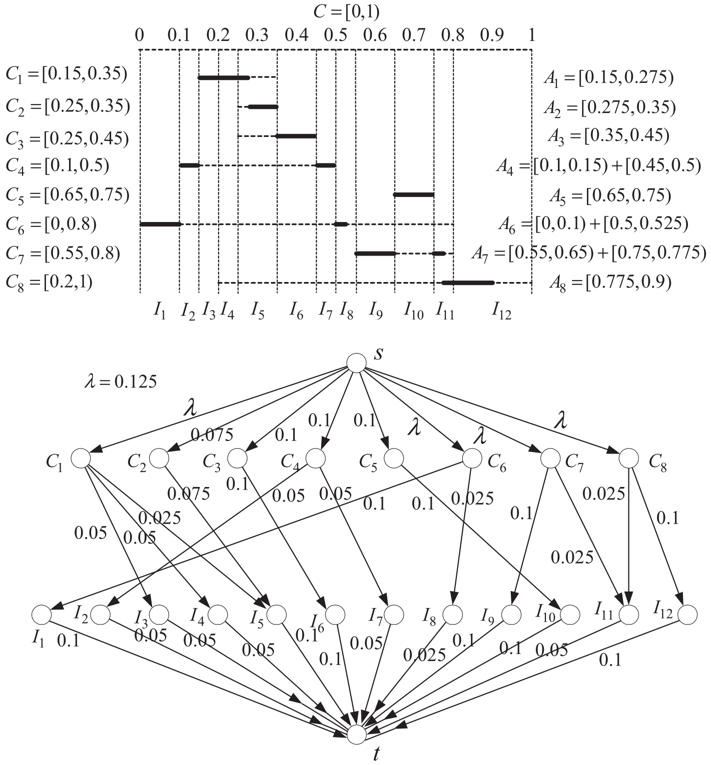


Figure 5: Maximum s-t flow $f = f_\lambda$ with $\lambda = 0.125$ found in FindMaxFlow($H_{N(\lambda)}(s,t)$) for valuation intervals $\mathcal{C}_N = (C_i : i \in N)$ (and $\mathcal{I}_N = (I_\ell : 1 \le \ell \le m)$) in Figure 3.

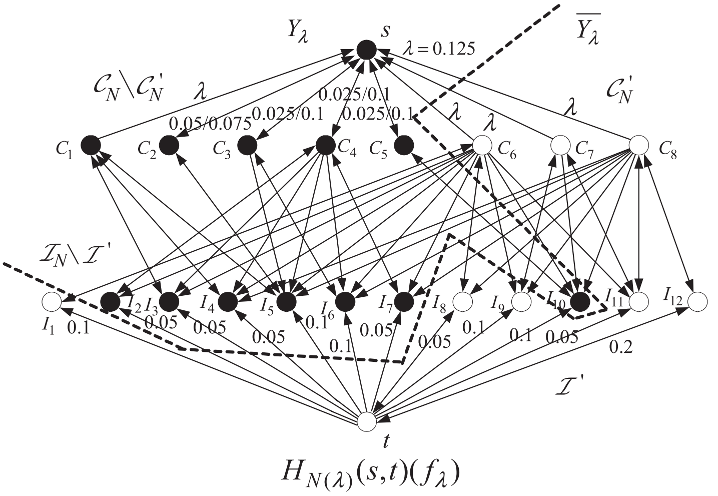


Figure 6: The residual network $H_{N(\lambda)}(s,t)(f_\lambda)$ with respect to f_λ in Figure 5 and $\overline{Y_\lambda}$ is the set of vertices v (shown by white circle) such that there is a v-t path in $H_{N(\lambda)}(s,t)(f_\lambda)$.

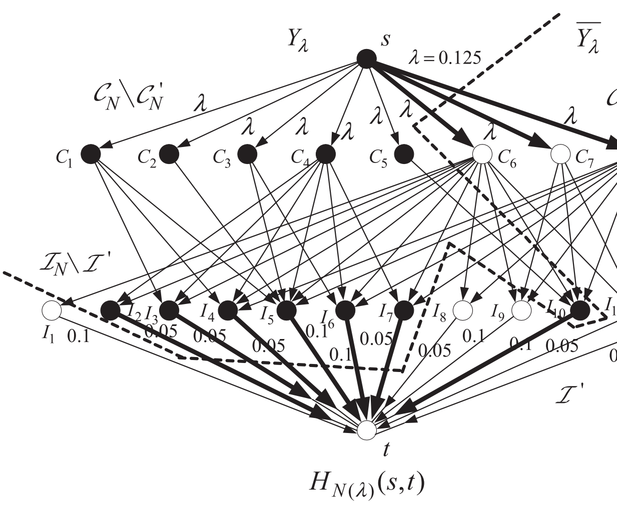


Figure 7: A minimum s-t cut $(Y_\lambda, \overline{Y_\lambda})$ in $H_{N(\lambda)}(s,t)$, where $\overline{Y_\lambda}$ is the set of vertices v (shown by white circle) such that there is a v-t path in $H_{N(\lambda)}(s,t)(f_\lambda)$ in Figure 6.

Thus, the capacity of the s-t cut $(Y_\lambda, \overline{Y_\lambda})$ in $H_{N(\lambda)}(s,t)$ in Figure 7 is

$$\text{capa}(Y_\lambda, \overline{Y_\lambda}) = \lambda|\overline{Y_\lambda} \cap \mathcal{C}_N| + \sum_{v \in Y_\lambda \cap \mathcal{I}_N} \text{capa}(v,t) = 3\lambda + 0.5 = 0.875,$$

since $\overline{Y_\lambda} = \{C_6, C_7, C_8, I_1, I_8, I_9, I_{11}, I_{12}, t\}$ and $\lambda = 0.125$.

The following lemma holds.

Lemma 4.1 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 $i \in N$ and $\bigcup_{C_i \in \mathcal{C}_N} C_i = C$, Procedure FindMaxFlow($H_{N(\lambda)}(s,t)$) correctly finds a maximum s-t flow $f = f_\lambda$ in $H_{N(\lambda)}(s,t)$ in $O(n \log n)$ time and uses at most $2n-2$ cuts, where $f_\lambda(s, C_i) = \mathit{len}(A_i) = \min\{\lambda, \mathit{len}(C_i \backslash \sum_{i'=0}^{i-1} A_{i'})\}$ with $A_i \subseteq C_i$ for each $C_i \in \mathcal{C}_N$, $f_\lambda(I_\ell, t) = \mathit{len}(A \cap I_\ell)$ for each $I_\ell \in \mathcal{I}_N$ with $A = \sum_{i=1}^{n} A_i$ and $f_\lambda(C_i, I_\ell) = \mathit{len}(A_i \cap I_\ell)$ implicitly for each edge $(C_i, I_\ell) \in E_N$ in $H_{N(\lambda)}(s,t)$.

Before giving a proof, we show one more example which will be of help to understand the proof more easily.

Figure 8 shows an example of solid valuation intervals $\mathcal{C}_N = (C_i : i \in N)$, $\mathcal{I}_N = (I_\ell : 1 \leq \ell \leq m)$ and flow network $H_{N(\lambda)}(s,t)$. Figure 9 shows an allocation $A = (A_i : i \in N)$ found by FindMaxFlow($H_{N(\lambda)}(s,t)$) with $\lambda = 0.1$ for valuation intervals $\mathcal{C}_N = (C_i : i \in N)$ (and $\mathcal{I}_N = (I_\ell : 1 \leq \ell \leq m)$) in Figure 8. Figure 10 shows the maximum s-t flow f_λ with $\lambda = 0.1$ found by FindMaxFlow($H_{N(\lambda)}(s,t)$) corresponding to allocation $A = (A_i : i \in N)$ in Figure 9. Figure 11 shows the residual network $H_{N(\lambda)}(s,t)(f_\lambda)$ with respect to f_λ in Figure 9. Figure 12 shows the minimum s-t cut $(Y_\lambda, \overline{Y_\lambda})$ in $H_{N(\lambda)}(s,t)$, where $\overline{Y_\lambda}$ is the set of vertices v (shown by white circle) such that there is a v-t path in $H_{N(\lambda)}(s,t)(f_\lambda)$ in Figure 11.

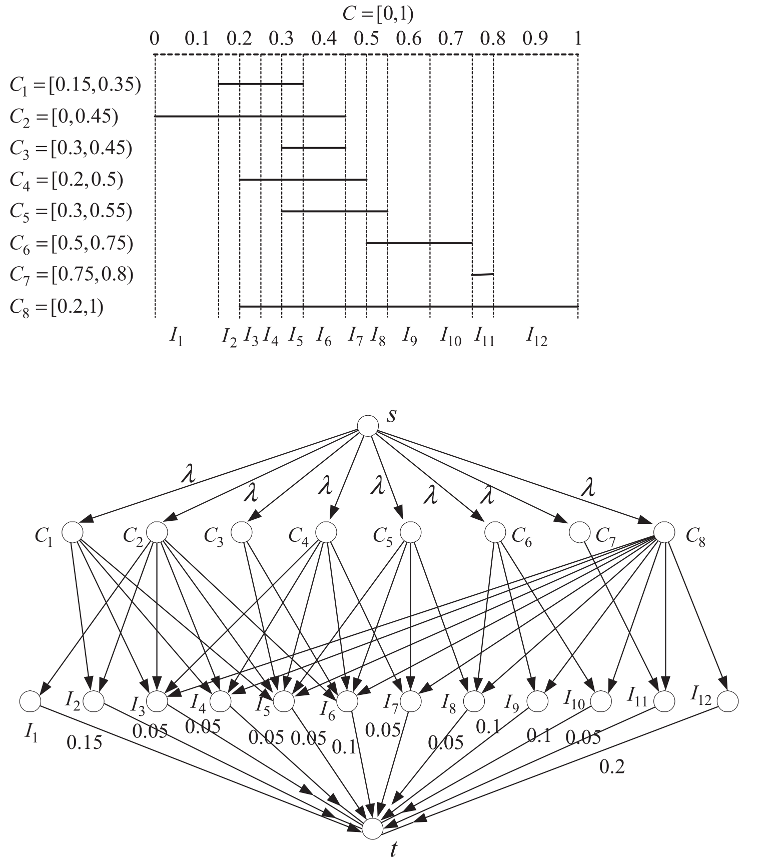


Figure 8: Example of solid valuation intervals $\mathcal{C}_N = (C_i : i \in N)$, $\mathcal{I}_N = (I_\ell : 1 \leq \ell \leq m)$ and flow network $H_{N(\lambda)}(s,t)$.

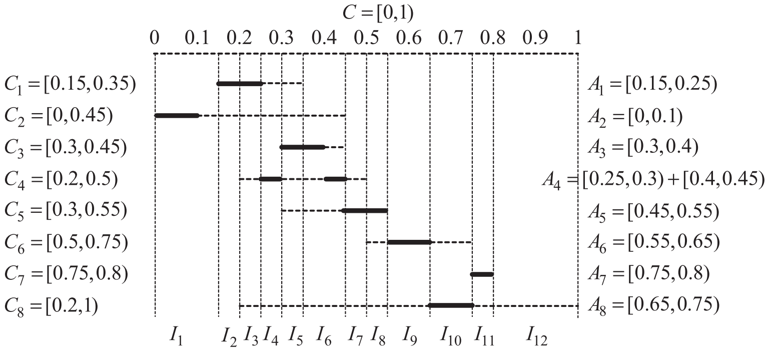


Figure 9: Allocation $A = (A_i : i \in N)$ found by FindMaxFlow$(H_{N(\lambda)}(s,t))$ with $\lambda = 0.1$ for valuation intervals $\mathcal{C}_N = (C_i : i \in N)$ (and $\mathcal{I}_N = (I_\ell : 1 \leq \ell \leq m)$) in Figure 8.

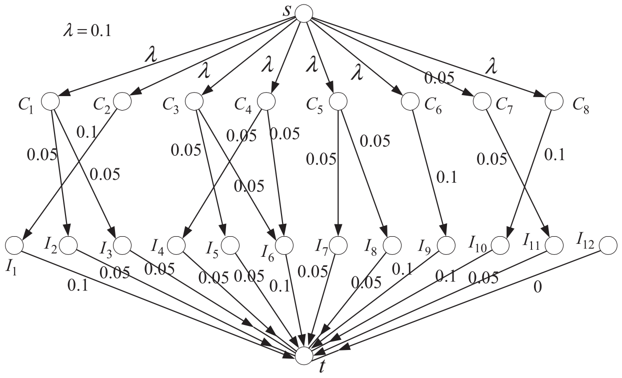


Figure 10: Maximum s-t flow $f = f_\lambda$ with $\lambda = 0.1$ found by FindMaxFlow($H_{N(\lambda)}(s,t)$) corresponding to allocation $A = (A_i : i \in N)$ in Figure 9.

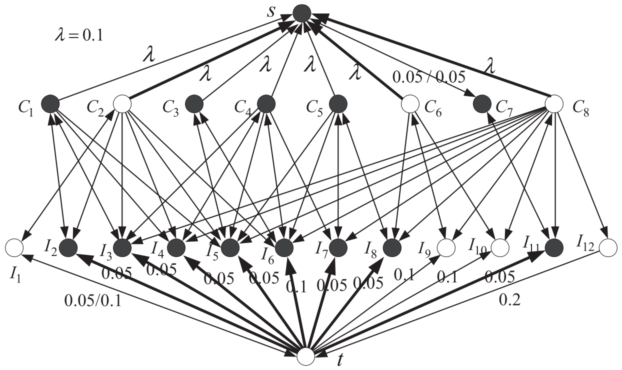


Figure 11: The residual network $H_{N(\lambda)}(s,t)(f_\lambda)$ with respect to f_λ in Figure 10.

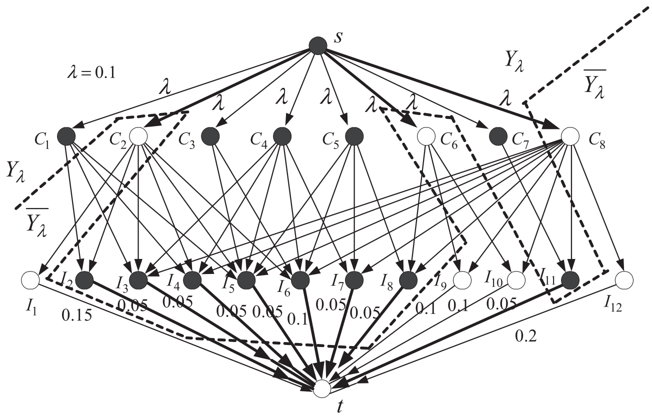


Figure 12: Minimum s-t cut $(Y_\lambda, \overline{Y_\lambda})$ in $H_{N(\lambda)}(s,t)$, where $\overline{Y_\lambda}$ is the set of vertices v (shown by white circle) such that there is a v-t path in $H_{N(\lambda)}(s,t)(f_\lambda)$ in Figure 11.

Now we are ready to give our proof.

Proof of Lemma 4.1: It is clear that $A_i \subseteq C_i$ and $f_\lambda(s, C_i) = len(A_i)$ for each $C_i \in \mathcal{C}_N$, $f_\lambda(I_\ell, t) = len(A \cap I_\ell)$ for each $I_\ell \in \mathcal{I}_N$ with $A = \sum_{i=1}^n A_i$, and $f_\lambda(C_i, I_\ell) = len(A_i \cap I_\ell)$ implicitly for each edge $(C_i, I_\ell) \in E_N$ in $H_{N(\lambda)}(s,t)$. Thus, f_λ is an s-t flow in $H_{N(\lambda)}(s,t)$.

We only give a proof that f_λ is a maximum s-t flow in $H_{N(\lambda)}(s,t)$, since the time complexity $O(n \log n)$ and the number of cuts at most $2n-2$ can be obtained by the same argument as in Proof of Theorem 3.1. Our proof here is almost the same as Proof of Theorem3.1.

If $\lambda = \min\{\lambda, len(C_i \setminus \sum_{i'=0}^{i-1} A_{i'})\} < len(C_i \setminus \sum_{i'=0}^{i-1} A_{i'})$ for each $i \in N = \{1, 2, \ldots., n\}$, then we have $f_\lambda(s, C_i) = len(A_i) = \lambda = \text{capa}(s, C_i)$ for all $i \in N = \{1, 2, \ldots., n\}$ and there is no s-t path in the residual network $H_{N(\lambda)}(s,t)(f_\lambda)$, which implies that f_λ is a maximum s-t flow in $H_{N(\lambda)}(s,t)$.

Thus, we can assume, there is $i \in N = \{1, 2, \ldots., n\}$ such that $len(C_i \setminus \sum_{i'=0}^{i-1} A_{i'}) = \min\{\lambda, len(C_i \setminus \sum_{i'=0}^{i-1} A_{i'})\}$. Let $\{j_1, j_2, \ldots, j_p\}$ be the set of $i \in N = \{1, 2, \ldots., n\}$ with $len(C_i \setminus \sum_{i'=0}^{i-1} A_{i'}) = \min\{\lambda, len(C_i \setminus \sum_{i'=0}^{i-1} A_{i'})\}$. Thus,

$$\lambda = \min\{\lambda, len(C_i \setminus \sum_{i'=0}^{i-1} A_{i'})\} < len(C_i \setminus \sum_{i'=0}^{i-1} A_{i'}) \text{ for each } i \in N \setminus \{j_1, j_2, \ldots, j_p\}. \tag{8}$$

Without loss of generality, we can assume

$$1 \leq j_1 < j_2 < \cdots < j_p \leq n. \tag{9}$$

We will show that f_λ is a maximum s-t flow in $H_{N(\lambda)}(s,t)$ by induction on p.

We first consider when $p = 1$. Let $j = j_1$ and let $J = \{1, 2, \ldots, j\}$. Thus, $A_j = C_j \setminus \sum_{i=0}^{j-1} A_i$, since $len(C_i \setminus \sum_{i=0}^{j-1} A_i) = \min\{\lambda, len(C_i \setminus \sum_{i=0}^{j-1} A_i)\}$. 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 $\mathcal{C}_N = (C_i : i \in N)$ was sorted in the lexicographic order with respect to (β_i, α_i). Then $\sum_{i \in J} A_i$ of allocation $(A_i : i \in J)$ consists of several maximal contiguous intervals. Let $I = [a, b)$ be the rightmost maximal contiguous interval among the maximal contiguous intervals in $\sum_{i \in J} A_i$. Thus, $b = \beta_j$. Define $K \subseteq J$ by

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

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, which can be obtained as follows.

Of course, $C_j = [\alpha_j, \beta_j)$ is contained in I. Actually, if $A_j = C_j \setminus \sum_{i=0}^{j-1} A_i = \emptyset$ 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 $\sum_{i=0}^{j} A_i = \sum_{i=0}^{j-1} A_i$ (i.e., $C_j \subseteq I$). Otherwise (i.e., if $A_j = C_j \setminus \sum_{i=0}^{j-1} A_i \neq \emptyset$), $C_j \subseteq A_j \cup \sum_{i=0}^{j-1} A_i = \sum_{i=0}^{j} A_i$ and a single contiguous interval C_j is contained in the rightmost maximal contiguous interval I in $\sum_{i=0}^{j} A_i$.

Now suppose that there were $i \in K \setminus \{j\}$ such that $C_i \in \mathcal{C}_K$ is not contained in I. Thus, $I = [a, b)$ is a proper subinterval of $[0, b) = [0, \beta_j)$ (i.e., $a > 0$) and $C_i = [\alpha_i, \beta_i) \in \mathcal{C}_K \setminus \{C_j\}$ contains a point x in $[0, b) \setminus I = [0, a)$. Let $k \in K \setminus \{j\}$ be the minimum among such is and let x_k be a point of $C_k = [\alpha_k, \beta_k) \in \mathcal{C}_K$ contained in $[0, a) = [0, b) \setminus I$. Note that $C_k \cap I \supseteq A_k \cap I \neq \emptyset$ since $k \in K \setminus \{j\} \subseteq J \setminus \{j\}$. Thus, $\beta_k \leq \beta_j$ and $\alpha_k \leq x_k < a \leq a'_k < \beta_k$ for some $a'_k \in A_k \cap I \neq \emptyset$. Furthermore, since we chose $I = [a, b) \neq [0, b)$ as the rightmost maximal contiguous interval among the maximal contiguous intervals in $\sum_{i \in J} A_i$, we have

$\sum_{i\in J} A_i \neq [0,b) = [0,\beta_j)$. Let $I' = [a', a)$ be the rightmost maximal contiguous interval in $[0,b) \setminus \sum_{i\in J} A_i$. 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, $x_k \notin A_k$ by $I' \cap A_k \subseteq I' \cap \sum_{i\in J} A_i = \emptyset$. Then, however, Procedure FindMaxFlow($H_{N(\lambda)}(s,t)$) would have included x_k into A_k in place of some $a''_k \in A_k \cap I \neq \emptyset$, because Procedure FindMaxFlow($H_{N(\lambda)}(s,t)$) sets $A_k = [a_k, b_k) \setminus \sum_{i=0}^{k-1} A_i \subseteq C_k$ with length $\lambda = \min\{\lambda, \mathit{len}(C_k \setminus \sum_{i=0}^{k-1} A_i)\} < \mathit{len}(C_k \setminus \sum_{i=0}^{k-1} A_i)$ such that 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

$$\bigcup_{i\in K} C_i = I = \sum_{i\in K} A_i, \tag{10}$$

since $A_h \cap I = \emptyset$ for $h \in J \setminus K$ and

$$I = \sum_{i\in J} A_i \cap I = \sum_{i\in K} A_i \cap I \subseteq \sum_{i\in K} A_i \subseteq \sum_{i\in K} C_i$$

by the definitions of I and K and $A_i \subseteq C_i$ for each $i \in K$. Thus, I can be written by

$$I = I_{\ell_1} + I_{\ell_1+1} + \cdots + I_{\ell_2} = \sum_{\ell=\ell_1}^{\ell_2} I_\ell, \tag{11}$$

where $I_{\ell_1} = [x_{\ell_1-1}, x_{\ell_1})$ with $x_{\ell_1-1} = \min\{\alpha_i \mid C_i = [\alpha_i, \beta_i) \in \mathcal{C}_K\}$ and $I_{\ell_2} = [x_{\ell_2-1}, x_{\ell_2})$ with $x_{\ell_2} = \beta_j$. Thus, for each $\ell = \ell_1, \ell_1+1, \ldots, \ell_2$ (i.e., for each $I_\ell \subseteq I$), we have $\sum_{i=1}^{n} A_i \cap I_\ell = \sum_{i\in K} A_i \cap I_\ell$, since

$$A_h \cap I = \emptyset \text{ for } h \in N \setminus K \tag{12}$$

(i.e., for $h \in J \setminus K$ by definition of J and K and for $h \in N \setminus J$ by the definition of A_h in Procedure FindMaxFlow($H_{N(\lambda)}(s,t)$)). Furthermore, there is no edge from $C_i \in \mathcal{C}_K$ to $I_\ell \in \mathcal{I}_N$ with $I_\ell \cap I = \emptyset$ in $H_{N(\lambda)}(s,t)$ by Eq.(10) (Figure 13).

Since $A_i \subseteq C_i$ and $f_\lambda(s, C_i) = \mathit{len}(A_i)$ for each $i \in N$, $f_\lambda(I_\ell, t) = \mathit{len}(A \cap I_\ell)$ for each $I_\ell \in \mathcal{I}_N$ with $A = \sum_{i=1}^{n} A_i$, and $f_\lambda(C_i, I_\ell) = \mathit{len}(A_i \cap I_\ell)$ implicitly for each edge (C_i, I_ℓ) in E_N of $H_{N(\lambda)}(s,t)$ as mentioned above, we have $f_\lambda(s, C_j) = \mathit{len}(A_j) = \mathit{len}(C_j \setminus \sum_{i=0}^{j-1} A_i) \leq \lambda = \mathrm{capa}(s, C_j)$ and $f_\lambda(s, C_i) = \mathit{len}(A_i) = \lambda = \mathrm{capa}(s, C_i) < \mathit{len}(C_i \setminus \sum_{i'=0}^{i-1} A_{i'})$ for each $i \in N \setminus \{j\}$ by Eq.(8) and $p = 1$,

$$f_\lambda(I_\ell, t) = \mathit{len}(A \cap I_\ell) = \mathit{len}(\sum_{i\in K} A_i \cap I_\ell) = \mathit{len}(I_\ell) = \mathrm{capa}(I_\ell, t), \tag{13}$$

for each $\ell = \ell_1, \ell_1+1, \ldots, \ell_2$ (i.e., for each $I_\ell \subseteq I$) by $\bigcup_{i\in K} C_i = I = \sum_{i\in K} A_i$ in Eq.(10) and $f_\lambda(C_h, I_\ell) = 0$ for each C_h with $h \in N \setminus K$ and for each $I_\ell \in \mathcal{I}_N$ with $I_\ell \subseteq I$ by $A_h \cap I = \emptyset$ for $h \in N \setminus K$ in Eq.(12).

Let Y be the set of vertices v of $H_{N(\lambda)}(s,t)$ such that there is a s-v path in the residual network $H_{N(\lambda)}(s,t)(f_\lambda)$ with respect to f_λ. If $f_\lambda(s, C_j) = \mathit{len}(A_j) = \lambda = \mathrm{capa}(s, C_j)$ then $Y = \{s\}$. Otherwise (i.e., if $f_\lambda(s, C_j) = \mathit{len}(A_j) < \lambda = \mathrm{capa}(s, C_j)$), then $s, C_j \in Y \subseteq \mathcal{C}_K + I + \{s\}$, since there is no edge from $\mathcal{C}_K$ to $\mathcal{I}_N \setminus I$ in $H_{N(\lambda)}(s,t)$, there is no edge from I to t in $H_{N(\lambda)}(s,t)(f_\lambda)$ by Eq.(13), and $f_\lambda(C_h, I_\ell) = 0$ for each C_h with $h \in N \setminus K$ and

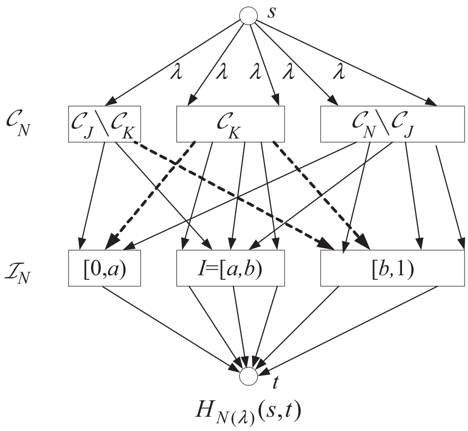


Figure 13: $\bigcup_{i\in K} C_i = I$ and there is no edge from $C_i \in \mathcal{C}_K$ to $I_\ell \in \mathcal{I}_N$ with $I_\ell \cap I = \emptyset$ in $H_{N(\lambda)}(s,t)$. There is no edge from $C_i \in \mathcal{C}_J \setminus \mathcal{C}_K$ to $I_\ell \in \mathcal{I}_N$ with $I_\ell \subseteq [b,1)$ in $H_{N(\lambda)}(s,t)$.

for each $I_\ell \in \mathcal{I}_N$ with $I_\ell \subseteq I$ as mentioned above (see Figure 13). Thus, t is not contained in Y and f_λ is a maximum s-t flow in $H_{N(\lambda)}(s,t)$.

Now we assume that f_λ is a maximum s-t flow in $H_{N(\lambda)}(s,t)$ when there are at most $p-1$ numbers $i \in N = \{1,2,\ldots,n\}$ such that $len(C_i \setminus \sum_{i'=0}^{i-1} A_{i'}) = \min\{\lambda, len(C_i \setminus \sum_{i'=0}^{i-1} A_{i'})\}$ and we consider when there are exactly p numbers $i \in N = \{1,2,\ldots,n\}$ such that $len(C_i \setminus \sum_{i'=0}^{i-1} A_{i'}) = \min\{\lambda, len(C_i \setminus \sum_{i'=0}^{i-1} A_{i'})\}$. Thus, we can assume that $\{j_1, j_2, \ldots, j_p\}$ is the set of $i \in N = \{1,2,\ldots,n\}$ with $len(C_i \setminus \sum_{i'=0}^{i-1} A_{i'}) = \min\{\lambda, len(C_i \setminus \sum_{i'=0}^{i-1} A_{i'})\}$ and that Eq.(9) holds, i.e.,

$$1 \le j_1 < j_2 < \cdots < j_p \le n.$$

Let $j = j_1$ and let $J = \{1,2,\ldots,j\}$. Then the argument above holds in this case, that is, if we use the same notation J, K, I as above, then Eq.(10) holds, i.e.,

$$\bigcup_{i\in K} C_i = I = \sum_{i\in K} A_i,$$

and there is no edge from $C_i \in \mathcal{C}_K$ to $I_\ell \in \mathcal{I}_N$ with $I_\ell \cap I = \emptyset$ in $H_{N(\lambda)}(s,t)$. Furthemore, since $\mathcal{C}_N = (C_i : i \in N)$ was sorted in the lexicographic order with respect to (β_i, α_i), it is clear that there is no edge from $C_i \in \mathcal{C}_J$ to $I_\ell = [x_{\ell-1}, x_\ell) \in \mathcal{I}_N$ in $H_{N(\lambda)}(s,t)$ such that I_ℓ lies to the right of I (i.e., $\ell \ge \ell_2 + 1$) as in Figure 13. We then consider cake $C' = C \setminus \sum_{i\in J} A_i$ (thus $C' = C \setminus \sum_{i\in J} A_i$ is obtained from C by deleting all the maximal contiguous intervals in $\sum_{i\in J} A_i$) and valuations $\mathcal{C}'_{N\setminus J} = (C'_i : i \in N \setminus J)$ with $C'_i = C_i \setminus \sum_{i\in J} A_i$. We can vitually consider that all the deleted maximal contiguous intervals in $\sum_{i\in J} A_i$) are contracted (i.e., the both endpoints of each deleted maximal contiguous interval in $\sum_{i\in J} A_i$ are considered the same) and that the cake C' is a single interval and each $C'_i \in \mathcal{C}'_{N\setminus J}$ is also a single interval.

Thus, the cake cutting problem with cake C', players $N \setminus J$ and valuation intervals $\mathcal{C}'_{N\setminus J} = (C'_i : i \in N \setminus J)$ is almost the same as the original cake-cutting problem. Only difference is that valuation intervals $\mathcal{C}'_{N\setminus J} = (C'_i : i \in N \setminus J)$ may be not solid. In this case, we have only to modify C' and set $C' = \cup_{i\in N\setminus J} C'_i$. By this modification, we have the cake cutting problem with cake C', players $N \setminus J$ and solid valuation intervals $\mathcal{C}'_{N\setminus J} = (C'_i : i \in N \setminus J)$. Note that, the lexicographic order of $\mathcal{C}_{N\setminus J} = (C_i : i \in N \setminus J)$ is preserved in $\mathcal{C}'_{N\setminus J} = (C'_i : i \in N \setminus J)$. Let $H_{N\setminus J(\lambda)}(s,t)$ be the flow network obtained from valuation intervals $\mathcal{C}'_{N\setminus J} = (C'_i : i \in N \setminus J)$. Then, FindMaxFlow($H_{N\setminus J(\lambda)}(s,t)$) runs in the same way as FindMaxFlow($H_{N(\lambda)}(s,t)$) after $i = j+1$ in the instruction "**for** $i = 1$ **to** n **do**". Thus, by induction hypothesis, f_λ restricted to $H_{N\setminus J(\lambda)}(s,t)$ is a maximum s-t flow in $H_{N\setminus J(\lambda)}(s,t)$. Since f_λ restricted to $H_{J(\lambda)}(s,t)$ is also a maximum s-t flow in $H_{J(\lambda)}(s,t)$ which can be obtained by the same argument above in the case when $p = 1$ (Figure 13), we have that there is no s-t path in the residual network $H_{N(\lambda)}(s,t)(f_\lambda)$ and that f_λ is a maximum s-t flow in $H_{N(\lambda)}(s,t)$. □

4.2 Parametric Flow in $H_{N(\lambda)}(s,t)$

Parametric flows and parametric searching have been studied by many researchers [1, 5, 21, 38]. The density $\rho(X)$ of interval $X = [x', x'')$ of cake $C = [0,1)$ is closely related to the parameter λ in $H_{N(\lambda)}(s,t)$. For a maximum s-t flow f_λ in $H_{N(\lambda)}(s,t)$, we denote by $\overline{Y_\lambda}$ throughout this paper, the set of vertices v such that there is a v-t path in the residual network $H_{N(\lambda)}(s,t)(f_\lambda)$ and let

$$Y_\lambda = V_N(s,t) \setminus \overline{Y_\lambda}. \tag{14}$$

Then $(Y_\lambda, \overline{Y_\lambda})$ is a minimum s-t cut in $H_{N(\lambda)}(s,t)$ [2] (Figure 14) and

$$Y'_\lambda \subseteq Y_\lambda \quad \text{(thus, } \overline{Y_\lambda} \subseteq \overline{Y'_\lambda}) \tag{15}$$

holds for each minimum s-t cut $(Y'_\lambda, \overline{Y'_\lambda})$ in $H_{N(\lambda)}(s,t)$. That is, Y_λ is a maximum set ($\overline{Y_\lambda}$ is a minimum set) among the minimum s-t cuts $(Y'_\lambda, \overline{Y'_\lambda})$ in $H_{N(\lambda)}(s,t)$. Furthermore, for two distinct parameters λ' and λ,

$$\text{if } \lambda' < \lambda \text{ then } Y_{\lambda'} \subseteq Y_\lambda \quad \text{(thus, } \overline{Y_\lambda} \subseteq \overline{Y_{\lambda'}}) \tag{16}$$

holds [21].

Specifically, for $\lambda = \rho_{\min}$ ($\rho_{\min}$ is the minimum density of valuation intervals $\mathcal{C}_N$) and the minimum s-t cut $(Y_\lambda, \overline{Y_\lambda})$ in $H_{N(\lambda)}(s,t)$ defined by Eq.(14) above, Y_λ is the disjoint union of all the maximal intervals of minimum density $\rho_{\min}$ and its capacity capa$(Y_\lambda, \overline{Y_\lambda})$ is

$$\mathrm{capa}(Y_\lambda, \overline{Y_\lambda}) = \lambda|\overline{Y_\lambda} \cap \mathcal{C}_N| \;\; + \sum_{I_\ell \in Y_\lambda \cap \mathcal{I}_N} \mathrm{capa}(I_\ell, t). \tag{17}$$

Of course,

$$\lambda|\overline{Y_\lambda} \cap \mathcal{C}_N| = \sum_{C_i \in \overline{Y_\lambda} \cap \mathcal{C}_N} \mathrm{capa}(s, C_i), \tag{18}$$

since capa$(s, C_i) = \lambda$ for each $C_i \in \mathcal{C}_N$. 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 $0 \le \lambda \le 1$, since $Y_{\lambda'} \subseteq Y_\lambda$ (i.e., $\overline{Y_{\lambda'}} \supseteq \overline{Y_\lambda}$) holds for two distinct parameters $\lambda' < \lambda$ as described in Eq.(16) above [3].

[2] For two maximum s-t flows f_λ and g_λ in $H_{N(\lambda)}(s,t)$, let $\overline{Y_\lambda}(f_\lambda)$ be the set of vertices v such that there is a v-t path in the residual network $H_{N(\lambda)}(s,t)(f_\lambda)$ and $\overline{Y_\lambda}(g_\lambda)$ be the set of vertices v such that there is a v-t path in the residual network $H_{N(\lambda)}(s,t)(g_\lambda)$. Then $\overline{Y_\lambda}(f_\lambda) = \overline{Y_\lambda}(g_\lambda)$ holds. Thus, $(Y_\lambda, \overline{Y_\lambda})$ is uniquely determined even if we choose an arbitrary maximum s-t flow f_λ in $H_{N(\lambda)}(s,t)$.

[3] It is easy to show that if $Y_{\lambda'} \subset Y_\lambda$ then $Y_{\lambda'} \cap \mathcal{C}_N \subset Y_\lambda \cap \mathcal{C}_N$ holds.

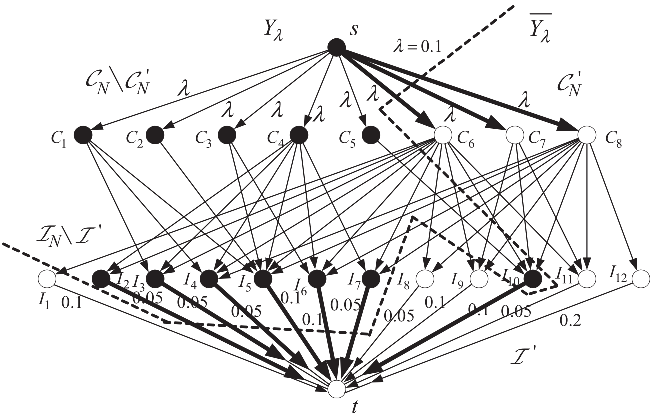


Figure 14: Minimum s-t cut $(Y_\lambda, \overline{Y_\lambda})$ in $H_{N(\lambda)}(s,t)$ in Figure 4 with $\lambda = \rho_{\min} = 0.1$, where $\mathcal{C}_N' = \mathcal{C}_N \cap \overline{Y_\lambda}$ and $\mathcal{I}' = \mathcal{I}_N \cap \overline{Y_\lambda}$.

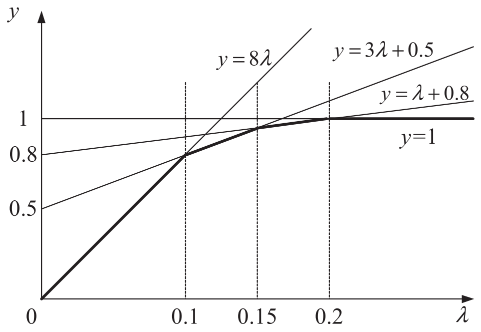


Figure 15: Minimum s-t cuts $(Y_\lambda, \overline{Y_\lambda})$ in $H_{N(\lambda)}(s,t)$ in Figure 4 for parameters λ with $0 \leq \lambda \leq 1$ form a lower envelope of the arrangement of lines generated by $y = \text{capa}(Y_\lambda, \overline{Y_\lambda}) = \lambda|\overline{Y_\lambda} \cap \mathcal{C}_N| + \sum_{I_\ell \in Y_\lambda \cap \mathcal{I}_N} \text{capa}(I_\ell, t)$ ($K = 3$ and $\lambda_1 = 0.1 < \lambda_2 = 0.15 < \lambda_3 = 0.2$).

Suppose that there are exactly $K+1$ 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 $\lambda_1, \lambda_2, \ldots, \lambda_K$ be the breakpoints of such $K+1$ distinct minimum s-t cuts $(Y_\lambda, \overline{Y_\lambda})$ in $H_{N(\lambda)}(s,t)$. We assume

$$\lambda_0 = 0 < \lambda_1 < \lambda_2 < \cdots < \lambda_K \leq 1 = \lambda_{K+1}, \tag{19}$$

where we consider $\lambda_0 = 0$ and $\lambda_{K+1} = 1$ for convenience. That is, $(Y_{\lambda_k}, \overline{Y_{\lambda_k}})$ is a minimum s-t cuts in $H_{N(\lambda)}(s,t)$ if and only if $\lambda_{k-1} \leq \lambda \leq \lambda_k$ for each $k = 1, 2, \ldots, K+1$. Thus, both $(Y_{\lambda_k}, \overline{Y_{\lambda_k}})$ and $(Y_{\lambda_{k+1}}, \overline{Y_{\lambda_{k+1}}})$ are minimum s-t cuts in $H_{N(\lambda_k)}(s,t)$ in each breakpoint λ_k for $k = 1, 2, \ldots, K$, but $(Y_{\lambda_k}, \overline{Y_{\lambda_k}})$ is not a minimum s-t cut in $H_{N(\lambda)}(s,t)$ if $\lambda < \lambda_{k-1}$ or $\lambda > \lambda_k$.

Figure 14 shows an example of network $H_{N(\lambda)}(s,t)$ corresponding to valuation intervals $\mathcal{C}_N = (C_i : i \in N)$ (and $\mathcal{I}_N = (I_\ell : 1 \leq \ell \leq m)$) in Figure 3 and the minimum s-t cut $(Y_\lambda, \overline{Y_\lambda})$ in $H_{N(\lambda)}(s,t)$ with $\lambda = \rho_{\min} = 0.1$. Figure 15 shows that the minimum s-t cuts $(Y_\lambda, \overline{Y_\lambda})$ in $H_{N(\lambda)}(s,t)$ in Figure 4 for parameters λ with $0 \leq \lambda \leq 1$ form a lower envelope

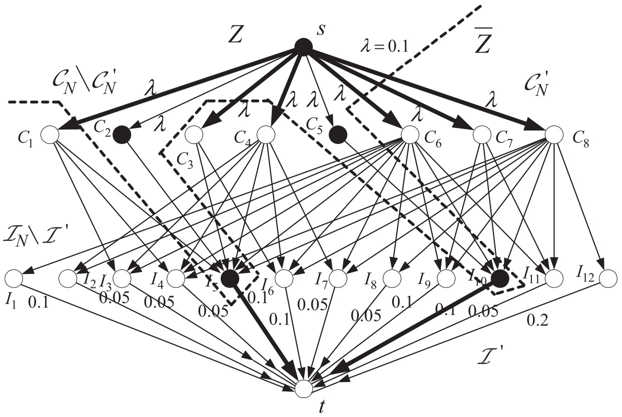


Figure 16: Another minimum s-t cut $(Z, \overline{Z})$ in $H_{N(\lambda)}(s,t)$ in Figure 4 with $\lambda = \rho_{\min} = 0.1$, where $Z = \{s, C_2, C_5, I_5, I_{10}\}$ and $\overline{Z} = \{t\} + (\mathcal{C}_N \setminus \{C_2, C_5\}) + (\mathcal{I}_N \setminus \{I_5, I_{10}\})$.

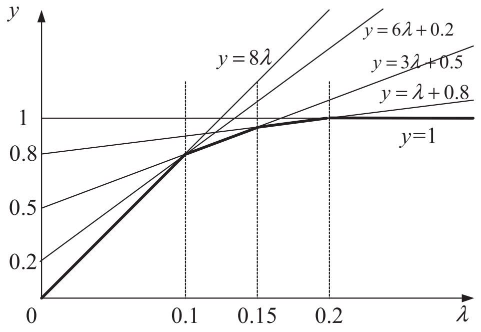


Figure 17: There are more minimum s-t cuts in $H_{N(\lambda)}(s,t)$.

of the arrangement of lines generated by

$$y = \mathrm{capa}(Y_\lambda, \overline{Y_\lambda}) = \lambda|\overline{Y_\lambda} \cap \mathcal{C}_N| \quad + \sum_{I_\ell \in Y_\lambda \cap \mathcal{I}_N} \mathrm{capa}(I_\ell, t) \tag{20}$$

$(K = 3$ 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,

$$(Z, \overline{Z}) \text{ with } Z = \{s, C_2, C_5, I_5, I_{10}\} \text{ and } \overline{Z} = \{t\} + (\mathcal{C}_N \setminus \{C_2, C_5\}) + (\mathcal{I}_N \setminus \{I_5, I_{10}\})$$

is a minimum s-t cut $H_{N(\lambda)}(s,t)$ with $\lambda = 0.1$ (Figure 16) and the corresponding line is

$$y = \mathrm{capa}(Z, \overline{Z}) = 6\lambda + \mathrm{capa}(I_5, t) + \mathrm{capa}(I_{10}, t) = 6\lambda + 0.2$$

(Figure 17).

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 $0 \le \lambda \le 1$, it is sufficient to consider only all the minimum s-t cuts $(Y_\lambda, \overline{Y_\lambda})$ in $H_{N(\lambda)}(s,t)$ defined by Eq.(14) above.

4.3 Finding Breakpoints $\lambda_1, \lambda_2, \ldots, \lambda_K$ in Parametric Flow

In order to find all breakpoints $\lambda_1, \lambda_2, \ldots, \lambda_K$, we use a binary search on interval (λ^-, λ^+) to find λ_k with $\lambda^- < \lambda_k < \lambda^+$ based on the method in [21].

We initially set $\lambda^- = 0$, $\lambda^+ = 1$ and $H_{N(\lambda)(\lambda^-,\,\lambda^+)}(s,t) = H_{N(\lambda)(0,1)}(s,t) = H_{N(\lambda)}(s,t)$. Then we find the minimum s-t cut $(Y_{\lambda^-}, \overline{Y_{\lambda^-}})$ in $H_{N(\lambda)(\lambda^-,\,\lambda^+)}(s,t)$ with $\lambda = \lambda^-$ (which is denoted by $H_{N(\lambda^-)(\lambda^-,\,\lambda^+)}(s,t)$) and the minimum s-t cut $(Y_{\lambda^+}, \overline{Y_{\lambda^+}})$ in $H_{N(\lambda)(\lambda^-,\,\lambda^+)}(s,t)$ with $\lambda = \lambda^+$ (which is also denoted by $H_{N(\lambda^+)(\lambda^-,\,\lambda^+)}(s,t)$). Thus, intially, we find the two minimum s-t cuts $(Y_\lambda, \overline{Y_\lambda})$ in $H_{N(\lambda)(0,1)}(s,t) = H_{N(\lambda)}(s,t)$ with $\lambda = \lambda^- = 0$ and $\lambda = \lambda^+ = 1$.

Let $y_{\lambda^-}(\lambda)$ be the capacity of the s-t cut $(Y_{\lambda^-}, \overline{Y_{\lambda^-}})$ in $H_{N(\lambda)(\lambda^-,\,\lambda^+)}(s,t)$. Similarly, let $y_{\lambda^+}(\lambda)$ be the capacity of the s-t cut $(Y_{\lambda^+}, \overline{Y_{\lambda^+}})$ in $H_{N(\lambda)(\lambda^-,\,\lambda^+)}(s,t)$. Thus,

$$\begin{aligned} y_{\lambda^-}(\lambda) &= \mathrm{capa}(Y_{\lambda^-}, \overline{Y_{\lambda^-}}) = \lambda|\overline{Y_{\lambda^-}} \cap \mathcal{C}_N| + \sum_{I_\ell \in Y_{\lambda^-} \cap \mathcal{I}_N} \mathrm{capa}(I_\ell, t), \qquad (21) \\ y_{\lambda^+}(\lambda) &= \mathrm{capa}(Y_{\lambda^+}, \overline{Y_{\lambda^+}}) = \lambda|\overline{Y_{\lambda^+}} \cap \mathcal{C}_N| + \sum_{I_\ell \in Y_{\lambda^+} \cap \mathcal{I}_N} \mathrm{capa}(I_\ell, t). \qquad (22) \end{aligned}$$

Initially, since $\lambda^- = 0$, $\lambda^+ = 1$ and the solidness of valuation intervals $\mathcal{C}_N = (C_i : i \in N)$, we have $Y_{\lambda^-} = \{s\}$, $\overline{Y_{\lambda^-}} = V_N(s,t) \setminus \{s\}$ and $Y_{\lambda^+} = V_N(s,t) \setminus \{t\}$, $\overline{Y_{\lambda^+}} = \{t\}$. Thus, initially, $y_{\lambda^-}(\lambda) = y_0(\lambda) = n\lambda$ and $y_{\lambda^+}(\lambda) = y_1(\lambda) = 1$. Note that,

$$\begin{aligned} y_{\lambda^-}(\lambda^-) &= \mathrm{capa}(Y_{\lambda^-}, \overline{Y_{\lambda^-}}) = \lambda^-|\overline{Y_{\lambda^-}} \cap \mathcal{C}_N| + \sum_{I_\ell \in Y_{\lambda^-} \cap \mathcal{I}_N} \mathrm{capa}(I_\ell, t), \\ y_{\lambda^+}(\lambda^+) &= \mathrm{capa}(Y_{\lambda^+}, \overline{Y_{\lambda^+}}) = \lambda^+|\overline{Y_{\lambda^+}} \cap \mathcal{C}_N| + \sum_{I_\ell \in Y_{\lambda^+} \cap \mathcal{I}_N} \mathrm{capa}(I_\ell, t). \end{aligned}$$

In each iteration, we first find λ^* such that $y_{\lambda^-}(\lambda^*) = y_{\lambda^+}(\lambda^*)$ by Eq.(21) and Eq.(22). Then we find the minimum s-t cut $(Y_{\lambda^*}, \overline{Y_{\lambda^*}})$ in $H_{N(\lambda^*)(\lambda^-,\,\lambda^+)}(s,t)$ (i.e., in $H_{N(\lambda)(\lambda^-,\,\lambda^+)}(s,t)$ with $\lambda = \lambda^*$) and let $y_{\lambda^*}(\lambda)$ be the capacity of the s-t cut $(Y_{\lambda^*}, \overline{Y_{\lambda^*}})$ in $H_{N(\lambda)(\lambda^-,\,\lambda^+)}(s,t)$. Thus,

$$y_{\lambda^*}(\lambda) = \lambda|\overline{Y_{\lambda^*}} \cap \mathcal{C}_N| + \sum_{I_\ell \in Y_{\lambda^*} \cap \mathcal{I}_N} \mathrm{capa}(I_\ell, t). \qquad (23)$$

If $y_{\lambda^*}(\lambda^*) = y_{\lambda^-}(\lambda^*) = y_{\lambda^+}(\lambda^*)$ then we set $\lambda_k = \lambda^*$ and 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 $Y_{\lambda^*} \setminus \{s\}$ (Figure 18).

Let $\mathcal{C}_{N(\lambda^-,\,\lambda^+)}$ be the set of vertices $C_i \in \mathcal{C}_N$ which are contained in $H_{N(\lambda)(\lambda^-,\,\lambda^+)}(s,t)$. Similarly, let $N(\lambda^-,\,\lambda^+)$ be the set of players $i \in N$ with $C_i \in \mathcal{C}_{N(\lambda^-,\,\lambda^+)}$ and let $\mathcal{I}_{N(\lambda^-,\,\lambda^+)}$ be the set of vertices $I_\ell \in \mathcal{I}_N$ which are contained in $H_{N(\lambda)(\lambda^-,\,\lambda^+)}(s,t)$. Thus, initially, $\mathcal{C}_{N(0,1)} = \mathcal{C}_N$ and $\mathcal{I}_{N(0,1)} = \mathcal{I}_N$, since $H_{N(\lambda)(0,1)}(s,t) = H_{N(\lambda)}(s,t)$.

Thus, all $\lambda_1, \lambda_2, \ldots, \lambda_K$ can be found by Parametric Flow Algorithm (ALGORITHM 2) below, where Procedure FindBreakpoints($H_{N(\lambda)(\lambda^-,\lambda^+)}(s,t)$) below is used in order to find the minimum s-t cut $(Y_{\lambda^*}, \overline{Y_{\lambda^*}})$ in $H_{N(\lambda^*)(\lambda^-,\,\lambda^+)}(s,t)$.

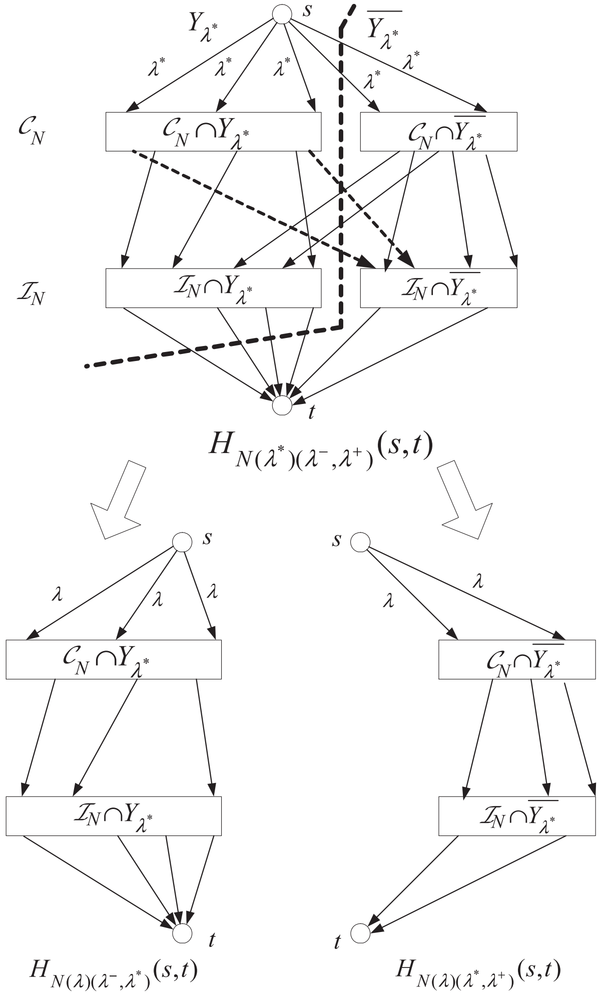

s
Y_{λ^*}
$\overline{Y_{\lambda^*}}$
λ^*
$\mathcal{C}_N$
$\mathcal{C}_N \cap Y_{\lambda^*}$
$\mathcal{C}_N \cap \overline{Y_{\lambda^*}}$
$\mathcal{I}_N$
$\mathcal{I}_N \cap Y_{\lambda^*}$
$\mathcal{I}_N \cap \overline{Y_{\lambda^*}}$
t
$H_{N(\lambda^*)(\lambda^-,\lambda^+)}(s,t)$
λ
$H_{N(\lambda)(\lambda^-,\lambda^*)}(s,t)$
$H_{N(\lambda)(\lambda^*,\lambda^+)}(s,t)$

Figure 18: There is no edge from a vertex in $Y_{\lambda^*} \cap \mathcal{C}_N$ to a vertex in $\overline{Y_{\lambda^*}} \cap \mathcal{I}_\mathcal{N}$ in $H_{N(\lambda^*)(\lambda^-,\lambda^+)}(s,t)$. $H_{N(\lambda)(\lambda^-,\lambda^*)}(s,t)$ is obtained from $H_{N(\lambda)(\lambda^-,\lambda^+)}(s,t)$ by deleting $\overline{Y_{\lambda^*}} \setminus \{t\}$ and $H_{N(\lambda)(\lambda^*,\lambda^+)}(s,t)$ is obtained from $H_{N(\lambda)(\lambda^-,\lambda^+)}(s,t)$ by deleting $Y_{\lambda^*} \setminus \{s\}$.

ALGORITHM 2: Parametric Flow Algorithm

```
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 $\sum_{i \in N} A'_i = C$.
Let $X_N = \{x_0, x_1, \ldots, x_m\}$ be the set of all $\alpha_i, \beta_i$ of $C_i = [\alpha_i, \beta_i)$ of $\mathcal{C}_N = (C_i : i \in N)$;
sort $X_N$ and assume $x_0 < x_1 < \cdots < x_m$, where $x_0 = 0$ and $x_m = 1$;
let $I_\ell = [x_{\ell-1}, x_\ell)$ for each $\ell$ with $1 \leq \ell \leq m$;
let $\mathcal{I}_N = (I_\ell : 1 \leq \ell \leq m)$;
sort $\mathcal{C}_N = (C_i : i \in N)$ in a lexicographic order with respect to $(\beta_i, \alpha_i)$ and assume
    $C_1 \leq C_2 \leq \cdots \leq C_n$ in this order;
set $\lambda^- = 0$ and $\lambda^+ = 1$;
Consider $H_{N(\lambda)(\lambda^-, \lambda^+)}(s, t)$ with $\lambda^- < \lambda < \lambda^+$ implicitly;
$K = 0$;
FindBreakpoints($H_{N(\lambda)(\lambda^-, \lambda^+)}(s, t)$);
```

Procedure FindBreakpoints($H_{N(\lambda)(\lambda^-,\lambda^+)}(s,t)$)

```
find $\lambda^*$ 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 $i_1 < i_2 < \ldots < i_p$;
// find a maximum $s$-$t$ flow $f$ in $H_{N(\lambda^*)(\lambda^-, \lambda^+)}(s, t)$ as follows:
$A_{i_0} = \emptyset$;
for $j = 1$ to $p$ do
    let $Z = \sum_{j'=0}^{j-1} A_{i_{j'}} + \sum_{I_\ell \in \mathcal{I}_N \setminus \mathcal{I}_{N(\lambda^-, \lambda^+)}} I_\ell$;
    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
        leftmost 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 $I_\ell \in \mathcal{I}_{N(\lambda^-, \lambda^+)}$ do  $f(I_\ell, t) = len(A \cap I_\ell)$;
for each $(C_{i_j}, I_\ell) \in E_N$ in $H_{N(\lambda)(\lambda^-, \lambda^+)}(s, t)$ do  $f(C_{i_j}, I_\ell) = len(A_{i_j} \cap I_\ell)$ implicitly;
// $f$ is a maximum $s$-$t$ flow in $H_{N(\lambda^*)(\lambda^-, \lambda^+)}(s, t)$ as shown in FindMaxFlow($\cdot$)
let $H_{N(\lambda^*)(\lambda^-, \lambda^+)}(s, t)(f)$ be the 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 $Y_{\lambda^*}$ be the set of vertices $v$ of $H_{N(\lambda^*)(\lambda^-, \lambda^+)}(s, t)(f)$ not contained in $\overline{Y_{\lambda^*}}$;
let $y_{\lambda^*}(\lambda) = \lambda|\overline{Y_{\lambda^*}} \cap \mathcal{C}_N| + \sum_{I_\ell \in Y_{\lambda^*} \cap \mathcal{I}_N} \mathrm{capa}(I_\ell, t)$;
if $y_{\lambda^*}(\lambda^*) = y_{\lambda^-}(\lambda^*) = y_{\lambda^+}(\lambda^*)$ then
    $K = K + 1$; $\lambda_K = \lambda^*$; for $j = 1$ to $p$ do $s_{i_j} = \lambda_K$;
else
    FindBreakpoints($H_{N(\lambda)(\lambda^-, \lambda^*)}(s, t)$) where $H_{N(\lambda)(\lambda^-, \lambda^*)}(s, t)$ is obtained from
        $H_{N(\lambda)(\lambda^-, \lambda^+)}(s, t)$ by deleting $\overline{Y_{\lambda^*}} \setminus \{t\}$;
    FindBreakpoints($H_{N(\lambda)(\lambda^*, \lambda^+)}(s, t)$) where $H_{N(\lambda)(\lambda^*, \lambda^+)}(s, t)$ is obtained from
        $H_{N(\lambda)(\lambda^-, \lambda^+)}(s, t)$ by deleting $Y_{\lambda^*} \setminus \{s\}$.
```

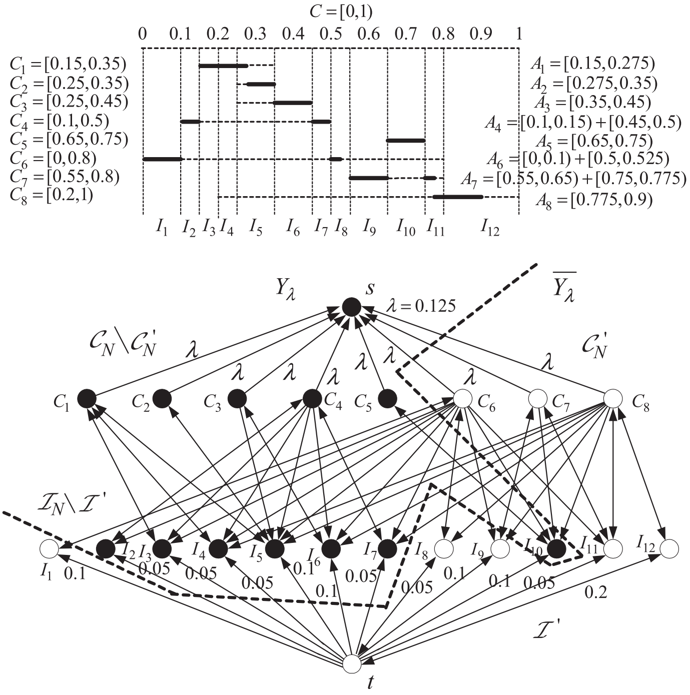


Figure 19: Maximum s-t flow $f = f_{\lambda^*}$ with $\lambda = \lambda^* = 0.125$ found in the first call of FindBreakpoints($H_{N(\lambda)(\lambda^-,\lambda^+)}(s,t)$) in Parametric Flow Algorithm (ALGORITHM 2) for valuation intervals $\mathcal{C}_N = (C_i : i \in N)$ (and $\mathcal{I}_N = (I_j : 1 \leq j \leq m)$) in Figure 3 and the residual network $H_{N(\lambda^*)(\lambda^-,\lambda^+)}(s,t)(f_{\lambda^*})$.

Figure 19 shows how Parametric Flow Algorithm works for valuation intervals $\mathcal{C}_N = (C_i : i \in N)$ (and $\mathcal{I}_N = (I_j : 1 \leq j \leq m)$) in Figure 3. Initially, since $\lambda^- = 0$, $\lambda^+ = 1$, $y_{\lambda^-}(\lambda) = y_0(\lambda) = n\lambda$, and $y_{\lambda^+}(\lambda) = y_1(\lambda) = 1$, we have $n\lambda = 8\lambda = 1$ and $\lambda^* = \frac{1}{8} = 0.125$. By FindBreakpoints($H_{N(\lambda)(\lambda^-,\lambda^+)}(s,t)$), we have

$$\overline{Y_{\lambda^*}} = \overline{Y_{0.125}} = \{C_6, C_7, C_8, I_1, I_8, I_9, I_{11}, I_{12}, t\}, \quad Y_{\lambda^*} = Y_{0.125} = V(s,t) \setminus \overline{Y_{\lambda^*}}$$

and

$$y_{\lambda^*}(\lambda^*) = 3\lambda^* + 0.5 = 0.875 \neq y_{\lambda^-}(\lambda^*) = 8\lambda^* = 1 = y_{\lambda^+}(\lambda^*).$$

Thus, FindBreakpoints($H_{N(\lambda)(\lambda^-,\lambda^*)}(s,t)$) is recursively called, where

$$H_{N(\lambda)(\lambda^-,\lambda^*)}(s,t) = H_{N(\lambda)(0,0.125)}(s,t)$$

is obtained from $H_{N(\lambda)(0,1)}(s,t)$ by deleting $\overline{Y_{\lambda^*}} \setminus \{t\}$.
Then FindBreakpoints($H_{N(\lambda)(\lambda^*,\lambda^+)}(s,t)$) is recursively called, where

$$H_{N(\lambda)(\lambda^*,\lambda^+)}(s,t) = H_{N(\lambda)(0.125,1)}(s,t)$$

is obtained from $H_{N(\lambda)(\lambda^-,\lambda^+)}(s,t)$ by deleting $Y_{\lambda^*} \setminus \{s\}$.

Figure 20 shows FindBreakpoints($H_{N(\lambda)(0,0.125)}(s,t)$). Since $y_{\lambda^-}(\lambda) = y_0(\lambda) = 5\lambda$, and $y_{\lambda^+}(\lambda) = y_{0.125}(\lambda) = 0.5$, we have $5\lambda = 0.5$ and $\lambda^* = \frac{0.5}{5} = 0.1$.

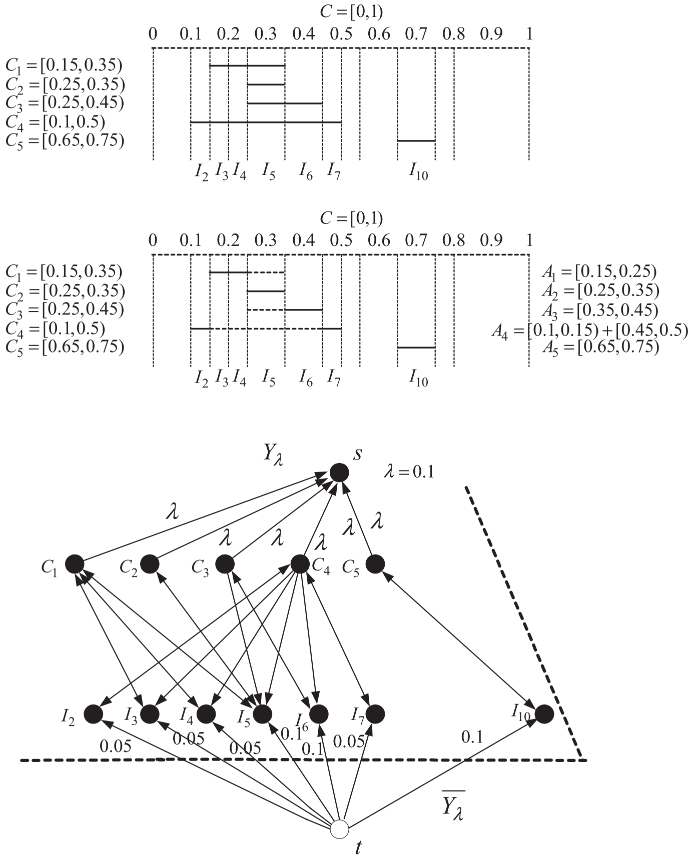


Figure 20: In the second call of FindBreakpoints($H_{N(\lambda)(\lambda^-,\lambda^+)}(s,t)$).

By FindBreakpoints($H_{N(\lambda)(0,0.125)}(s,t)$), we have $\overline{Y_{\lambda^*}} = \overline{Y_{0.1}} = \{t\}$, $Y_{\lambda^*} = Y_{0.1} = (Y_{0.125} \cup \{t\}) \setminus \overline{Y_{\lambda^*}} = Y_{0.125}$ and $y_{\lambda^*}(\lambda^*) = 0.5 = y_{\lambda^-}(\lambda^*) = 5\lambda^* = y_{\lambda^+}(\lambda^*)$. Thus, FindBreakpoints($H_{N(\lambda)(0,0.125)}(s,t)$) sets $K = K + 1 = 1$ and $\lambda_1 = \lambda^* = 0.1$.

Figure 21 shows FindBreakpoints($H_{N(\lambda)(0.125,1)}(s,t)$). Since $y_{\lambda^-}(\lambda) = y_{0.125}(\lambda) = 3\lambda$, and $y_{\lambda^+}(\lambda) = y_1(\lambda) = 0.5$, we have $3\lambda = 0.5$ and $\lambda^* = \frac{0.5}{3} = \frac{1}{6} = 0.166....$ By FindBreakpoints($H_{N(\lambda)(0.125,1)}(s,t)$), we have $\overline{Y_{\lambda^*}} = \overline{Y_{0.166...}} = \{C_8, I_{12}, t\}$, $Y_{\lambda^*} = Y_{0.166...} = (\overline{Y_{0.125}} \cup \{s\}) \setminus \overline{Y_{\lambda^*}} = \{C_6, C_7, I_1, I_8, I_9, I_{11}, s\}$ and $y_{\lambda^*}(\lambda^*) = \lambda^* + 0.3 \neq 0.5 = y_{\lambda^-}(\lambda^*) = 3\lambda^* = y_{\lambda^+}(\lambda^*)$.

Thus, FindBreakpoints($H_{N(\lambda)(0.125,0.166...)}(s,t)$) is recursively called, where

$$H_{N(\lambda)(\lambda^-,\lambda^*)}(s,t) = H_{N(\lambda)(0.125,0.166...)}(s,t)$$

is obtained from $H_{N(\lambda)(0.125,1)}(s,t)$ by deleting $\overline{Y_{\lambda^*}} \setminus \{t\} = \{C_8, I_{12}\}$. Since $y_{\lambda^-}(\lambda) = y_{0.125}(\lambda) = 2\lambda$, and $y_{\lambda^+}(\lambda) = y_{0.166...}(\lambda) = 0.3$, we have $2\lambda = 0.3$ and $\lambda^* = \frac{0.3}{2} = 0.15$.

By FindBreakpoints($H_{N(\lambda)(0.125,0.166...)}(s,t)$), we have $\overline{Y_{\lambda^*}} = \overline{Y_{0.15}} = \{t\}$, $Y_{\lambda^*} = Y_{0.15} = (Y_{0.166...} \cup \{t\}) \setminus \overline{Y_{\lambda^*}} = \{C_6, C_7, I_1, I_8, I_9, I_{11}, s\}$ and $y_{\lambda^*}(\lambda^*) = 0.3 = y_{\lambda^-}(\lambda^*) = 2\lambda^* = y_{\lambda^+}(\lambda^*)$. Thus, FindBreakpoints($H_{N(\lambda)(0.125,0.166...)}(s,t)$) sets $K = K + 1 = 2$ and $\lambda_2 = \lambda^* = 0.15$.

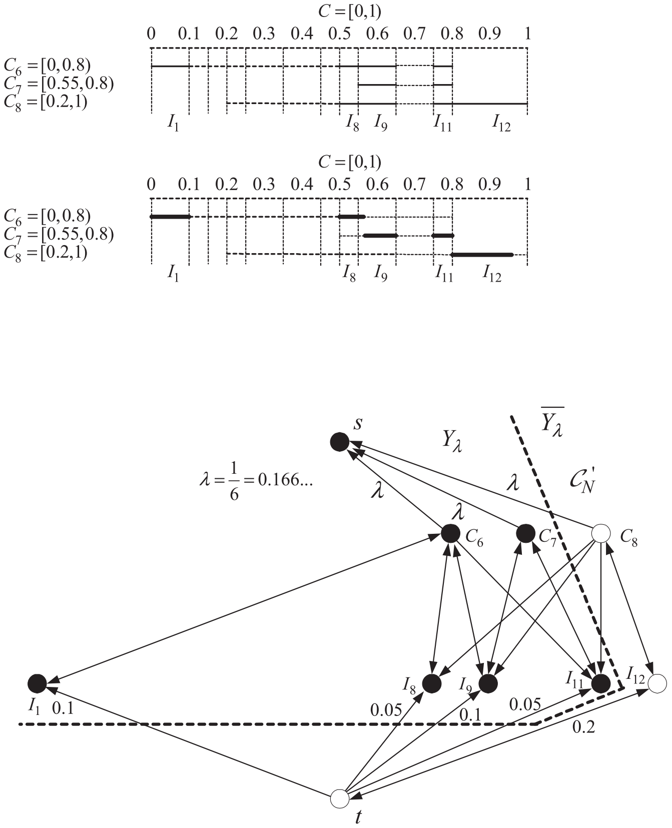


Figure 21: In the third call of FindBreakpoints($H_{N(\lambda)(\lambda^-,\lambda^+)}(s,t)$).

Then FindBreakpoints($H_{N(\lambda)(0.166...,1)}(s,t)$) is recursively called, where $H_{N(\lambda)(0.166...,1)}(s,t)$ is obtained from $H_{N(\lambda)(0.125,1)}(s,t)$ by deleting $Y_{\lambda^*} \setminus \{s\}$. Since $y_{\lambda^-}(\lambda) = y_{0.166...}(\lambda) = \lambda$, and $y_{\lambda^+}(\lambda) = y_1(\lambda) = 0.2$, we have $\lambda = 0.2$ and $\lambda^* = \frac{0.2}{1} = 0.2$.

By FindBreakpoints($H_{N(\lambda)(0.166...,1)}(s,t)$), we have $\overline{Y_{\lambda^*}} = \overline{Y_{0.2}} = \{t\}$, $Y_{\lambda^*} = Y_{0.2} = (\overline{Y_{0.166...}} \cup \{s\}) \setminus \overline{Y_{\lambda^*}} = \{C_8, I_{12}, s\}$ and $y_{\lambda^*}(\lambda^*) = 0.2 = y_{\lambda^-}(\lambda^*) = \lambda^* = y_{\lambda^+}(\lambda^*)$. Thus, FindBreakpoints($H_{N(\lambda)(0.166...,1)}(s,t)$) sets $K = K + 1 = 3$ and $\lambda_3 = \lambda^* = 0.2$.

Lemma 4.2 Procedure FindBreakpoints($H_{N(\lambda)(\lambda^-,\lambda^+)}(s,t)$) correctly finds all the breakpoints of $H_{N(\lambda)}(s,t)$ which are contained in interval (λ^-, λ^+).

Proof: We will show that the lemma holds by induction on the number of recursive calls FindBreakpoints($\cdot$) in FindBreakpoints($H_{N(\lambda)(\lambda^-,\lambda^+)}(s,t)$). If there is no recursive call FindBreakpoints($\cdot$), then $y_{\lambda^*}(\lambda^*) = y_{\lambda^-}(\lambda^*) = y_{\lambda^+}(\lambda^*)$ and λ^* is the unique breakpoint in interval (λ^-, λ^+), and thus, the lemma holds.

Now we assume that the lemma holds when the number of recursive calls FindBreakpoints($\cdot$) in FindBreakpoints($H_{N(\lambda)(\lambda^-,\lambda^+)}(s,t)$) is less than p and consider the case when exactly p recursive calls FindBreakpoints($\cdot$) are contained in FindBreakpoints($H_{N(\lambda)(\lambda^-,\lambda^+)}(s,t)$). Without loss of generality, we can assume $\lambda^- = 0$ and $\lambda^+ = 1$. Thus, $H_{N(\lambda)(\lambda^-,\,\lambda^+)}(s,t) = H_{N(\lambda)}(s,t)$, $\mathcal{C}_{N(\lambda^-,\,\lambda^+)} = \mathcal{C}_N$, $N(\lambda^-,\,\lambda^+) = N$ and $\mathcal{I}_{N(\lambda^-,\,\lambda^+)} = \mathcal{I}_N$.

Now FindBreakpoints($H_{N(\lambda)(\lambda^-,\lambda^+)}(s,t)$) first calls FindBreakpoints($H_{N(\lambda)(\lambda^-,\lambda^*)}(s,t)$) and then calls FindBreakpoints($H_{N(\lambda)(\lambda^*,\lambda^+)}(s,t)$). Both FindBreakpoints($H_{N(\lambda)(\lambda^-,\lambda^*)}(s,t)$) and FindBreakpoints($H_{N(\lambda)(\lambda^*,\lambda^+)}(s,t)$) contain less than p recursive calls FindBreakpoints($\cdot$) and, by induction hypothesis, FindBreakpoints($H_{N(\lambda)(\lambda^-,\lambda^*)}(s,t)$) finds K_1 breakpoints $\lambda'_1, \lambda'_2, \ldots, \lambda'_{K_1}$ and FindBreakpoints($H_{N(\lambda)(\lambda^*,\lambda^+)}(s,t)$) finds K_2 breakpoints $\lambda''_1, \lambda''_2, \ldots, \lambda''_{K_2}$. Thus, FindBreakpoints($H_{N(\lambda)(\lambda^-,\lambda^+)}(s,t)$) finds $K' = K_1 + K_2$ breakpoints and we can assume

$$\lambda'_1 < \lambda'_2 < \cdots < \lambda'_{K_1} < \lambda'_{K_1+1} < \cdots < \lambda_{K'}, \tag{24}$$

where $\lambda'_{K_1+k} = \lambda''_k$ for each $k = 1, 2, \ldots, K_2$. Then we have the following proposition that $K' = K$ and $\lambda'_k = \lambda_k$ for each $k = 1, 2, \ldots, K$ in Eq.(19). The proposition can be proved as follows.

By the same argument as in Proof of Lemma 4.1 for Procedure FindMaxFlow($H_{N(\lambda)}(s,t)$), we can show that FindBreakpoints($H_{N(\lambda)(\lambda^-,\lambda^+)}(s,t)$) finds a maximum s-t flow $f = f_{\lambda^*}$ in $H_{N(\lambda^*)(\lambda^-,\lambda^+)}(s,t)$, and that $\overline{Y_{\lambda^*}}$ is the set of vertices v of the residual network $H_{N(\lambda^*)(\lambda^-,\,\lambda^+)}(s,t)(f_{\lambda^*})$ such that there is a path from v to t in $H_{N(\lambda^*)(\lambda^-,\,\lambda^+)}(s,t)(f_{\lambda^*})$ and Y_{λ^*} is the set of vertices v of $H_{N(\lambda^*)(\lambda^-,\,\lambda^+)}(s,t)(f_{\lambda^*})$ not contained in $\overline{Y_{\lambda^*}}$. Thus, $(Y_{\lambda^*}, \overline{Y_{\lambda^*}})$ is a minimum s-t cut in $H_{N(\lambda^*)(\lambda^-,\,\lambda^+)}(s,t)$ and there is no edge from a vertex in $Y_{\lambda^*} \cap \mathcal{C}_N$ to a vertex in $\overline{Y_{\lambda^*}} \cap \mathcal{I}_N$ in $H_{N(\lambda^*)(\lambda^-,\,\lambda^+)}(s,t)$ (Figure 18). As mentioned before, $H_{N(\lambda)(\lambda^-,\lambda^*)}(s,t)$ is obtained from $H_{N(\lambda)(\lambda^-,\,\lambda^+)}(s,t)$ by deleting $\overline{Y_{\lambda^*}} \setminus \{t\}$ and $H_{N(\lambda)(\lambda^*,\lambda^+)}(s,t)$ is obtained from $H_{N(\lambda)(\lambda^-,\,\lambda^+)}(s,t)$ by deleting $Y_{\lambda^*} \setminus \{s\}$. The capacity of $(Y_{\lambda^*}, \overline{Y_{\lambda^*}})$ in $H_{N(\lambda^*)(\lambda^-,\,\lambda^+)}(s,t)$ is $y_{\lambda^*}(\lambda^*)$, where

$$y_{\lambda^*}(\lambda) = \mathrm{capa}(Y_{\lambda^*}, \overline{Y_{\lambda^*}}) = \lambda|\overline{Y_{\lambda^*}} \cap \mathcal{C}_N| \; + \sum_{I_\ell \in Y_{\lambda^*} \cap \mathcal{I}_N} \mathrm{capa}(I_\ell, t) \tag{25}$$

is the capacity of $(Y_{\lambda^*}, \overline{Y_{\lambda^*}})$ in $H_{N(\lambda)(\lambda^-,\,\lambda^+)}(s,t)$ as mentioned in Eq.(23).

Since f_{λ^*} is a maximum s-t flow in $H_{N(\lambda^*)(\lambda^-,\,\lambda^+)}(s,t)$ and the restriction of f_{λ^*} to $H_{N(\lambda^*)(\lambda^*,\,\lambda^+)}(s,t)$ is also a maximum s-t flow in $H_{N(\lambda^*)(\lambda^*,\,\lambda^+)}(s,t)$ by the structure of $H_{N(\lambda^*)(\lambda^-,\,\lambda^+)}(s,t)$ as shown in Figure 18, we can show that, for a parameter λ with $\lambda^- < \lambda < \lambda^*$, there is a maximum s-t flow f_λ in $H_{N(\lambda)(\lambda^-,\,\lambda^+)}(s,t)$ whose restriction to $H_{N(\lambda)(\lambda^*,\,\lambda^+)}(s,t)$ is also a maximum s-t flow in $H_{N(\lambda)(\lambda^*,\,\lambda^+)}(s,t)$ (note that λ is not in the interval $(\lambda^*,\ \lambda^+)$ and that the restriction of f_λ to $H_{N(\lambda)(\lambda^*,\,\lambda^+)}(s,t)$ can be obatained from the restriction of f_{λ^*} to $H_{N(\lambda^*)(\lambda^*,\,\lambda^+)}(s,t)$ by decreasing flow). Thus, we can write the minimum s-t cut $(Y_\lambda, \overline{Y_\lambda})$ in $H_{N(\lambda)(\lambda^-,\,\lambda^+)}(s,t)$ defined by Eq.(14) by using the minimum s-t cut $(Y'_\lambda, \overline{Y'_\lambda})$ in $H_{N(\lambda)(\lambda^-,\,\lambda^*)}(s,t)$ as follows [4] :

$$Y_\lambda = Y'_\lambda, \quad \overline{Y_\lambda} = \overline{Y'_\lambda} + \overline{Y_{\lambda^*}}.$$

Let $y'_\lambda(\lambda) = \mathrm{capa}'(Y'_\lambda, \overline{Y'_\lambda})$ be the capacity of $(Y'_\lambda, \overline{Y'_\lambda})$ in $H_{N(\lambda)(\lambda^-,\,\lambda^*)}(s,t)$, i.e.,

$$y'_\lambda(\lambda) = \mathrm{capa}'(Y'_\lambda, \overline{Y'_\lambda}) = \lambda|\overline{Y'_\lambda} \cap \mathcal{C}_N| \; + \sum_{I_\ell \in Y'_\lambda \cap \mathcal{I}_N} \mathrm{capa}(I_\ell, t). \tag{26}$$

Thus, for a parameter λ with $\lambda^- < \lambda < \lambda^*$, the capacity $y_\lambda(\lambda)$ of $(Y_\lambda, \overline{Y_\lambda})$ in $H_{N(\lambda)(\lambda^-,\,\lambda^+)}(s,t)$

[4] $\overline{Y'_\lambda}$ is the set of vertices v of the residual network $H_{N(\lambda)(\lambda^-,\,\lambda^*)}(s,t)(f'_\lambda)$ with respect to a maximum s-t flow f'_λ in $H_{N(\lambda)(\lambda^-,\,\lambda^*)}(s,t)$ such that there is a path from v to t in $H_{N(\lambda)(\lambda^-,\,\lambda^*)}(s,t)(f'_\lambda)$.

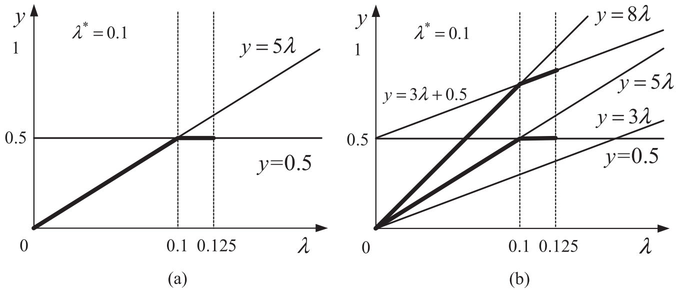


Figure 22: For a parameter λ with $\lambda^- < \lambda < \lambda^*$, (a) the capacity $y'_\lambda(\lambda) = \mathrm{capa}'(Y'_\lambda, \overline{Y'_\lambda})$ of $(Y'_\lambda, \overline{Y'_\lambda})$ in $H_{N(\lambda)(\lambda^-,\,\lambda^*)}(s,t)$, and (b) the capacity $y_\lambda(\lambda)$ of $(Y_\lambda, \overline{Y_\lambda})$ in $H_{N(\lambda)(\lambda^-,\,\lambda^+)}(s,t)$ is given by $y_\lambda(\lambda) = \mathrm{capa}(Y_\lambda, \overline{Y_\lambda}) = \mathrm{capa}'(Y'_\lambda, \overline{Y'_\lambda}) + \lambda|\overline{Y_{\lambda^*}} \cap \mathcal{C}_N| = y'_\lambda(\lambda) + \lambda|\overline{Y_{\lambda^*}} \cap \mathcal{C}_N|$.

is written by

$$\begin{aligned}
y_\lambda(\lambda) &= \mathrm{capa}(Y_\lambda, \overline{Y_\lambda}) \\
&= \lambda|(\overline{Y'_\lambda} + \overline{Y_{\lambda^*}}) \cap \mathcal{C}_N| \;+ \sum_{I_\ell \in Y'_\lambda \cap \mathcal{I}_N} \mathrm{capa}(I_\ell, t) \\
&= \lambda|\overline{Y'_\lambda} \cap \mathcal{C}_N| \;+\; \lambda|\overline{Y_{\lambda^*}} \cap \mathcal{C}_N| \;+ \sum_{I_\ell \in Y'_\lambda \cap \mathcal{I}_N} \mathrm{capa}(I_\ell, t) \\
&= y'_\lambda(\lambda) + \lambda|\overline{Y_{\lambda^*}} \cap \mathcal{C}_N|.
\end{aligned}$$

That is, for a parameter λ with $\lambda^- < \lambda < \lambda^*$, by using the capacity $y'_\lambda(\lambda) = \mathrm{capa}'(Y'_\lambda, \overline{Y'_\lambda})$ of $(Y'_\lambda, \overline{Y'_\lambda})$ in $H_{N(\lambda)(\lambda^-,\,\lambda^*)}(s,t)$, the capacity $y_\lambda(\lambda)$ of $(Y_\lambda, \overline{Y_\lambda})$ in $H_{N(\lambda)(\lambda^-,\,\lambda^+)}(s,t)$ can be written as follows:

$$y_\lambda(\lambda) = \mathrm{capa}(Y_\lambda, \overline{Y_\lambda}) = \mathrm{capa}'(Y'_\lambda, \overline{Y'_\lambda}) + \lambda|\overline{Y_{\lambda^*}} \cap \mathcal{C}_N| = y'_\lambda(\lambda) + \lambda|\overline{Y_{\lambda^*}} \cap \mathcal{C}_N|. \qquad (27)$$

See Figure 22 (a) and (b).

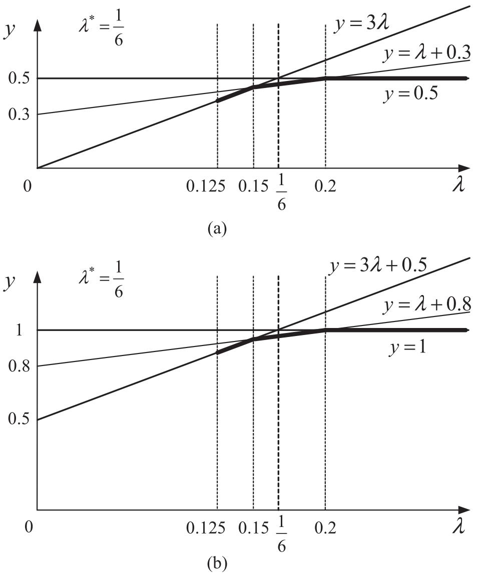
Similarly, since f_{λ^*} is a maximum s-t flow in $H_{N(\lambda^*)(\lambda^-,\,\lambda^+)}(s,t)$ and the restriction of f_{λ^*} to $H_{N(\lambda^*)(\lambda^-,\,\lambda^*)}(s,t)$ is also a maximum s-t flow in $H_{N(\lambda^*)(\lambda^-,\,\lambda^*)}(s,t)$ by the structure of $H_{N(\lambda^*)(\lambda^-,\,\lambda^+)}(s,t)$ as shown in Figure 18, we can show that, for a parameter λ with $\lambda^* < \lambda < \lambda^+$, there is a maximum s-t flow f_λ in $H_{N(\lambda)(\lambda^-,\,\lambda^+)}(s,t)$ whose restriction to $H_{N(\lambda)(\lambda^-,\,\lambda^*)}(s,t)$ is also a maximum s-t flow in $H_{N(\lambda)(\lambda^-,\,\lambda^*)}(s,t)$ (actually, we can choose the restriction of f_{λ^*} to $H_{N(\lambda^*)(\lambda^-,\,\lambda^*)}(s,t)$). Thus, we can write the minimum s-t cut $(Y_\lambda, \overline{Y_\lambda})$ in $H_{N(\lambda)(\lambda^-,\,\lambda^+)}(s,t)$ by using the minimum s-t cut $(Y''_\lambda, \overline{Y''_\lambda})$ in $H_{N(\lambda)(\lambda^*,\,\lambda^+)}(s,t)$ as follows [5] :

$$Y_\lambda = Y''_\lambda + Y_{\lambda^*}, \quad \overline{Y_\lambda} = \overline{Y''_\lambda}.$$

Let $y''_\lambda(\lambda) = \mathrm{capa}''(Y''_\lambda, \overline{Y''_\lambda})$ be the capacity of $(Y''_\lambda, \overline{Y''_\lambda})$ in $H_{N(\lambda)(\lambda^*,\,\lambda^+)}(s,t)$, i.e.,

$$y''_\lambda(\lambda) = \mathrm{capa}''(Y''_\lambda, \overline{Y''_\lambda}) = \lambda|\overline{Y''_\lambda} \cap \mathcal{C}_N| \;+ \sum_{I_\ell \in Y''_\lambda \cap \mathcal{I}_N} \mathrm{capa}(I_\ell, t). \qquad (28)$$

[5] $\overline{Y''_\lambda}$ is the set of vertices v of the residual network $H_{N(\lambda)(\lambda^*,\,\lambda^+)}(s,t)(f''_\lambda)$ with respect to a maximum s-t flow f''_λ in $H_{N(\lambda)(\lambda^*,\,\lambda^+)}(s,t)$ such that there is a path from v to t in $H_{N(\lambda)(\lambda^*,\,\lambda^+)}(s,t)(f''_\lambda)$.



(a)

(b)

Figure 23: For a parameter λ with $\lambda^* < \lambda < \lambda^+$, (c) the capacity $y_\lambda''(\lambda) = \mathrm{capa}''(Y_\lambda'', \overline{Y_\lambda''})$ of $(Y_\lambda'', \overline{Y_\lambda''})$ in $H_{N(\lambda)(\lambda^*,\lambda^+)}(s,t)$, and (d) the capacity $y_\lambda(\lambda)$ of $(Y_\lambda, \overline{Y_\lambda})$ in $H_{N(\lambda)(\lambda^-,\lambda^+)}(s,t)$ is given by $y_\lambda(\lambda) = \mathrm{capa}(Y_\lambda, \overline{Y_\lambda}) = \mathrm{capa}''(Y_\lambda'', \overline{Y_\lambda''}) + \lambda|\overline{Y_{\lambda^*}} \cap \mathcal{C}_N| = y_\lambda''(\lambda) + \sum_{I_\ell \in Y_{\lambda^*} \cap \mathcal{I}_N} \mathrm{capa}(I_\ell, t)$.

Thus, for a parameter λ with $\lambda^* < \lambda < \lambda^+$, the capacity $y_\lambda(\lambda)$ of $(Y_\lambda, \overline{Y_\lambda})$ in $H_{N(\lambda)(\lambda^-,\lambda^+)}(s,t)$ is written by

$$\begin{aligned} y_\lambda(\lambda) &= \mathrm{capa}(Y_\lambda, \overline{Y_\lambda}) \\ &= \lambda|\overline{Y_\lambda''} \cap \mathcal{C}_N| \;+ \sum_{I_\ell \in (Y_\lambda'' + Y_{\lambda^*}) \cap \mathcal{I}_N} \mathrm{capa}(I_\ell, t) \\ &= \lambda|\overline{Y_\lambda''} \cap \mathcal{C}_N| \;+ \sum_{I_\ell \in Y_\lambda'' \cap \mathcal{I}_N} \mathrm{capa}(I_\ell, t) \;+ \sum_{I_\ell \in Y_{\lambda^*} \cap \mathcal{I}_N} \mathrm{capa}(I_\ell, t) \\ &= y_\lambda''(\lambda) \;+ \sum_{I_\ell \in Y_{\lambda^*} \cap \mathcal{I}_N} \mathrm{capa}(I_\ell, t). \end{aligned}$$

That is, for a parameter λ with $\lambda^* < \lambda < \lambda^+$, by using the capacity $y_\lambda''(\lambda) = \mathrm{capa}''(Y_\lambda'', \overline{Y_\lambda''})$ of $(Y_\lambda'', \overline{Y_\lambda''})$ in $H_{N(\lambda)(\lambda^*,\lambda^+)}(s,t)$, the capacity $y_\lambda(\lambda)$ of $(Y_\lambda, \overline{Y_\lambda})$ in $H_{N(\lambda)(\lambda^-,\lambda^+)}(s,t)$ can be written as follows:

$$y_\lambda(\lambda) = \mathrm{capa}(Y_\lambda, \overline{Y_\lambda}) = \mathrm{capa}''(Y_\lambda'', \overline{Y_\lambda''}) + \lambda|\overline{Y_{\lambda^*}} \cap \mathcal{C}_N| = y_\lambda''(\lambda) + \sum_{I_\ell \in Y_{\lambda^*} \cap \mathcal{I}_N} \mathrm{capa}(I_\ell, t). \quad (29)$$

See Figure 23 (a) and (b).

This completes our proof of lemma. □

By Lemma 4.2 and Theorem 3.1, we have the following theorem.

Theorem 4.1 For 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 $\bigcup_{C_i \in \mathcal{C}_N} C_i = C$, Parametric Flow Algorithm (ALGORITHM 2) not only correctly finds $(s_i : i \in N)$ 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 $\sum_{i \in N} A'_i = C$, but also actually 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 $\sum_{i \in N} A_i = C$ in $O(n^2 \log n)$ time using at most $2n - 2$ cuts on cake C.

Thus, Parametric Flow Algorithm (ALGORITHM 2) is envy-free. We can also show that Parametric Flow Algorithm (ALGORITHM 2) is truthful by an argument which is much simpler than the argument in [3]. We omit the details.

5 Application to Mechanism of Chen et al. [16]

By Theorem 3.1, 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 $\sum_{i \in N} A'_i = C$. Thus, Theorem 3.1 can be applied to the mechanism of Chen, et al. for the cake-cutting problem when the valuation function v_i of each player $i \in N$ is piecewise uniform [16]: Given a cake $C = [0, 1)$, n players $N = \{1, 2, \ldots, n\}$ and solid piecewise 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 $m_i \geq 1$ maximal contiguous intervals in C (i.e., $D(v_i) = \sum_{j=1}^{m_i} C_{i_j}$ where each C_{i_j} is a maximal contiguous interval in C) and $\bigcup_{i \in N} D(v_i) = C$.

The mechanism of Chen, et al. [16] finds an envy-free allocation $A'_N = (A'_i : i \in N)$ such that $\sum_{i \in N} A'_i = C$ and $A'_i = \sum_{j=1}^{m_i} A'_{i_j}$ with $A'_{i_j} \subseteq C_{i_j}$ for each $i \in N$ and for each $j = 1, 2, \ldots, m_i$ (we give an outline of Mechanism of Chen et al. below). Thus, we can set $s_{i_j} = len(A'_{i_j})$ and apply Theorem 3.1 to obtain an envy-free allocation $A_N = (A_i : i \in N)$ such that $A_i = \sum_{j=1}^{m_i} A_{i_j}$ for each $i \in N$ with $A_{i_j} \subseteq C_{i_j}$ and $len(A_{i_j}) = s_{i_j}$ for each $j = 1, 2, \ldots, m_i$ with at most $2(\sum_{i \in N} m_i) - 2$ cuts. Note that, we can delete all C_{i_j} if $s_{i_j} = len(A'_{i_j}) = 0$, and thus, we can assume $s_{i_j} = len(A'_{i_j}) > 0$ for each $i \in N$ and for each $j = 1, 2, \ldots, m_i$, without loss of generality.

In summary, we have the following corollary.

Collorary 5.1 Suppose that we are given $(s_{i_j} : i \in N, j = 1, 2, \ldots, m_i)$ such that there is an envy-free allocation $A'_N = (A'_i : i \in N)$ to players N satisfying $\sum_{i \in N} A'_i = C$ and $A'_i = \sum_{j=1}^{m_i} A'_{i_j}$ with $A'_{i_j} \subseteq C_{i_j}$ and $len(A'_{i_j}) = s_{i_j} > 0$ for each $i \in N$ and for each $j = 1, 2, \ldots, m_i$ for the cake-cutting problem with cake $C = [0, 1)$, n players $N = \{1, 2, \ldots, n\}$ and solid piecewise uniform valuation functions $(v_i : i \in N)$ such that $D(v_i) = \{x \in C \mid v_i(x) > 0\}$ of each piecewise uniform valuation function v_i consists of $m_i \geq 1$ maximal contiguous intervals $C_{i_1}, \ldots, C_{i_{m_i}}$ in C and $\bigcup_{i \in N} D(v_i) = C$ (such $A'_N = (A'_i : i \in N)$ to players N can be obtained, for example, by Mechanism of Chen, et al. [16]). Then, Core Mechanism $\mathcal{M}_1$ (ALGORITHM 1) correctly finds an allocation $A_N = (A_i : i \in N)$ with $\sum_{i \in N} A_i = C$ and $A_i = \sum_{j=1}^{m_i} A_{i_j}$ with $A_{i_j} \subseteq C_{i_j}$ and $len(A_{i_j}) = s_{i_j}$ for each $i \in N$ and for each $j = 1, 2, \ldots, m_i$ in $O(\sum_{i \in N} m_i \log \sum_{i \in N} m_i)$ time. Furthermore, the number of cuts made by $\mathcal{M}_1$ on cake C is at most $2(\sum_{i \in N} m_i) - 2$. Thus, Mechanism of Chen, et al. [16] can be implemented to make at most $2(\sum_{i \in N} m_i) - 2$ cuts on cake C.

5.1 An Outline of Mechanism of Chen et al. [16]

In this subsection, we give a brief outline of the envy-free and truthful mechanism proposed by Chen, Lai, Parkes, and Procaccia for the cake-cutting problem where the valuation function of each player is piecewise uniform by borrowing their description in [16].

Thus, we are given a divisible heterogeneous cake C, n players $N = \{1, 2, \ldots, n\}$ with piecewise uniform valuation v_i over C for each player $i \in N$ with $D(v_i) = \sum_{j=1}^{m_i} C_{i_j}$ such that each C_{i_j} is a maximal contiguous interval in C and $\bigcup_{i \in N} D(v_i) = C$, and the mechanism proposed by Chen, et al. [16] finds an allocation $A_N = (A_i : i \in N)$ of C to the players N such that $A_i = \sum_{j=1}^{m_i} A_{i_j}$ with $A_{i_j} \subseteq C_{i_j}$ for each $i \in N$ and each $j = 1, 2, \ldots, m_i$ and $\sum_{i \in N} A_i = C$ and that $A_N = (A_i : i \in N)$ is envy-free and truthful.

We denote by $\mathcal{V}_N$ the (multi-) set of piecewise uniform valuations of all the players N, i.e., $\mathcal{V}_N = (v_1, \ldots, v_n)$. We also write $\mathcal{V}_N = (v_i : i \in N)$. For a subset P of players N and a piece X of cake C (thus, $P \subseteq N$ and X is a set of maximal disjoint subintervals of C), let $\mathrm{DOM}(P, X)$ denote the set of $x \in X$ such that there are at least one player $i \in P$ with $x \in D(v_i)$, i.e.,

$$\mathrm{DOM}(P, X) = \{x \in X \mid x \in D(v_i) \text{ for some } i \in P\}.$$

Note that $\mathrm{DOM}(P, X)$ is a set of maximal disjoint subintervals of C, i.e., a piece of C. Actually,

$$\mathrm{DOM}(P, X) = (\bigcup_{i \in P} D(v_i)) \cap X.$$

Define $\mathrm{avg}(P, X)$ by

$$\mathrm{avg}(P, X) = \frac{len(\mathrm{DOM}(P, X))}{|P|}.$$

Thus, $\mathrm{avg}(P, X)$ is the average length of pieces of the players in P when the piece $\mathrm{DOM}(P, X)$ of cake C is divided among the players in P.

Mechanism of Chen et al. [16] for cake C, n players $N = \{1, 2, \ldots, n\}$ with piecewise uniform valuation v_i over C for each player $i \in N$ is a recursive mechanism that finds a subset of players with a certain property, makes the allocation decision for the subset, and then makes a recursive call on the remaining players and the remaining intervals. Specifically, for a given set of players $P \subseteq N$ and a remaining piece D of cake to be allocated, the mechanism finds the subset $P' \subseteq P$ of players with the minimum $\mathrm{avg}(P', D)$. Then the mechanism finds an allocation $A_{P'} = (A_i : i \in P')$ of $\mathrm{DOM}(P', D)$ to players P' such that $A_i \subseteq D(v_i)$ and $len(A_i) = \mathrm{avg}(P', D)$ for each $i \in P'$ and $\sum_{i \in P'} A_i = \mathrm{DOM}(P', D)$. The mechanism is recursively called on the remaining players $P \setminus P'$ and the remaining intervals, i.e., $D \setminus \mathrm{DOM}(P', D)$.

ALGORITHM 3: Mechanism of Chen et al. [16]

```
Input: A cake $C = [0, 1)$, $n$ players $N = \{1, 2, \ldots, n\}$ and solid piecewise uniform
       valuations $\mathcal{V}_N = (v_i : i \in N)$ with piecewise uniform valuation $v_i$ over $C$ for
       each player $i \in N$.
Output: Allocation $A_N = (A_i : i \in N)$ to players $N$ with $A_i \subseteq D(v_i)$ for each $i \in N$.
ChenCutCake($N, C, \mathcal{V}_N$);
```

To find a maximal subset $P_{\min} \subseteq P$ such that $\mathrm{avg}(P_{\min}, D) = \min_{P' \subseteq P} \mathrm{avg}(P', D)$, Chen, et al. considered a flow network arising piecewise uniform valuations $\mathcal{V}_N = (v_i : i \in$

```
Procedure ChenCutCake($P, D, \mathcal{V}_N$)
find a maximal subset $P_{\min} \subseteq P$ such that $\mathrm{avg}(P_{\min}, D) = \min_{P' \subseteq P} \mathrm{avg}(P', D)$;
find an allocation $A_{P_{\min}} = (A_i : i \in P_{\min})$ of $\mathrm{DOM}(P_{\min}, D)$ to players $P_{\min}$ with
    $A_i \subseteq D(v_i)$ and $len(A_i) = \mathrm{avg}(P_{\min}, D)$ for each $i \in P_{\min}$ and
    $\sum_{i \in P_{\min}} A_i = \mathrm{DOM}(P_{\min}, D)$;
$P = P \setminus P_{\min}$; $D = D \setminus \mathrm{DOM}(P_{\min}, D)$;
if $P \neq \emptyset$ then ChenCutCake($P, D, \mathcal{V}_N$);
```

N) [16]. We give a flow network $H_{N(\lambda)}(s,t)$ which is almost the same as the network used in [16].

Let X_N be the set of all endpoints $\alpha_{i_j}, \beta_{i_j}$ of intervals $C_{i_j} = [\alpha_{i_j}, \beta_{i_j})$ of the piecewise uniform valuations v_i in $\mathcal{V}_N = (v_i : i \in N)$ such that $D(v_i) = C_{i_1} + \cdots + C_{i_{m_i}}$ (i.e., $D(v_i) = (C_{i_j} : j = 1, \ldots, m_i)$) for each $i \in N$ and we assume the elements in X_N are sorted

$$x_0 < x_1 < \cdots < x_m$$

where $x_0 = 0$, $x_m = 1$ and $m \leq \sum_{i \in N} 2m_i - 1$. For each ℓ with $1 \leq \ell \leq m$, let $I_\ell = [x_{\ell-1}, x_\ell)$ and let

$$\mathcal{I}_N = (I_\ell : 1 \leq \ell \leq m)$$

(Figure 24(a)). Let $\mathcal{C}_N = (C_{i_j} : i \in N, j = 1, \ldots, m_i)$ and 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 $(C_{i_j}, I_\ell) \in E_N$ if and only if $I_\ell \subseteq C_{i_j}$. $G_N = (\mathcal{C}_N, \mathcal{I}_N, E_N)$ is called a *convex bipartite graph* since it has a property that if $(C_{i_j}, I_\ell), (C_{i_j}, I_{\ell'}) \in E_N$ with $\ell < \ell'$ then $(C_{i_j}, I_{\ell''}) \in E_N$ for each ℓ'' with $\ell < \ell'' < \ell'$.

Let $G_N(s,t)$ be the directed graph obtained from $G_N = (\mathcal{C}_N, \mathcal{I}_N, E_N)$ by adding new vertices s, t and v_i ($i = 1, \ldots, n$) and directed edges (s, v_i) ($i \in N$), (I_ℓ, t) ($\ell = 1, \ldots, m$) and $(v_i, C_{i_j}), (C_{i_j}, v_i)$ ($i \in N,\ j = 1, \ldots, m_i$). We consider each edge $(C_{i_j}, I_\ell) \in E_N$ is directed from C_{i_j} to I_ℓ (Figure 24). 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 $G_N(s,t)$, respectively. Thus,

$$\begin{aligned} V_N(s,t) &= V_N + \{s,t\} = \mathcal{C}_N + \mathcal{I}_N + \{s,t\} + \mathcal{V}_N, \\ E_N(s,t) &= E_N + \{(s, v_i) \mid v_i \in \mathcal{V}_N\} + \{(I_\ell, t) \mid I_\ell \in \mathcal{I}_N\} \\ &\quad + \{(v_i, C_{i_j}), (C_{i_j}, v_i) \mid i \in N,\ j = 1, \ldots, m_i\}. \end{aligned}$$

Then the flow network $H_{N(\lambda)}(s,t)$ can be obtained from $G_N(s,t)$ by defining the capacity of each directed edge of $G_N(s,t)$ as follows. Each directed edge (s, v_i) ($i \in N$) has capacity λ with parameter $0 \leq \lambda \leq 1$, both directed edges $(v_i, C_{i_j}), (C_{i_j}, v_i)$ ($i \in N,\ j = 1, \ldots, m_i$) have capacity $len(C_{i_j})$, each directed edge (I_ℓ, t) ($\ell = 1, \ldots, m$) has capacity $len(I_\ell)$, and each directed edge $(C_{i_j}, I_\ell) \in E_N$ has capacity ∞ (Figure 24(b)).

An s-t flow f in $H_{N(\lambda)}(s,t)$ is called a *parametric flow* in $H_{N(\lambda)}(s,t)$. The parameter λ is closely related to $\mathrm{avg}(P, X)$. Actually, we have the following.

For a maximum flow f_λ in $H_{N(\lambda)}(s,t)$, let $\overline{Y_\lambda}$ be the set of vertices v such that there is a v-t path in $H_{N(\lambda)}(s,t)(f_\lambda)$, which is the residual network of $H_{N(\lambda)}(s,t)$ with respect to f_λ, 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

$$Y'_\lambda \subseteq Y_\lambda$$

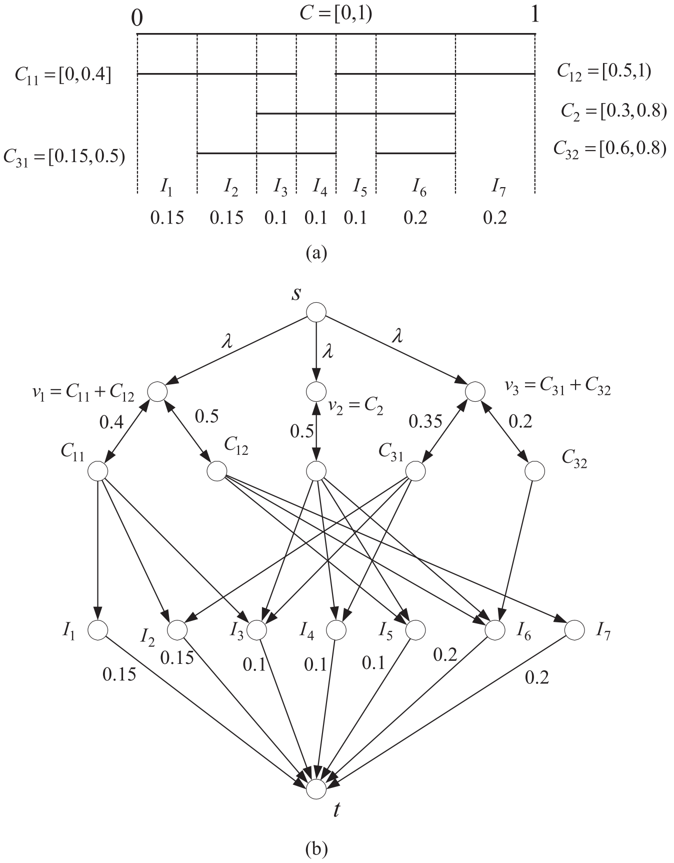


Figure 24: (a) Example of piecewise uniform valuations $\mathcal{V}_N = (v_i : i \in N)$ with $D(v_1) = C_{11} + C_{12}$, $D(v_2) = C_2$ and $D(v_3) = C_{31} + C_{32}$ and $\mathcal{I}_N = (I_1, \ldots, I_7)$. (b) $G_N(s,t)$ and flow network $H_{N(\lambda)}(s,t)$ on $\mathcal{V}_N = (v_i : i \in N)$.

(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, Y_λ is a maximum set ($\overline{Y_\lambda}$ is a minimum set) among the minimum s-t cuts $(Y'_\lambda, \overline{Y'_\lambda})$ in $H_{N(\lambda)}(s,t)$. Furthermore, for two distinct parameters $\lambda' < \lambda$,

$$Y_{\lambda'} \subseteq Y_\lambda$$

(i.e., $\overline{Y_{\lambda'}} \supseteq \overline{Y_\lambda}$) holds. There are at most $n + \sum_{i \in N} m_i + m + 1$ 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 $Y_{\lambda'} \subseteq Y_\lambda$ (i.e., $\overline{Y_{\lambda'}} \supseteq \overline{Y_\lambda}$) holds for two distinct parameters $\lambda' < \lambda$ as described above.

Suppose that there are exactly $K+1$ 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 $\lambda_1, \lambda_2, \ldots, \lambda_K$ be the breakpoints where minimum s-t cuts $(Y_\lambda, \overline{Y_\lambda})$ in $H_{N(\lambda)}(s,t)$ change. Assume

$$\lambda_0 = 0 < \lambda_1 < \lambda_2 < \cdots < \lambda_K \leq 1 = \lambda_{K+1}, \tag{30}$$

where we consider $\lambda_0 = 0$ and $\lambda_{K+1} = 1$, for convenience. Note that $Y_\lambda = \{s\}$ for $0 \leq \lambda < \lambda_1$ and $\overline{Y_\lambda} = \{t\}$ for $\lambda_K \leq \lambda \leq \lambda_{K+1} = 1$.

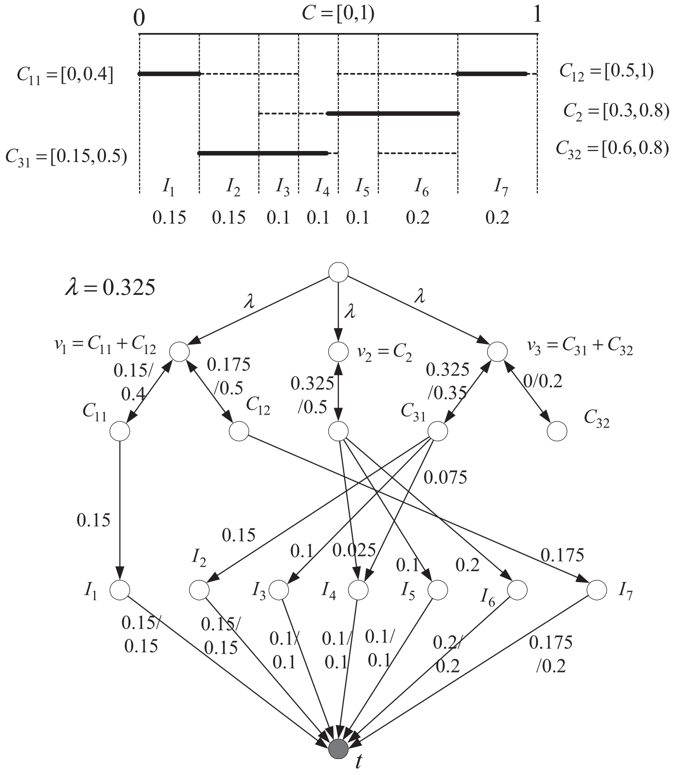


Figure 25: A maximum s-t flow f_λ and the minimum s-t cut $(Y_\lambda, \overline{Y_\lambda})$ with parameter $\lambda = 0.325$ in network $H_{N(\lambda)}(s,t)$ in Figure 24.

Figure 25 shows a maximum s-t flow f_λ with parameter $\lambda = 0.325$ in network $H_{N(\lambda)}(s,t)$ in Figure 24 and Figure 26 shows the residual network $H_{N(\lambda)}(s,t)(f_\lambda)$ for the maximum s-t flow f_λ in Figure 25 and the minimum s-t cut $(Y_\lambda, \overline{Y_\lambda})$ in network $H_{N(\lambda)}(s,t)$ in Figure 24. In this example,

$$\text{avg}(N_{\min}, C) = \min_{N' \subseteq N} \text{avg}(N', C) = \lambda_1 = 0.325$$

and $N_{\min} = \{2, 3\}$, and $K = 2$ and $\lambda_2 = 0.35$.

Let Procedure ChenCutCake$(P, D, \mathcal{V}_N)$ be called K' times in Mechanism of Chen et al. (ALGORITHM 3). Then, it can be shown that $K' = K$ and, in the kth call of ChenCutCake$(P, D, \mathcal{V}_N)$ which is denoted by ChenCutCake$(P^{(k)}, D^{(k)}, \mathcal{V}_N)$,

$$\text{avg}(P^{(k)}_{\min}, D^{(k)}) = \min_{P' \subseteq P^{(k)}} \text{avg}(P', D^{(k)}) = \lambda_k$$

holds for each $k = 1, \ldots, K$.

Thus, we can set $s_{i_j} = f_{\lambda_k}(v_i, C_{i_j})$ for each $j = 1, \ldots, m_i$ if $v_i \in P^{(k)}_{\min}$ for each $i \in N$ and for each $k = 1, \ldots, K$. This implies that by applying $\mathcal{M}_1$ (ALGORITHM 1), we can make the envy-free and truthful mechanism proposed by Chen, et al. [16] use at most $2\sum_{i \in N} m_i - 2$ cuts, where m_i is the number of maximal contiguous intervals in $D(v_i) = \{x \in C \mid v_i(x) > 0\}$ of each piecewise uniform valuation v_i.

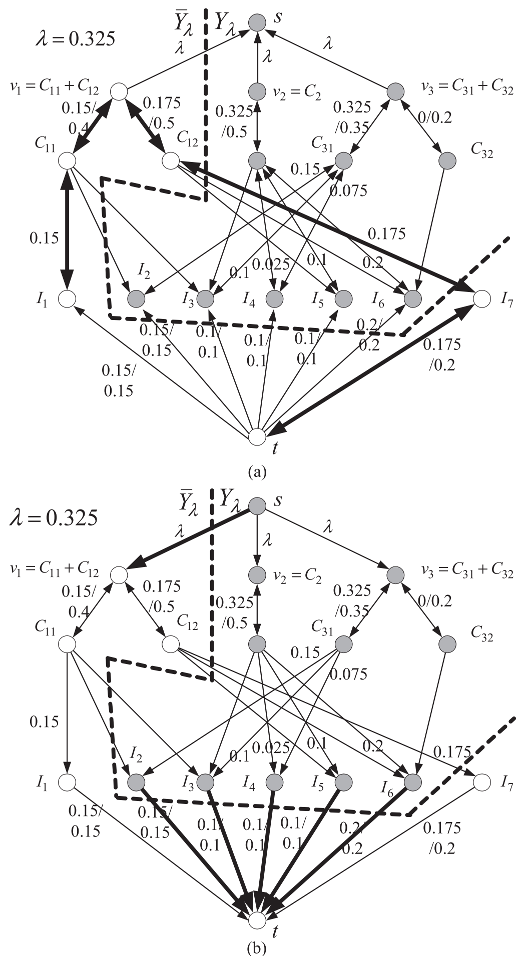


Figure 26: (a) The residual network $H_{N(\lambda)}(s,t)(f_\lambda)$ for the maximum s-t flow f_λ in Figure 25 and (b) the minimum s-t cut $(Y_\lambda, \overline{Y_\lambda})$ in network $H_{N(\lambda)}(s,t)$ in Figure 24.

6 Concluding Remarks

We gave a much simpler envy-free and truthful mechanism with a small number of cuts for the cake-cutting problem posed in [2, 30]. Furthermore, we showed that this approach can be applied to the envy-free and truthful mechanism proposed by Chen, et al. for the more general cake-cutting problem where the valuation function of each player is piecewise uniform [16] based on parametric flows on a network arising from piecewise uniform valuations v_i and can make their envy-free and truthful mechanism use $2\sum_{i\in N} m_i - 2$ cuts, where m_i is the number of maximal contiguous intervals in $D(v_i) = \{x \in C \mid v_i(x) > 0\}$ of each piecewise uniform valuation v_i.

If we require the piecewise uniform valuation v_i of each player i to be a single contiguous interval C_i in cake C, then parametric flows on the network arising from valuation intervals C_i can be found efficiently. Thus, Mechanism of Asano and Umeda in [4] can be implemented to run in $O(n^2 \log n)$ time.

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] S. Athanassoglou and J. Sethuraman, House allocation with fractional endowments, *International Journal of Game Theory*, Vol. 40, pp.481–513, 2011.

[6] 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.

[7] H. Aziz and S. Mackenzie, A discrete and bounded envy-free cake cutting protocol for any number of agents, *Proc. of 57th IEEE Symposium on Foundations of Computer Science*, pp. 416–427, 2016.

[8] J.B. Barbanel and S.J. Brams, Cake division with minimal cuts: envy-free procedures for three persons, four persons, and beyond, *Mathematical Social Sciences*, Vol.48, pp.251–269, 2004.

[9] X. Bei, N. Chen, X. Hua, B. Tao, and E. Yang, Optimal proportional cake cutting with connected pieces, *Proc. of 26th AAAI Conference on Artificial Intelligence*, pp. 1263–1269, 2012.

[10] X. Bei, N. Chen, G. Huzhang, B. Tao, and J. Wu, Cake cutting: envy and truth, *Proc. of 26th International Joint Conference on Artificial Intelligence*, pp.3625–3631, 2017.

[11] A. Bogomolnaia and H. Moulin, A new solution to the random assignment problem, *Journal of Economic Theory*, Vol.100, pp.295–328, 2001.

[12] S.J. Brams, M. Feldman, J.K. Lai, J. Morgenstern, and A. D. Procaccia, On maxsum fair cake divisions, *Proc. of 26th AAAI Conference on Artificial Intelligence*, pp. 1285–1291, 2012.

[13] S.J. Brams, M. Jones, and C. Klamler, Proportional pie-cutting, *International Journal of Game Theory*, Vol. 36, pp.353–367, 2008.

[14] S.J. Brams and A.D. Taylor, *Fair Division: From Cake Cutting to Dispute Resolution*, Cambridge University Press, 1996.

[15] I. Caragiannis, J.K. Lai, and A.D. Procaccia, Towards more expressive cake cutting, *Proc. of 22nd International Joint Conference on Artificial Intelligence*, pp.127–132, 2011.

[16] 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.

[17] X. Deng, Q. Qi, and A. Saberi, Algorithmic solutions for envy-free cake cutting, *Operations Research*, Vol.60, pp. 1461–1476, 2012.

[18] J. Edmonds and K. Pruhs, Cake cutting really is not a piece of cake, *Proc. of 17th ACM-SIAM Symposium on Discrete Algorithms*, pp.271–278, 2006.

[19] J. Edmonds and K. Pruhs, Balanced allocations of cake, *Proc. of 47th IEEE Symposium on Foundations of Computer Science*, pp.623–634, 2006.

[20] S. Even and A. Paz, A note on cake-cutting, *Discrete Applied Mathematics*, Vol. 7, pp.285–296, 1984.

[21] 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.

[22] P.W. Goldberg, A. Hollender, and W. Suksompong, Contiguous cake cutting: hardness results and approximation algorithms, *Proc. of 34th AAAI Conference on Artificial Intelligence*, pp.1990–1997, 2020.

[23] P. Hall, On representations of subsets, *Journal of London Mathematical Society*, Vol.10, pp.26–30, 1934.

[24] D. Kurokawa, J.K. Lai, and A.D. Procaccia, How to cut a cake before the party ends, *Proc. of 27th AAAI Conference on Artificial Intelligence*, pp.555-561, 2013.

[25] A. Maya and N. Nisan, Incentive compatible two player cake cutting, *Proc. of 8th International Conference on Web and Internet Economics*, pp.170–183, 2012.

[26] A.D. Procaccia, Thou shalt covet thy neighbor's cake, *Proc. of 21st International Joint Conference on Artificial Intelligence*, pp.239–244, 2009.

[27] A.D. Procaccia, Cake cutting: not just child's play, *Communications of ACM*, Vol.56, pp.78–87, 2013.

[28] A.D. Procaccia, Cake cutting algorithms, *Handbook of Computational Social Choice*, F. Brandt et al., Eds. Cambridge University Press, Chapter 13, pp.311-329, 2016.

[29] J.M. Robertson and W.A. Webb, *Cake Cutting Algorithms: Be Fair If you Can*, A.K. Peters, 1998.

[30] 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.

[31] K. Sherstyuk, How to gerrymander: A formal analysis, *Public Choice*, Vol. 95, pp. 27–49, 1998.

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

[33] H. Steinhaus, Sur la division pragmatique, *Econometrica (suppliment)*, Vol. 17, pp. 315–319, 1949.

[34] W. Stromquist, How to cut a cake fairly, *American Mathematical Monthly*, Vol. 87, pp. 640–644, 1980.

[35] W. Stromquist, Envy-free cake divisions cannot be found by finite protocols, *The Electronic Journal of Combinatorics*, Vol.15, #R11 (pp.1–10), 2008.

[36] F. E. Su, Rental harmony: Sperner's lemma in fair division, *American Mathematical Monthly*, Vol. 106, pp. 930–942, 1999.

[37] W. Thomson, Children crying at birthday parties. Why?, *Economic Theory*, Vol. 31, pp. 501–521, 2007.

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

[39] D.P. Williamson, *Network Flow Algorithms*, Cambridge University Press, 2019.