The Complexity of Envy-Free Graph Cutting

Argyrios Deligkas[1] , **Eduard Eiben**[1] , **Robert Ganian**[2] , **Thekla Hamm**[2] , **Sebastian Ordyniak**[3]
[1]Royal Holloway, University of London, UK
[2]TU Wien, Austria
[3]University of Leeds, UK
{argyrios.deligkas,eduard.eiben}@rhul.ac.uk, {rganian, thekla.hamm,sordyniak}@gmail.com

Abstract

We consider the problem of fairly dividing a set of heterogeneous divisible resources among agents with different preferences. We focus on the setting where the resources correspond to the edges of a connected graph, every agent must be assigned a connected piece of this graph, and the fairness notion considered is the classical envy freeness. The problem is NP-complete, and we analyze its complexity with respect to two natural complexity measures: the number of agents and the number of edges in the graph. While the problem remains NP-hard even for instances with 2 agents, we provide a dichotomy characterizing the complexity of the problem when the number of agents is constant based on structural properties of the graph. For the latter case, we design a polynomial-time algorithm when the graph has a constant number of edges.

1 Introduction

Cake cutting is, without a doubt, among the most influential problems in social choice and has received significant attention in computer science, mathematics, and economics [Brams and Taylor, 1996; Robertson and Webb, 1998; Moulin, 2004; Procaccia, 2013]. The cake corresponds to a heterogeneous divisible resource that is to be divided between a set of n agents with different preferences in a "fair" manner. In this paper, the fairness concept we focus on is *envy-freeness*, where every agent prefers the piece of the cake they get allocated over the piece any other agent gets.

In the classical formulation of the problem, the cake is represented as an interval and the preference of every agent over the cake is given via a *valuation* over any subinterval of the cake. In this setting, Dubins and Spanier (1961) showed that an envy-free allocation *always* exists for arbitrary valuations for the agents, where the piece an agent gets consists of a countable number of subintervals. Stromquist (1980) strengthened this result by removing the possibility of pieces that consist of a "union of crumbs". In fact, he showed that there is an envy-free allocation where the piece of every agent is *contiguous*, i.e., it is a *single* subinterval.

Recently, Bei and Suksompong (2021) considered a generalized version cake cutting on *graphs*. This augmented model allows to capture more general scenarios which cannot be represented by splitting an interval into connected pieces—consider, e.g., the task of splitting road or railway networks between companies. We note that these settings do not always give rise to large graphs: for instance, the ICE train network in Germany can be modeled as a graph with merely 23 edges. Or, for yet another example, suppose that one wants to schedule time on a high-performance computing cluster between teams (agents). Suppose furthermore that the day is partitioned into, e.g., four time-slots, with each time-slot being less or more desirable for different agents. This setting, too, can easily be modeled using a graph with four edges. In these as well as other examples, it is still desirable to ensure that each agent receives a connected piece of the graph, but the natural model for the cake is a graph where each individual edge may be split and behaves as a single, uniform piece.

Observe that depending on the setting, it may either be the case that a vertex is allocated to only a single agent (as in the case of junctions in the division of road networks over maintenance companies), or that a vertex merely acts as a bridge between edges and may be used by multiple agents (as in the case of train stations in division of railway networks over rail companies). We call the former setting "*graph cutting*" and the latter "*vertex-disjoint graph cutting*".

While both fair division problems always admit a contiguous envy free solution when the underlying graph is a path (since they are special cases of the classical cake cutting problem), this is no longer true for more general graphs. In fact, it is not hard to construct graphs with no envy-free solutions; this holds even for stars with three leafs and two agents with identical valuations. Hence, in the setting studied here, the natural task is to *decide if a solution exists, and if it does to efficiently compute one*.

Our Contributions. In this work, we explore the frontiers of tractability for both variants of graph cutting under the envy free solution concept; we refer to these problems as EF-GC and EF-VDGC, respectively. While it is not difficult to show that both problems are NP-complete in general, here we analyze the problem with respect to two of the most natural complexity measures that characterize the input: the number of agents and the number of edges.

When considering the number of agents (i.e., n), we begin by extending the NP-completeness lower bounds for both variants to the case of $n = 2$ even on very simple graphs—as

simple as two vertices plus a matching (Theorem 3). However, here we can show that the two variants do sometimes behave differently: while EF-GC is also NP-hard on stars when $n = 2$ (Theorem 2), we design a polynomial-time algorithm for EF-VDGC on trees when n is upper-bounded by an arbitrary but fixed constant (Theorem 6). In order to achieve tractability for EF-GC, we need to restrict ourselves to instances with a constant number of agents and where the graph is a tree with a constant maximum degree (Theorem 7).

tw	Δ	EF-GC	EF-VDGC
2	3	NP-c (Th. 4)	NP-c (Th. 4)
2	2	P (Th. 8)	P (Th. 8)
1	arbitrary	NP-c (Th. 2)	P (Th. 6)
1	constant	P(Th. 7)	P (Th. 6)

Table 1: Complexity of EF-GC and EF-VDGC for a constant number of agents for different restrictions on the treewidth (tw) and the maximum degree (Δ). All NP-completeness (NP-c) results hold already for only 2 agents.

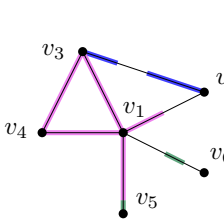
In fact, we prove this is the best one can do from this perspective. Both problems become NP-hard on instances with two agents and graphs of maximum degree 3 which are "almost trees"—in particular, have treewidth 2 (Theorem 4). On the other hand, we show that both problems are polynomial-time solvable on cycles, which are graphs of maximum degree and treewidth 2 (Theorem 8); this provides a complete dichotomy for the complexity of both problems with respect to treewidth and maximum degree (see Table 1).

Next, we target instances where the number of edges is bounded by a constant. As the main technical contribution of this article, we show that both problems under consideration become polynomial-time tractable under this restriction (Theorem 9). The algorithm is non-trivial and combines insights into the structure of a hypothetical solution with branching techniques, linear programming subroutines and insights from multidimensional geometry.

Related Work. Bei and Suksompong (2021) studied graph cutting under the fairness notions of proportionality and equitability; this was the first paper that considered a graph structure with *divisible* items. For indivisible items there are several different graph-based approaches. In the most common modelling scenario the items correspond to the vertices of the graph and each agent must get a connected subgraph [Bei *et al.*, 2019; Bilò *et al.*, 2021; Bouveret *et al.*, 2017; Deligkas *et al.*, 2021; Elkind *et al.*, 2021a; Igarashi and Peters, 2019].

A different line of work uses graphs to denote the relationships between the agents, where an agent compares their bundle only against the bundles of the agents they are connected with [Abebe *et al.*, 2017; Aziz *et al.*, 2018; Bei *et al.*, 2017; Bei *et al.*, 2020; Chevaleyre *et al.*, 2017; Eiben *et al.*, 2020].

In addition to the above, there are many other works that study variants of cake cutting [Elkind *et al.*, 2021c; Elkind *et al.*, 2021b; Segal-Halevi *et al.*, 2016; Menon and Larson, 2017; Marenco and Tetzlaff, 2014; Caragiannis *et al.*, 2011; Bei *et al.*, 2021; Balkanski *et al.*, 2014; Aumann and Dombb, 2015; Brânzei and Miltersen, 2015].



Edge pieces:
pink:$(v_1v_2, [0, \frac{1}{2}])$,$(v_1v_3, [0, 1])$, $(v_1v_4, [0, 1])$, $(v_1v_5, [0, \frac{5}{6}])$, $(v_3v_4, [0, 1])$;
blue: $(v_2v_3, [0, \frac{1}{2}])$, $(v_2v_3, [\frac{3}{4}, 1])$;
green: $(v_1v_5, [\frac{5}{6}, 1])$, $(v_1v_6, [\frac{1}{2}, \frac{3}{4}])$
Only the pink edge pieces form a piece of the graph together; they are adjacent via v_1, v_3 and v_4.

Figure 1: Examples of pieces of edges and of a graph.

2 Preliminaries and Problem Definition

Notation. For rational numbers $i, j \in \mathbb{Q}$, we use the standard notation $[i, j] = \{k \in \mathbb{Q} \mid i \leq k \leq j\}$ for intervals and for an interval $I = [i, j]$ we denote its length by $|I| = \max\{j - i, 0\}$. We denote the set of all non-negative rational numbers by $\mathbb{Q}^+$. Throughout the paper we will consider simple undirected graphs.

Graph Cutting. Consider a set A of n agents, a connected graph $G = (V, E)$, and for each agent $a \in A$ an edge weight function $u_a : E \to \mathbb{Q}^+$ over the edges of G; u_a is the *utility* (function) of agent a, and for a specific edge $e \in E$ we call $u_a(e)$ the *utility* of a for e.

A *piece of an edge* e is a tuple (e, I) where $I \subseteq [0, 1]$ is a possibly empty interval. We assume an arbitrary but fixed order V of the vertices of G, and say that pieces (e, I) and (e', I') of two different edges $e, e' \in E$ are *adjacent* if

- there is some $v \in V$ such that $e = uv$ and $e' = vu'$ (i.e. e and e' are adjacent in G), and
- if u has a smaller index in the ordering of V than v, then $1 \in I$ and $0 \in I$ otherwise, and
- similarly, if u' has a smaller index in the ordering of V than v, then $1 \in I'$ and $0 \in I'$ otherwise.

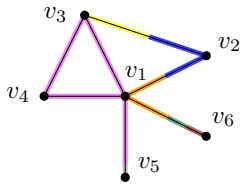
A *piece* of G is a collection P of pieces of edges of e such that for every pair of pieces of edges $(e_0, I_0), (e_\ell, I_\ell)$ in P there is a sequence of pieces of edges $(e_1, I_1), \ldots, (e_{\ell-1}, I_{\ell-1})$ in P such that for all j with $0 \leq j < \ell$, (e_j, I_j) and (e_{j+1}, I_{j+1}) are adjacent. An example is provided in Figure 1.

The utility of an agent a for a piece P is given as $u_a(P) = \sum_{(e,I)\in P} |I| \cdot u_a(e)$. Note that $\{(e, [0, 1]) \mid e \in E\}$ is also a piece of G. As is standard, our algorithms will assume *normalized* utilities, i.e., $u_a(G) = 1$ for all $a \in A$[1].

A *partition* of G into pieces is a set Π of pieces such that for every edge $e \in E$ and every real $0 \leq \alpha \leq 1$, there is precisely one piece $P \in \Pi$ and one $(e, I) \in P$ such that $\alpha \in I$. In some cases it is also necessary to allocate each vertex to a single piece: a partition of G into pieces is *vertex-disjoint* if all pieces of edges containing a vertex belong to the same piece of the graph. See Figure 2 for an example.

Finally we are ready to define our problem of interest. In (piecewise) envy-free graph cutting (EF-GC) we are given agents A, graph G and utilities $u_a : E \to \mathbb{Q}^+$ for each agent

[1]Instances with non-normalized utilities can be trivially transformed into equivalent ones with normalized utilities.



This partition is not vertex-disjoint as v_1 is contained in both the orange and pink piece of the graph. Already if one orange edge piece was pink, the remaining orange edge piece could be given by an open interval, making the partition vertex-disjoint.

Figure 2: Non-vertex-disjoint partition of a graph into seven connected pieces of different colors.

$a \in A$. Our task is to construct a partition Π of G into pieces and a bijection (called an *assignment*) $\pi : A \to \Pi$ such that for every pair of agents $a, a' \in A$, it holds that $u_a(\pi(a)) \geq u_a(\pi(a'))$. This condition is commonly referred to as *envy-freeness* in assignment and allocation problems, and when it is violated for some a and a', we say that a *envies* a'.

We can analogously define the problem of (piecewise) envy-free vertex-disjoint graph cutting (EF-VDGC) by additionally requiring Π to be vertex-disjoint. In fact, by replacing each vertex in an instance of EF-GC with a clique of size $|E(G)|$, we obtain a simple reduction from the former problem to EF-VDGC.

Observation 1. EF-GC *can be reduced to* EF-VDGC *in polynomial time.*

Bounding the Number of Cells in Metric Spaces. One prominent tool our main algorithm for solving EF-GC and EF-VDGC uses is a theorem that applies to the behavior of polynomials in higher-dimensional spaces. To provide a high-level intuition, consider a d-dimensional space that is cut into regions by s-many hyperplanes or, more generally, "well-behaved cuts" defined by bounded-degree polynomials. The combination of these cuts splits the whole space into "cells", each consisting of points that lie on the same side of each of the s-many cuts. While the trivial bound for the number of these cells is 2^s, it can be shown that the number of such cells is in fact polynomial in s for fixed d and that representatives of these cells can be computed efficiently. The result we use here is formalized in the book of Basu *et al.* (2006, Thm.13.22), see also Simonov et al. (2019). ($\star$)

3 Instances with Few Agents

In this section we consider the complexity of EF-GC and EF-VDGC for instances with only a few agents. Interestingly, we will show that while both problems are NP-hard even for instances with only two agents, EF-GC turns out to be significantly harder when additional restrictions are considered for the input graph. In particular, while EF-GC is already NP-hard on stars, EF-VDGC can be solved in polynomial-time (for a fixed number of agents) even on trees and only becomes NP-hard on graphs that have a vertex deletion set into a matching. Moreover, the problem becomes much harder if we relax the graph structure from trees to "tree-like graphs": both problems become NP-hard on graphs that have treewidth 2 and maximum degree 3 (see Figure 3 for an illustration).

All three NP-hardness results follow from polynomial-time reductions from the NP-complete NUMBER PARTITIONING problem: given a multi-set $S = \{s_1, \ldots, s_n\}$ of non-negative integers, decide whether there is a partition of S into two subsets S_1 and S_2 such that $\sum_{s \in S_1} s = \sum_{s \in S_2} s$.

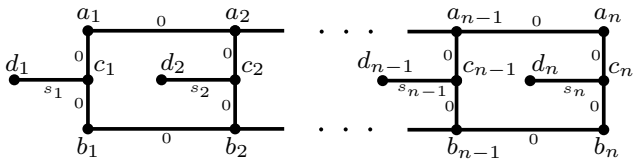


Figure 3: The graph used in the proof of Theorem 4.

Theorem 2 ($\star$)**.** EF-GC *is* NP*-hard even when restricted to instances with two agents where the graph is a star.*

Theorem 3 ($\star$)**.** EF-VDGC *is* NP*-hard even when restricted to instances with two agents where the graph consists of a matching plus two additional vertices.*

Theorem 4 ($\star$)**.** EF-VDGC *and* EF-GC *are* NP*-hard even when restricted to instances with two agents where the graph has treewidth 2 and maximum degree 3.*

Proof Sketch. Given an instance of NUMBER PARTITIONING with n positive integers, we create a graph with $4n$ vertices as follows (see also Fig. 3). For every $i \in [n]$, we construct the vertices a_i, b_i, c_i, and d_i and the edges $(a_i, c_i), (c_i, b_i)$ and (c_i, d_i). In addition, for every $i \in [n-1]$ we create the edges (a_i, a_{i+1}) and (b_i, b_{i+1}). We create two agents with identical valuations: for every $i \in [n]$, both agents value the edge (c_i, d_i) with s_i, while every other edge has value 0. □

Next, we move on to the aforementioned algorithmic result for EF-VDGC. To establish that, we first prove the following technical lemma.

Lemma 5 ($\star$)**.** *Let $\mathcal{I} = (A, G, \{u_a\}_{a \in A})$ be an instance of* EF-GC *(*EF-VDGC*) and let G be a tree, or a cycle, and let F be a set of edges of G. There is an algorithm that runs in time polynomial in $|A|^{|F|+1} \cdot |F|^{|A|} \cdot |\mathcal{I}|$, and either outputs an assignment π that is envy-free such that each connected component of $G - F$ is assigned to exactly one agent (and, for* EF-VDGC*, $\pi(A)$ is additionally a vertex-disjoint partition) or correctly identifies that such an assignment does not exist.*

Proof Sketch. Observe that $G - F$ consists of most $|F| + 1$ connected components and that there are at most $|A|^{|F|+1}$ many assignments of these connected components to agents. By careful branching, we can determine for each agent whether they are (1) assigned to a specific single edge of F and their piece is fully inside some $e \in F$, or (2) are assigned all edges in a subgraph T of G induced by a union of some connected component of $G - F$ and edges of F. For each branch, we design an instance of Linear Programming (LP) and solve it; if the instance has a solution, we can translate it into an assignment π with the desired properties, and otherwise no such assignment exists for the current branch. □

Lemma 5 allows us to solve EF-VDGC on trees by applying initial branching to reach a situation satisfying the preconditions of Lemma 5.

Theorem 6 ($\star$)**.** *For each $c \in \mathbb{N}$,* EF-VDGC *restricted to instances with at most c agents is polynomial-time solvable on trees.*

Proof Sketch. Let π_{SOL} be an arbitrary vertex-disjoint envy-free assignment. Since G is a tree and agents do not share vertices, one can show that there is a set F of at most $|A| - 1$ edges with the following properties: (I) each of the exactly $|F| + 1$ components of $G - F$ is assigned to a single agent, and (II) every edge $e \in F$ is split between the two agents assigned to the components that contain a vertex incident with e plus some of the agents who are not assigned any of these components.

We can now enumerate all of the at most $|V(G)|^{|A|}$ possible sets of edges $F \subseteq E(G)$ such that $|F| \leq |A| - 1$. The theorem then follows by applying Lemma 5 to G and F.

Using the notation and the running time bound from the proof of Lemma 5, it follows that the overall complexity lies in $\mathcal{O}(|V(G)|^{|A|} \cdot |A|^{|A|} \cdot (|A| - 1)^{|A|} \cdot T(|A|^2))$. □

To complete our understanding of EF-GC and EF-VDGC for instances with only boundedly-many agents, we show that both problems are also polynomial-time solvable on graphs of maximum degree 2 (i.e., cycles) and that the former is polynomial-time solvable on bounded-degree trees. These results follow a similar approach as the proof of Theorem 6, which is a combination of initial branching and Lemma 5.

Theorem 7 ($\star$)**.** *For each $c, d \in \mathbb{N}$,* EF-GC *restricted to instances with at most c agents is polynomial-time tractable on trees with maximum degree at most d.*

Theorem 8 ($\star$)**.** *For each $c \in \mathbb{N}$,* EF-GC *and* EF-VDGC *restricted to instances with at most c agents is polynomial-time solvable on cycles.*

4 Instances with Few Edges

In the section we provide an algorithm showing that both EF-GC and EF-VDGC are in XP parametrized by the number of edges in the input graph G.

Theorem 9. EF-GC *and* EF-VDGC *can be solved in time* $|A|^{\mathcal{O}(|E(G)|^2)}$.

The algorithm can be divided into three main steps. We start with a direct brute-force branching over all assignments of agents that span more than one edge, and for these special agents we identify precisely the edges from which they will receive a piece. We also branch to determine the exact number of agents that will be assigned to each edge. This will result in $|A|^{\mathcal{O}(|E(G)|)}$ many initial branches, and each branch already provides useful information about a hypothetical sought-after solution—but not enough to solve the problem. Crucially, every solution to the original instance corresponds to one of the branches.

Our aim in the second step will be to construct, for each branch, a Linear Program (LP) to determine the exact lengths of all the pieces in an envy-free partitioning. In particular, if the branch corresponds to a solution, then we require that the LP outputs a partitioning that can be matched to agents in a way which also produces a solution. Unfortunately, the branching carried out in the previous step is not yet sufficient to construct such an LP: during the construction, we need to apply an additional advanced branching step to identify a small number of *envy-critical* agents that are assigned completely to a single edge. The property of these agents is that they will be the "closest" to envying agents assigned to other edges in the graph, and in the LP these will serve as anchors which ensure that an envy-free assignment will exist as long as the assignment of agents is carried out in a way which respects the selected envy-critical agents. Defining, bounding the number of, and branching on these envy-critical agents is the most challenging part of the algorithm, and is also where Theorem 13.22 [Basu *et al.*, 2006] is used.

Finally, based on the branching decisions and a solution to the constructed LP, we design an instance of bipartite matching that matches the remaining unassigned agents with the pieces given by the LP instance. If a matching exists we are guaranteed to have found a solution; if not, then our branch does not correspond to a valid solution.

Initial Branching. For the remainder of this section, we fix an instance $\mathcal{I}$ of EF-GC or EF-VDGC given by the set of agents A, graph $G = (V, E)$, and utilities $u_a : E \to \mathbb{Q}^+$ for each agent $a \in A$. Denote $k = |E(G)|$. We can now start with the branching phase. Let us assume $\mathcal{I}$ admits an envy-free assignment π_{SOL} into some partition of G into pieces; we will describe the branching as a series of "guesses" of the properties of this solution $\mathcal{I}$ and its interactions with G.

First, observe that for each edge e there are at most 2 agents that can be assigned a piece of the edge e together with a piece of some other edge. These are the agents in π_{SOL} that receive the piece $(e, [0, c])$ and the piece $(e, [d, 1])$ for some constant $c, d \in [0, 1]$. For each edge e, we guess the agent that gets the piece $(e, [0, c])$ (for some unspecified $c \in [0, 1]$) and say that this is the guess for pair $(e, 0)$; analogously, we guess the agent that gets the piece $(e, [d, 1])$ (for some unspecified $d \in [0, 1]$) and say that this is the guess for pair $(e, 1)$. This results in $|A|^{2k}$ many branches. Let A_V be the set of the at most $2k$ agents guessed in the previous step.

All the remaining agents are assigned by π_{SOL} a piece $\{(e, [c, d])\}$ for some edge e. For every such agent, we say that it gets a piece *fully contained inside edge* e. While it is too computationally expensive to guess precisely which agents get a piece fully contained in an edge e, we will guess the number of agents that get a piece fully contained in an edge e. This results in $|A| + 1$ many guesses for each edge, amounting to a branching factor of at most $(|A| + 1)^k$. Let us denote by n_e the number of agents that get a piece fully contained in the edge e. We now perform a set of sanity checks on our branching; in particular, we discard branches which do not fulfil the following conditions:

1. $|A_V| + \sum_{e \in E(G)} n_e = |A|$.
2. For every agent $a \in A_V$, the guesses of pieces assigned to a form a connected subset of G. More formally, whenever our branch assigns an agent a to two distinct pairs (e, i) and (f, j), where $e, f \in E(G)$ and $i, j \in \{0, 1\}$, there must exist a path $P = e_1 e_2 \dots e_q$ from $e[i]$ to $f[j]$ such that for each $\iota \in [q]$ it holds that

(I) $n_{e_\iota} = 0$ and (II) the agent a is also the guess for both $(e_\iota, 0)$ and $(e_\iota, 1)$.

3. In the case of EF-VDGC, we will also verify that the branching corresponds to a vertex-disjoint partition. In particular, for each vertex v let $E_{v,0}$ be the set of edges incident to v and a vertex preceding v in the ordering and $E_{v,1}$ be the set of edges incident to v and a vertex succeeding v in the ordering. We check that there is a single agent a such that for each edge $e \in E_{v,0}$, a is the guess for $(e, 0)$, and at the same time for each edge $e' \in E_{v,1}$, a is the guess for $(e', 1)$.

Linear Programming. We can now begin describing the instance of LP that we will use to determine the partition. For this, it will be useful to observe that agents fully contained in the same edge must receive a segment of the same length.

Observation 10. *Let $e \in E(G)$ be an edge and $a_1, a_2 \in A$ two agents such that $\pi_{\mathrm{SOL}}(a_1) = \{(e, [x_1, y_1])\}$ and $\pi_{\mathrm{SOL}}(a_2) = \{(e, [x_2, y_2])\}$, then $|y_1 - x_1| = |y_2 - x_2|$.*

For each edge $e \in E(G)$, the LP instance will have variables x_e^0, δ_e, and x_e^1. The variable δ_e represents the length of each piece $(e, [c, d])$ assigned to any agent that gets a piece fully inside e. The variable x_e^i represents the length of the piece of the edge e that was assigned to the agent $a \in A_V$ which is the guess for pair (e, i). We start by adding constraints ensuring that all pieces have non-negative length and that the sum of lengths of pieces on each edge is exactly 1.
For each agent $a \in A_V$, let Pieces_a denote the set of pieces that a is the guess for: $\mathrm{Pieces}_a = \{(e, i) \mid e \in E(G), i \in \{0, 1\}$, and a is the guess for $(e, i)\}$. Given the intended meaning of variables x_e^0, δ_e, and x_e^1, we can now add constraints to guarantee envy-freeness between agents in $a \in A_V$. For all $a, a' \in A_V$ we create the constraint

$$\sum_{(e,i)\in \mathrm{Pieces}_a} u_a(e) \cdot x_e^i \geq \sum_{(e',i')\in \mathrm{Pieces}_{a'}} u_a(e') \cdot x_{e'}^{i'}, \quad (1)$$

and for every $a \in A_V$ and $e \in E(G)$ we create the constraint

$$\sum_{(f,i)\in \mathrm{Pieces}_a} u_a(f) \cdot x_f^i \geq u_a(e) \cdot \delta_e. \quad (2)$$

Next, for every (ordered) pair of edges e, f such that $n_e > 0$ and $n_f > 0$, need to guarantee that agents that get a piece of length δ_e do not envy agents with pieces fully inside edge f. If we knew that $a \in A \setminus A_V$ gets a piece fully inside e, then for this agent we could express this via the constraint $u_a(e) \cdot \delta_e \geq u_a(f) \cdot \delta_f$. Unfortunately, we do not know which agents get a piece fully inside e and cannot obtain this information by exhaustive branching in view of our time bounds.

To overcome this obstacle, let us order the agents in $A \setminus A_V$ by the ratio $\frac{u_a(e)}{u_a(f)}$ and consider two agents a_1, a_2 such that $\frac{u_{a_1}(e)}{u_{a_1}(f)} \geq \frac{u_{a_2}(e)}{u_{a_2}(f)}$. It is easy to see that $u_{a_2}(e) \cdot \delta_e \geq u_{a_2}(f) \cdot \delta_f$ implies $u_{a_1}(e) \cdot \delta_e \geq u_{a_1}(f) \cdot \delta_f$. Hence to capture the desired constraint it will be sufficient to guess, for each ordered pair of edges (e, f) the agent $a_{(e,f)}$ that is assigned a piece fully inside e (by π_{SOL}), and has the smallest value for the fraction $\frac{u_{a_{(e,f)}}(e)}{u_{a_{(e,f)}}(f)}$ among all the agents that are assigned a piece fully inside e. Intuitively, this corresponds to guessing an envy-critical agent a: among all the agents fully assigned to e, the agent a is "closest" to envying agents fully assigned to f. Note that the guessed envy-critical agents will later preclude some agents from receiving a piece of e (in particular, those that precede a in the linear order defined by the fractions).

The procedure described above introduces at most k^2 many guesses of agents, which amounts to an additional branching factor of at most $|A \setminus A_V|^{k^2}$. For each agent $a_{(e,f)}$, we then add the following constraint to the LP instance

$$u_{a_{(e,f)}}(e) \cdot \delta_e \geq u_{a_{(e,f)}}(f) \cdot \delta_f. \quad (3)$$

We also perform additional consistency checks for this branching. First of all, we discard branches which select the same agent as being envy-critical in multiple pieces (i.e., if $e \neq e'$ then we require $a_{(e,f)} \neq a_{(e',f')}$). Moreover, since we have guessed at most $k-1$ agents for an edge e, we also check that the intended meaning of the choice of the agent $a_{(e,f)}$ is satisfied so far: for all triples of edges e, f, f' we check that

$$\frac{u_{a_{(e,f')}}(e)}{u_{a_{(e,f')}}(f)} \geq \frac{u_{a_{(e,f)}}(e)}{u_{a_{(e,f)}}(f)}. \quad (A)$$

At this point we have added constraints which prevent—assuming our guesses were correct—an agent in A_V from envying any other agent, and agents outside of A_V from envying each other. Finally, for every edge e and every agent $a \in A_V$ we would like to guarantee that the agents that get a piece fully inside e do not envy the agent a. Similarly as before, for each specific agent a' that gets a piece fully inside e we could hypothetically ensure this via the constraint $u_{a'}(e) \cdot \delta_e \geq \sum_{(f,i)\in \mathrm{Pieces}_a} u_{a'}(f) \cdot x_f^i$. However, we again do not know the agents that are assigned a piece fully inside e. Unfortunately, while for two edges e and f it was not too difficult to define and identify envy-critical agents and write linear constraints only for those, when comparing the envy of agents fully assigned to e towards an agent $a \in A_V$ that receives multiple pieces of edges, the notion of "envy-criticality" we need depends on the size of the pieces a gets from each edge. In particular, there is no fixed total ordering of the agents that allows us to define envy-criticality. To give a concrete example of this issue, for two different instantiations of the x_f^i variables, say $x_f^i := c_f^i$ and $x_f^i := d_f^i$, and two agents a_1 and a_2 it may hold that

$$\frac{\sum_{(f,i)\in \mathrm{Pieces}_a} u_{a_1}(f) \cdot c_f^i}{u_{a_1}(e)} \geq \frac{\sum_{(f,i)\in \mathrm{Pieces}_a} u_{a_2}(f) \cdot c_f^i}{u_{a_2}(e)},$$

but

$$\frac{\sum_{(f,i)\in \mathrm{Pieces}_a} u_{a_1}(f) \cdot d_f^i}{u_{a_1}(e)} \leq \frac{\sum_{(f,i)\in \mathrm{Pieces}_a} u_{a_2}(f) \cdot d_f^i}{u_{a_2}(e)}.$$

On the other hand, the assignment π_{SOL} does define some specific instantiation of x_e^i's for which there is a (not necessarily strict) total ordering on the agents a' in A capturing how "close" they are to envying a, i.e., based on the value of $\frac{\sum_{(f,i)\in \mathrm{Pieces}_a} u_{a'}(f) \cdot x_f^i}{u_{a'}(e)}$. While we have no way of computing which total ordering arises from the hypothetical assignment

π_{SOL}, we will later (in Lemma 11) use Theorem 13.22 [Basu *et al.*, 2006] described in the Preliminaries to show that only $|A|^{\mathcal{O}(k)}$ many such orderings are possible, and moreover that we can enumerate all of these in time $|A|^{\mathcal{O}(k)}$. For now, let us complete the description of the LP with this in mind. Since the number of relevant orderings is bounded by $|A|^{\mathcal{O}(k)}$, we can apply branching to guess the ordering that arises from a hypothetical targeted assignment. At that point we can also guess, for each edge e and agent $a \in A_V$, the *envy-critical* (according to this ordering) agent $\alpha_{e,a}$ that is assigned to the edge e and envies the agent a the most—and later use this guess to preclude some agents from being fully assigned to e.

For each guess, we add constraints to the LP which will ensure that the guess will be consistent with whatever solution the LP produces. In particular, we add the constraint

$$u_{a'}(e) \cdot \delta_e \geq \sum_{(f,i) \in \mathrm{Pieces}_a} u_{a'}(f) \cdot x_f^i \tag{4}$$

for every agent a' that envies a at most as much as $\alpha_{e,a}$ according to the guessed ordering. More precisely, with each guess we will get some instantiation $x_e^i = y_e^i$ witnessing this guess, and we will insert a copy of Constraint 4 for every agent a' satisfying the following property:

$$\frac{\sum_{(f,i) \in \mathrm{Pieces}_a} u_{\alpha_{e,a}}(f) \cdot y_f^i}{u_{\alpha_{e,a}}(e)} \geq \frac{\sum_{(f,i) \in \mathrm{Pieces}_a} u_{a'}(f) \cdot y_f^i}{u_{a'}(e)}. \tag{B}$$

Our last task in this step is to provide a way to perform the branching over total orders described above. Recall that each instantiation of the variables x_e^i, for $i \in \{0,1\}$ and $e \in E(G)$, gives rise to a set of total orderings of all agents in A. In particular, each ordering is associated with precisely one pair $(a \in A_V, e \in E(G))$. Here, each ordering captures the relative envy towards a under the assumption that the agents would be assigned to e (see Inequality B). We call a set of such orderings a *portfolio*. Moreover, since π_{SOL} also corresponds to an instantiation of the variables x_e^i, it too gives rise to a set of total orderings, which we call a *portfolio of* π_{SOL}.

Lemma 11 ($\star$)**.** *It is possible to construct, in time $A^{\mathcal{O}(k)}$, a set $\mathcal{R}$ of at most $k^k A^{\mathcal{O}(k)}$-many portfolios which is guaranteed to contain the portfolio of π_{SOL}.*

To formalize the description provided earlier, we now branch over all at most $k^k|A|^{\mathcal{O}(k)}$ many portfolios obtained from Lemma 11, or equivalently, points in the described metric space. Given some point $y \in \mathbb{R}^{2k}$, we guess for each pair of edge $e \in E(G)$ and agent $a \in A_V$ an agent $a_{e,a}$ as described above and introduce the LP instance constraints described in (4). Finally, similarly as after introducing Constraints (3), we can again check that the intended meaning of guessed agents for each edge e hold by checking Inequalities (A) and (B) for every pair of agents guessed for each edge e. If at least one of the inequalities do not hold, then we reject the branch. This finishes the construction of the LP instance.

Bipartite Matching. Now, we can solve each LP instance in at most cubic time [Cohen *et al.*, 2019]. If the instance is unsatisfiable, then the algorithm rejects this branch and continues to the next one. Else, given an LP solution x, we can assign the values for the agents that we already guessed. Namely for each agent $a \in A_V$, we let

$$\pi(a) = \bigcup_{(e,0) \in \mathrm{Pieces}_a} \{(e, [0, x_e^0])\} \cup \bigcup_{(e,1) \in \mathrm{Pieces}_a} \{(e, [1 - x_e^1, 1])\}.$$

Every other agent a that we identified via a guess was fully assigned to some particular edge e. Moreover, we can split the interval $[x_e^0, 1 - x_e^1]$ into n_e pieces of length δ_e; note that if $n_e > 0$ then δ_e cannot be equal to 0. Let $I_a \subseteq [x_e^0, x_e^1]$ be any of the pieces that has not been assigned to another agent yet and let $\pi(a) = (e, I_a)$.

Finally, we are left with some unassigned agents and some unassigned pieces of the graph, each consisting of a single piece of an edge. Since, $|A_V| + \sum_{e \in E(G)} n_e = |A|$, the number of unsigned pieces equals the number of unassigned agents. Moreover, since at this point we have a concrete partition, for every pair of agent a and piece (e, I) we can in polynomial time check whether a would envy a piece in the partition or not (since this check can be performed without knowing the assignments of the other agents); in the former case we say that a is *compatible* with (e, I), and otherwise we say that they are *incompatible*. We can thus create an auxiliary bipartite graph $H = (X \uplus Y, F)$ such that each vertex in X is identified with an unassigned agent, each vertex in Y is identified with an unassigned piece, and there is an edge between an agent $a \in X$ and a piece $(e, I) \in Y$ if and only if they are compatible. We compute a maximum matching M in H in time at most $\mathcal{O}(|A|^3)$. If M is not a perfect matching, then we reject the branch of our algorithm and try another branch. Else for each unassigned agent a, we let $\pi(a) = M(a)$, where $M(a)$ denotes the piece $(e, I) \in Y$ matched with the agent $a \in X$ by the matching M. In this case the algorithm outputs Yes and optionally also the assignment π as a witness. If none of the branches lead to a positive outcome, the algorithm outputs No.

This concludes the description of the algorithm. It now remains to prove correctness and verify the running time. ($\star$)

5 Concluding Remarks

Our results provide a significantly improved understanding of the classical complexity of envy-free graph cutting. One direction that may be of interest for future work is to analyze the complexity of this problem through the more refined *parameterized complexity* paradigm. Indeed, in that setting the algorithmic results presented here can be viewed as XP algorithms. An immediate question in this context is whether these results can be strengthened to show fixed-parameter tractability. Most prominently, is there a fixed-parameter algorithm for EF-GC parameterized by the number of edges?

For the special case where the underlying graph is a path, the complexity of the problem is an even more intriguing question. As EF-GC on a path is a special case of EF CONTIGUOUS CAKE CUTTING, we know that it always admits a solution and hence the decision version of the problem cannot be W[1]-hard (*for any parameterization*). If the problem of computing an envy-free solution does not admit a fixed-parameter algorithm, would showing this require a variation of the W-hierarchy tailored specifically to TFNP problems?

Acknowledgements

Ganian and Hamm acknowledge support from the Austrian Science Fund (FWF, Projects P31336 and Y1329). Hamm also acknowledges support from FWF (Project W1255-N23). Ordyniak acknowledges support by the EPSRC (EP/V00252X/1).

References

[Abebe *et al.*, 2017] Rediet Abebe, Jon Kleinberg, and David C. Parkes. Fair division via social comparison. In *AAMAS*, page 281–289, 2017.

[Aumann and Dombb, 2015] Yonatan Aumann and Yair Dombb. The efficiency of fair division with connected pieces. *ACM TEAC*, 3(4):1–16, 2015.

[Aziz *et al.*, 2018] Haris Aziz, Sylvain Bouveret, Ioannis Caragiannis, Ira Giagkousi, and Jérôme Lang. Knowledge, fairness, and social constraints. In *AAAI*, number 1, 2018.

[Balkanski *et al.*, 2014] E. Balkanski, S. Brânzei, D. Kurokawa, and A. Procaccia. Simultaneous cake cutting. In *AAAI*, pages 566–572, 2014.

[Basu *et al.*, 2006] Saugata Basu, Richard Pollack, and Marie-Françoise Roy. *Algorithms in Real Algebraic Geometry*. Springer-Verlag, 2006.

[Bei and Suksompong, 2021] Xiaohui Bei and Warut Suksompong. Dividing a graphical cake. In *AAAI*, pages 5159–5166, 2021.

[Bei *et al.*, 2017] Xiaohui Bei, Youming Qiao, and Shengyu Zhang. Networked fairness in cake cutting. In *IJCAI*, pages 3632–3638, 2017.

[Bei *et al.*, 2019] Xiaohui Bei, Ayumi Igarashi, Xinhang Lu, and Warut Suksompong. Connected fair allocation of indivisible goods. *arXiv preprint arXiv:1908.05433*, 2019.

[Bei *et al.*, 2020] Xiaohui Bei, Xiaoming Sun, Hao Wu, Jialin Zhang, Zhijie Zhang, and Wei Zi. Cake cutting on graphs: A discrete and bounded proportional protocol. In *SODA*, pages 2114–2123, 2020.

[Bei *et al.*, 2021] Xiaohui Bei, Ayumi Igarashi, Xinhang Lu, and Warut Suksompong. The price of connectivity in fair division. In *AAAI*, pages 5151–5158, 2021.

[Bilò *et al.*, 2021] Vittorio Bilò, Ioannis Caragiannis, Michele Flammini, Ayumi Igarashi, Gianpiero Monaco, Dominik Peters, Cosimo Vinci, and William S Zwicker. Almost envy-free allocations with connected bundles. *Games and Economic Behavior*, 2021.

[Bouveret *et al.*, 2017] Sylvain Bouveret, Katarína Cechlárová, Edith Elkind, Ayumi Igarashi, and Dominik Peters. Fair division of a graph. In *IJCAI*, pages 135–141, 2017.

[Brams and Taylor, 1996] Steven J Brams and Alan D Taylor. *Fair Division: From cake-cutting to dispute resolution*. Cambridge University Press, 1996.

[Brânzei and Miltersen, 2015] Simina Brânzei and Peter Bro Miltersen. A dictatorship theorem for cake cutting. In *IJCAI*, pages 482–488, 2015.

[Caragiannis *et al.*, 2011] Ioannis Caragiannis, John K. Lai, and Ariel D. Procaccia. Towards more expressive cake cutting. In *IJCAI*, pages 127–132, 2011.

[Chevaleyre *et al.*, 2017] Yann Chevaleyre, Ulle Endriss, and Nicolas Maudet. Distributed fair allocation of indivisible goods. *Artificial Intelligence*, 242:1–22, 2017.

[Cohen *et al.*, 2019] Michael B. Cohen, Yin Tat Lee, and Zhao Song. Solving linear programs in the current matrix multiplication time. In *STOC*, pages 938–942, 2019.

[Deligkas *et al.*, 2021] Argyrios Deligkas, Eduard Eiben, Robert Ganian, Thekla Hamm, and Sebastian Ordyniak. The parameterized complexity of connected fair division. In *IJCAI*, pages 139–145, 2021.

[Dubins and Spanier, 1961] Lester E Dubins and Edwin H Spanier. How to cut a cake fairly. *The American Mathematical Monthly*, 68(1P1):1–17, 1961.

[Eiben *et al.*, 2020] Eduard Eiben, Robert Ganian, Thekla Hamm, and Sebastian Ordyniak. Parameterized complexity of envy-free resource allocation in social networks. In *AAAI*, pages 7135–7142, 2020.

[Elkind *et al.*, 2021a] Edith Elkind, Erel Segal-Halevi, and Warut Suksompong. Graphical cake cutting via maximin share. In *IJCAI*, pages 161–167, 2021.

[Elkind *et al.*, 2021b] Edith Elkind, Erel Segal-Halevi, and Warut Suksompong. Keep your distance: Land division with separation. In *IJCAI*, pages 168–174, 2021.

[Elkind *et al.*, 2021c] Edith Elkind, Erel Segal-Halevi, and Warut Suksompong. Mind the gap: Cake cutting with separation. In *AAAI*, pages 5330–5338, 2021.

[Igarashi and Peters, 2019] Ayumi Igarashi and Dominik Peters. Pareto-optimal allocation of indivisible goods with connectivity constraints. In *AAAI*, pages 2045–2052, 2019.

[Marenco and Tetzlaff, 2014] Javier Marenco and Tomás Tetzlaff. Envy-free division of discrete cakes. *Discrete Applied Mathematics*, 164:527–531, 2014.

[Menon and Larson, 2017] Vijay Menon and Kate Larson. Deterministic, strategyproof, and fair cake cutting. In *IJCAI*, pages 352–358, 2017.

[Moulin, 2004] Hervé Moulin. *Fair division and collective welfare*. MIT press, 2004.

[Procaccia, 2013] Ariel D Procaccia. Cake cutting: Not just child's play. *Commun. of the ACM*, 56(7):78–87, 2013.

[Robertson and Webb, 1998] Jack Robertson and William Webb. *Cake-cutting algorithms: Be fair if you can*. CRC Press, 1998.

[Segal-Halevi *et al.*, 2016] Erel Segal-Halevi, Avinatan Hassidim, and Yonatan Aumann. Waste makes haste: bounded time algorithms for envy-free cake cutting with free disposal. *Transactions on Algorithms*, 13(1):1–32, 2016.

[Simonov *et al.*, 2019] Kirill Simonov, Fedor V. Fomin, Petr A. Golovach, and Fahad Panolan. Refined complexity of PCA with outliers. In *ICML*, pages 5818–5826, 2019.

[Stromquist, 1980] Walter Stromquist. How to cut a cake fairly. *Am. Math. Mon.*, 87(8):640–644, 1980.