Parameterized Complexity of Envy-Free Resource Allocation in Social Networks

Eduard Eiben,[1] Robert Ganian,[2] Thekla Hamm,[2] Sebastian Ordyniak[3]
[1]Department of Computer Science, Royal Holloway, University of London, UK. Email: eduard.eiben@rhul.ac.uk
[2]Algorithms and Complexity Group, TU Wien, Austria. Emails: rganian@gmail.com, thamm@ac.tuwien.ac.at
[3]Department of Computer Science, The University of Sheffield, UK. Email: sordyniak@gmail.com

Abstract

We consider the classical problem of allocating resources among agents in an envy-free (and, where applicable, proportional) way. Recently, the basic model was enriched by introducing the concept of a social network which allows to capture situations where agents might not have full information about the allocation of all resources. We initiate the study of the parameterized complexity of these resource allocation problems by considering natural parameters which capture structural properties of the network and similarities between agents and items. In particular, we show that even very general fragments of the considered problems become tractable as long as the social network has bounded treewidth or bounded clique-width. We complement our results with matching lower bounds which show that our algorithms cannot be substantially improved.

Introduction

Envy-freeness ranks among the most important fairness requirements in the classical resource allocation problem of distributing indivisible items (resources) among agents (Bouveret and Lang 2008; Bouveret, Chevaleyre, and Maudet 2016). There has also been an extensive line of works studying envy-freeness in a more general setting where agents only directly compare themselves to a subset of other agents (Beynier et al. 2019; Aziz et al. 2018; Bredereck, Kaczmarczyk, and Niedermeier 2018). For instance, employees in a company would only compare themselves and "envy" other employees that are at a comparable level to them in the company's hierarchy. Another example is tied to the classical CAKE CUTTING problem (Abebe, Kleinberg, and Parkes 2017; Bei, Qiao, and Zhang 2017), where agents have preferences over different parts of a "cake" (representing some desired goods)—in many cases, agents may only have limited information about the pieces of the cake distributed to other agents.

In line with the above, we consider the following problem: given a set R of *resources* (or *items*), a set A of *agents* (each with their own numerical valuation for each resource), and a directed *social network* G representing "envy-relations" between the agents, find an allocation of resources that is considered "envy-free" by each agent. More specifically we study two well-established notions of envy and resulting variants of the problem:

(1) in LOCALLY ENVY-FREE ALLOCATION (LEFA) each agent a is satisfied if it does not envy any of its neighbors—this models situations where agents do not know or care about the total available amount of resources. This was studied, e.g., by Beynier et al. (2019) and Bredereck et al. (2018).

(2) in ENVY-FREE ALLOCATION (EFA) each agent must not only be envy-free of its neighbors, but must also view its allocated resources as being proportional to the total amount of available resources. This variant is better suited to scenarios where an agent might not have access to full information about which agent receives which resources, but knows what all the resources are and expects to receive a fair share. EFA was proposed and studied by Aziz et al. (2018).

Contribution. Unsurprisingly, both LEFA and EFA are NP-complete, and in fact remain NP-complete even on severely restricted instances (Bredereck, Kaczmarczyk, and Niedermeier 2018). For example, the well-known PARTITION PROBLEM can be encoded by LEFA and EFA on a complete bidirected social network with 2 agents using the same valuation. LEFA and EFA generalize the classical envy-free resource allocation problems in the sense that the envy-free resource allocation problem corresponds to the setting in which the social network is complete. Similarly, EFA on the social network with an empty edge set encodes the problem of proportional resource allocation.

In this work, we employ the *parameterized complexity* paradigm (Downey and Fellows 2013; Cygan et al. 2015; Niedermeier 2006) to obtain new algorithms and lower bounds for both of these problems. The core feature of the parameterized paradigm is that instead of measuring the performance of algorithms merely in terms of the size of the input (n), one links this to certain properties of the input (captured by one or several numerical *parameters*, k). In turn, this gives rise to two notions of tractability, both of which correspond to polynomial-time tractability in the classical setting:

- the class FPT contains all problems that can be solved in time $f(k)\cdot n^{\mathcal{O}(1)}$ (for some computable function f), while
- the (asymptotically less efficient) class XP contains all problems that can be solved in time $n^{f(k)}$.

The fundamental question one must ask at this point is what a reasonable parameterization for LEFA and EFA would be. First of all, since both problems remain NP-complete even on very simple social networks (e.g., edgeless graphs for EFA; see Theorem 9) and both problems are well-known to be NP-complete on complete social networks, restricting the structure of G is not sufficient to obtain tractability on its own. Second, we believe that in practical settings, one often needs to deal with instances consisting of many agents and resources, and so we would like to avoid parameterizing by $|A|$ or $|R|$.

A well-established and intuitive assumption about instances of LEFA and EFA is that many resources or agents behave "homogeneously" in terms of their valuations—indeed, for resources homogeneity arises naturally when dealing with copies of the same item, while for agents homogeneity may be caused by inherent limitations of how preferences are collected. This homogeneity can be mathematically captured through the notion of *item-* and *agent-types*, which contain items and agents that are indistinguishable from each other based on any valuation. Indeed, the numbers of item- and agent-types (hereby denoted $|T_R|$ and $|T_A|$, respectively) have been studied and used as parameters in various settings across the whole field of computational social choice (Brânzei, Lv, and Mehta 2016; Ganian, Ordyniak, and Rahul 2019).

Results. Our first result focuses on tree-like social networks, specifically social networks of bounded *treewidth* tw (Robertson and Seymour 1983; Downey and Fellows 2013). Specifically, we show that LEFA and EFA are in XP parameterized by $\text{tw}(G) + |T_R|$ (Theorem 1). As with virtually all such results using treewidth, the core of the algorithm is a dynamic procedure—however, the edge orientations in combination with the distinct valuations of agents required the use of unusually involved records.

Next, we turn to the question whether the above result can be improved to FPT. We provide a reduction (Theorem 7) which not only answers this negatively, but also shows that the problem does not become FPT even with additional parameters (such as $|T_A|$). Two additional hardness results (Theorems 8 and 9) show that it is not possible to strengthen Theorem 1 by dropping any of the two parameters either.

For our third result, we look at another way of extending Theorem 1 towards a richer graph class. In particular, while treewidth is an extremely well-motivated graph parameter (see, for instance, the survey by Marx (2010) and the result of Thorup linking treewidth and control flow graphs (1998)), restricting the treewidth of instances means that our results do not immediately generalize the original non-graph setting and cannot be used for dense networks. To this end, we turn to *clique-width* (cw)—a well-established generalization of treewidth towards dense graphs—and show that LEFA and EFA are both in XP parameterized by $\text{cw}(G) + |T_A| + |T_R|$ (Theorem 10). It is worth noting that this result immediately implies that envy-free resource allocation (in the non-graph setting) is in XP parameterized by $|T_A| + |T_R|$—a result which, while forming a special case for us, is not trivial to prove on its own.

For our fourth result, we tackle the question of whether one can strengthen the polynomial-time result of Theorem 7 to fixed-parameter tractability for the problem by using a stronger restriction on the structure of G. We answer this positively, but with a caveat: we also need to restrict the *bundle-size* (i.e., the maximum number of items assigned to any agent). Specifically, we combine integer linear programming with exhaustive branching to show that both LEFA and EFA are FPT when parameterized by the size of a minimum vertex cover of G, $|T_R|$, and the bundle-size (Theorem 16).

Preliminaries

For an integer i, we let $[i] = \{1, 2, \ldots, i\}$ and $[i]_0 = [i] \cup \{0\}$. We denote by $\mathbb{N}$ the set of natural numbers, by $\mathbb{N}_0$ the set $\mathbb{N} \cup \{0\}$. For a set S, we denote by 2^S the set of all subsets of S. We refer to the handbook by Diestel (2012) for standard graph terminology. For a directed graph G and a vertex v, we denote by $N^+_G(v)$ the open out-neighborhood of v in G.

Treewidth. A *tree-decomposition* $\mathcal{T}$ of a (directed or undirected) graph G is a pair (T, χ), where T is a tree and χ is a function that assigns each tree node t a set $\chi(t) \subseteq V(G)$ of vertices such that the following conditions hold:

(P1) For every (directed) edge $uv \in E(G)$ there is a tree node t such that $u, v \in \chi(t)$.

(P2) For every vertex $v \in V(G)$, the set of tree nodes t with $v \in \chi(t)$ induces a non-empty subtree of T.

The sets $\chi(t)$ are called *bags* of the decomposition $\mathcal{T}$ and $\chi(t)$ is the bag associated with the tree node t. The *width* of a tree-decomposition (T, χ) is the size of a largest bag minus 1. The *treewidth* of a graph G, denoted by $\text{tw}(G)$, is the minimum width over all tree-decompositions of G.

For presenting our dynamic programming algorithms, it is convenient to consider tree-decompositions in the following normal form (Kloks 1994): A tree-decomposition (T, χ) is a *nice tree-decomposition* of a graph G if the tree T is rooted at node r, and each node of T is of one of the following four types: (1) a *leaf node* t without children and $|\chi(t)| = 1$, 2) an *introduce node* t with exactly one child t' and $\chi(t) = \chi(t') \cup \{v\}$ for a node v of G, 3) a *forget node* t with exactly one child t', and $\chi(t) = \chi(t') \setminus \{v\}$ for a node v of G, and 4) a *join node* t with exactly two children t_1, t_2, and $\chi(t) = \chi(t_1) = \chi(t_2)$.

For $t \in V(T)$ we denote by T_t the subtree of T rooted at t and we write $\chi(T_t)$ for the set $\bigcup_{t' \in V(T_t)} \chi(t')$.

Computing a nice tree-decomposition of a graph with $\mathcal{O}(\text{tw}(G) \cdot |V(G)|)$ many nodes and optimal width is fixed-parameter tractable, and there are also even more efficient approximation algorithms available (Kloks 1994; Bodlaender 1996; Bodlaender et al. 2016).

Clique-width. To define *clique-width*, a prominent graph parameter which will be relevant for our results, we first need to introduce some basic terminology. For a positive integer k, we let a *k-graph* be a graph whose vertices are labeled by $[k]$. For convenience, we consider an arbitrary directed graph to be a k-graph with all vertices labeled by 1.

We call the k-graph consisting of exactly one vertex v (say, labeled by i) an initial k-graph and denote it by $i(v)$.

The (*directed*) *clique-width* of a graph G is the smallest integer k such that G can be constructed from initial k-graphs by means of repeated application of the following three operations:

1. Disjoint union (denoted by $\oplus$);
2. Relabeling: changing all labels i to j (denoted by $p_{i\to j}$);
3. Edge insertion: adding an edge from each vertex labeled by i to each vertex labeled by j ($i \neq j$; denoted by $\eta_{i,j}$).

A *k-expression tree* (Courcelle, Makowsky, and Rotics 2000) is a rooted tree representation of how the three operations are used to construct a given graph; specifically, the k-expression tree represents each $i(v)$ as a leaf, each $\oplus$ operator as an $\oplus$ node with two children, and each $p_{i\to j}$ or $\eta_{j,i}$ operator by a corresponding node with a single child.

Problem Statement. Let A be a set of agents, R be a set of items (or resources), and G be a directed graph with vertex set A. A *preference function* (or *valuation function*) for an agent $a \in A$ is a function $\tau_a : 2^R \to \mathbb{N}$. Throughout the paper we will assume that preference functions are additive, i.e., $\tau_a(R') = \sum_{r\in R'} \tau_a(r)$ for every $R' \subseteq R$.

An *allocation* is a mapping $\pi : A \to 2^R$ such that $\pi(a)$ and $\pi(a')$ are disjoint for every two distinct agents a and a' in A and $\bigcup_{a\in A} \pi(a) = R$. With a slight abuse of notation, we set $\pi(A') = \bigcup_{a\in A'} \pi(a)$ for a subset A' of A. We say that $\pi(a)$, or more generally any set of items, is a *bundle*. An allocation is:

- *proportional* if $\tau_a(\pi(a)) \geq \frac{\tau_a(\pi(A\setminus N_G^+(a)))}{|A\setminus N_G^+(a)|}$ for every $a \in A$.
- *locally envy-free* if $\tau_a(\pi(a)) \geq \tau_a(\pi(a'))$ for every $a, a' \in A$ with $a' \in N_G^+(a)$.
- *envy-free* if it is both proportional and locally envy-free.

We can now formalize our problems of interest:

ENVY-FREE ALLOCATION (EFA))	
Input:	A set A of agents, a set R of items, preference functions $\tau_a : 2^R \to \mathbb{N}$ for every agent $a \in A$, and a directed graph G with vertex set A.
Question:	Is there an envy-free allocation?

LOCALLY ENVY-FREE ALLOCATION (LEFA) is defined analogously, with the sole distinction that we ask for a locally envy-free allocation. We note that while both problems are stated as decision problems, all our algorithms are constructive and can also output an allocation with the desired properties as a witness.

Parameterizations and Properties of Instances. We say that two agents a and a' have the same *agent-type* if their preference functions τ_a and $\tau_{a'}$ are identical. We say two items r and r' have the same *item-type* if they are equally valued by any agent, i.e., if $\tau_a(r) = \tau_a(r')$ for every $a \in A$.

Let $\mathcal{I} = (A, R, (\tau_a)_{a\in A})$ be an instance of (LOCALLY) ENVY-FREE ALLOCATION. We denote by T_A and T_R the set of agent-types and item-types of $\mathcal{I}$, respectively, and define the preference function τ_a in the natural manner for agent-types and item-types, i.e., for an agent-type $t_a \in T_A$ and an item-type $t_r \in T_R$, we denote by $\tau_{t_a}(t_r)$, the valuation of any agent of type t_a of any item of type t_r.

We also call sets of items $R' \subseteq R$ *bundles*. In the context of item-types, we will alternatively (and interchangeably) denote bundles as $|T_R|$-dimensional vectors $\vec{b}$, where $\vec{b}[t_r]$ is equal to the number of items of item-type t_r in a bundle, for every $t_r \in T_R$. We use $\vec{b}(R')$ to denote the vector representing the bundle R' and conversely we denote by $\text{BUN}(\vec{b})$ a bundle representing the vector $\vec{b}$. Moreover, we denote by $\mathcal{B} = \{\vec{b}(R') \mid R' \subseteq R\}$, the set of all possible bundle vectors.

Our results will mainly be concerned with establishing the tractability of EFA and LEFA under the combination of (1) a graph parameter that restricts the structure of the network and (2) the number of agent-types or item-types which, in turn, restrict the complexity of the preference function. Both types of restrictions are necessary in order to achieve tractability. For (1), we will consider the treewidth, clique-width, and *vertex cover number* (vcn(G)—the size of a minimum vertex cover) of the network G). Our last result uses the maximum size of a bundle as an additional parameter.

Allocating Resources on Tree-Like Networks

As our first result, we show that instances with a bounded number of item-types can be solved in polynomial time on tree-like networks. We note that, as will be shown in the next section, both conditions are necessary for tractability.

Theorem 1. (LOCALLY) ENVY-FREE ALLOCATION *is in* XP *parameterized by treewidth of the social network and number of item-types.*

For the remainder of this section, let $\mathcal{I} = (A, R, (\tau_a)_{a\in A}, G)$ be an instance of (LOCALLY) ENVY-FREE ALLOCATION. Since a nice tree-decomposition of a graph can be computed efficiently (Kloks 1994; Bodlaender 1996; Bodlaender et al. 2016), it suffices to solve the problem when a minimum-width nice tree-decomposition of G is provided as part of the input.

Theorem 2. (LOCALLY) ENVY-FREE ALLOCATION *can be solved in time* $|R|^{\mathcal{O}(|T_R|\cdot \text{tw}(G))}|A|$*, provided with a minimum-width nice tree-decomposition of G in the input.*

Informally, the algorithm behind the above theorem works as follows. Let $\mathcal{T} = (T, \chi)$ be a minimum-width nice tree-decomposition of G. The algorithm uses a bottom-up dynamic programming approach on the nodes of T to compute a compact representation, in the following represented by a set of valid records, of all envy-free assignments of $\mathcal{I}$ restricted to the agents in $\chi(T_t)$ for every node $t \in V(T)$.

A *record* for a node $t \in V(T)$ is a triple $(\alpha, \vec{u}, \beta)$, where:

- $\alpha : \chi(t) \to \mathcal{B}$ is a function that provides an allocation for every agent $a \in \chi(t)$,
- $\vec{u} \in \mathcal{B}$ is the bundle containing all items already assigned to the agents in $\chi(T_t)$,

- $\beta : \chi(t) \rightarrow \mathcal{B}$ is a function that provides the bundle containing all items assigned to all out-neighbors of a in $\chi(T_t)$. β is only required to ensure proportionality (i.e., it can be omitted from the records when solving LEFA).

The semantics of a record are as follows. We say that a record $(\alpha, \vec{u}, \beta)$ for a node $t \in V(T)$ is *valid* if there is an allocation $\pi : \chi(T_t) \rightarrow 2^R$ satisfying:

(R1) $\alpha(a) = \vec{b}(\pi(a))$ for every $a \in \chi(t)$,

(R2) $\vec{u} = \vec{b}(\pi(\chi(T_t)))$,

(R3) $\beta(a) = \vec{b}(\pi(N_G^+(a) \cap \chi(T_t)))$ for every $a \in \chi(t)$,

(R4) π is locally envy-free on the instance induced by the agents in $[\chi(T_t)]$, i.e., for all $a \in \chi(T_t)$, it holds that $\tau_a(\pi(a)) \geq \tau_a(\pi(a'))$ for every $a' \in N_G^+(a) \cap \chi(T_t)$

(R5) (Only for EFA) π is proportional for all $a \in \chi(T_t) \setminus \chi(t)$, i.e., for every $a \in \chi(T_t) \setminus \chi(t)$, it holds that $\tau_a(\pi(a)) \geq \frac{\tau_a(\pi(A \setminus N_G^+(a)))}{|A \setminus N_G^+(a)|}$ for every $a \in A$.

For a node $t \in V(T)$ we denote by $\mathcal{R}(t)$ the set of all valid records for t. Note that $\mathcal{I}$ has a envy-free allocation if and only if $\mathcal{R}(r)$, for the root r of T, contains a record $(\alpha, \vec{u}, \beta)$ such that $\vec{u} = \vec{b}(R)$. Moreover, once we have computed the set of records for all nodes, a straightforward application of standard techniques (Downey and Fellows 2013) can be used to obtain an envy-free allocation using a second top-to-bottom run through the tree-decomposition.

We will show next that $\mathcal{R}(t)$ can be computed via a dynamic programming algorithm on $\mathcal{T}$ in a bottom-up manner. The algorithm starts by computing the set of all valid records for the leaves of T and then proceeds by computing the set of all valid records for the other three types of nodes of a nice tree-decomposition (always selecting nodes all of whose children have been processed). The following four lemmas show how this is achieved.

Lemma 3. *Let $l \in V(T)$ be a leaf node. Then $\mathcal{R}(l)$ can be computed in time $\mathcal{O}(|\mathcal{B}|)$, and $|\mathcal{R}(l)| \in \mathcal{O}(|\mathcal{B}|)$.*

Sketch of Proof. Let $\chi(t) = \{a\}$. $\mathcal{R}(l)$ contains all records $(\alpha, \vec{u}, \beta)$ such that $\alpha(a) \in \mathcal{B}$, $\vec{u} = \alpha(a)$, and $\beta(a) = \vec{b}(\emptyset)$. □

Lemma 4. *Let $t \in V(T)$ be an introduce node with child t'. Then $\mathcal{R}(t)$ can be computed from $\mathcal{R}(t')$ in time $\mathcal{O}(|\mathcal{R}(t')||\mathcal{B}||T_R|\text{tw}(G))$.*

Sketch of Proof. Let a be the unique agent in $\chi(t) \setminus \chi(t')$. Informally, the records in $\mathcal{R}(t)$ are obtained by extending a record in $\mathcal{R}(t')$ with an allocation $\alpha_a \in \mathcal{B}$ for a: For every record $(\alpha', \vec{u}', \beta') \in \mathcal{R}(t')$ and every allocation $\alpha_a \in \mathcal{B}$ for a for which there are sufficient items available and no agent in $\chi(t')$ envies a and vice versa, $\mathcal{R}(t)$ contains a record $(\alpha, \vec{u}, \beta)$, where α is the extension of α' by α_a, $\vec{u} = \vec{u}' + \alpha_a$, and β is updated for a as well as all in-neighbors of a in $\chi(t)$. □

Lemma 5. *Let $t \in V(T)$ be a forget node with child t'. Then $\mathcal{R}(t)$ can be computed from $\mathcal{R}(t')$ in time $\mathcal{O}(|\mathcal{R}(t')||R|)$.*

Sketch of Proof. Let a be the unique agent in $\chi(t') \setminus \chi(t)$. Informally, every record in $\mathcal{R}(t)$ is obtained from the restriction of a record in $\mathcal{R}(t')$ to $\chi(t)$. In the case of EFA one additionally needs to check that a satisfies proportionality. □

Lemma 6. *Let $t \in V(T)$ be a join node with children t_1 and t_2. Then $\mathcal{R}(t)$ can be computed from $\mathcal{R}(t_1)$ and $\mathcal{R}(t_2)$ in time $\mathcal{O}(|\mathcal{R}(t_1)||\mathcal{R}(t_2)||T_R|\text{tw}(G))$.*

Sketch of Proof. Informally, every record in $\mathcal{R}(t)$ is obtained as the combination of two records $(\alpha_1, \vec{u}_1, \beta_1) \in \mathcal{R}(t_1)$ and $(\alpha_2, \vec{u}_2, \beta_2) \in \mathcal{R}(t_2)$ such that $\alpha_1 = \alpha_2$ and $\vec{u}_1 + \vec{u}_2 - \sum_{a \in \chi(t)} \alpha_1(a) \leq \vec{b}(R)$. □

We are now ready to establish our main theorem.

Proof of Theorem 2. The algorithm computes the set of all valid records $\mathcal{R}(t)$ for every node t of T using a bottom-up dynamic programming algorithm starting in the leaves of T. It then solves $\mathcal{I}$ by checking whether $\mathcal{R}(r)$ contains a record $(\alpha, \vec{u}, \beta)$ such that $\vec{u} = \vec{b}(R)$. Note that the correctness of the algorithm follows from the correctness of Lemmas 3, 4, 5, and 6. The running-time of the algorithm is at most the number of nodes of T, which can be assumed to be upper-bounded by $\text{tw}(G) \cdot |A|$ (Cygan et al. 2015, Lemma 7.4), times the maximum time required to compute $\mathcal{R}(t)$ for any of the four node types of a nice tree-decomposition, which because of lemmas 3, 4, 5, and 6 is at most $\mathcal{O}(|\mathcal{R}(t)|^2|T_R|\text{tw}(G))$. Because $|\mathcal{R}(t)| \leq |R|^{|T_R|(2\text{tw}(G)+1)}$, we obtain $\mathcal{O}(|R|^{2|T_R|(2\text{tw}(G)+1)}(\text{tw}(G))^2|T_R||A|)$ as the total running-time of the algorithm. □

Algorithmic Lower Bounds

The aim of this section is to show that Theorem 1 is essentially tight—i.e., one can strengthen it neither by dropping one of the two parameters, nor by obtaining a fixed-parameter algorithm for the same parameterization. To obtain the latter result, we give a parameterized reduction which establishes that both LEFA and EFA are "W[1]-*hard*" (Downey and Fellows 2013; Cygan et al. 2015) under this parameterization.

The problem we reduce from is called EQUITABLE COLORING, and is the same as the classical COLORING problem but with the added requirement that each of the q colors occurs precisely the same number of times (w.l.o.g. one assumes that the number of vertices in the input graph G is divisible by q). EQUITABLE COLORING is known to be W[1]-hard parameterized by $\text{tw}(G) + q$ (Fellows et al. 2011).

Theorem 7. (LOCALLY) ENVY-FREE ALLOCATION *is* W[1]-*hard parameterized by treewidth, number of item-types, and number of agent-types, and bundle-size.*

Sketch of Proof. Given an instance (G, q) or EQUITABLE COLORING, we will construct an instance of $\mathcal{I} = (A, R, (\tau_a)_{a \in A}, G')$ of (LOCALLY) ENVY-FREE ALLOCATION as follows:

Agents. The set A is the union of the following sets:

- a set A_V containing an agent a_v for every $v \in V(G)$,

- For every edge $e = \{u, v\} \in E(G)$: Sets $A_e^u = \{ a_{e,u}^i \mid i \in [q] \}$, $A_e^v = \{ a_{e,v}^i \mid i \in [q] \}$, and $A_e = \{ a_e^{i,j} \mid i, j \in [q], i \neq j \}$.
- A set A_d of $10q^2(|V(G)| + |E(G)|)$ agents.
- A single agent a_d.

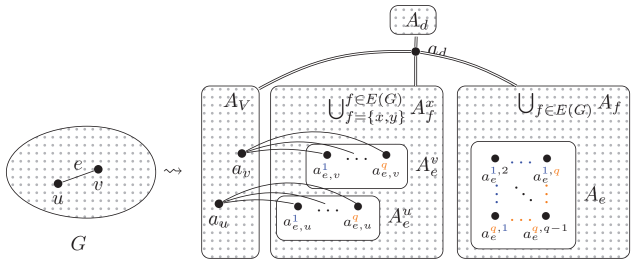

Figure 1: Social network in proof of Theorem 7; specifically for an edge $e = \{u, v\}$. Double lines denote compete connectivity and edges exist between vertices of the middle and right rectangle if they share a same-color (upper) index.

Graph. The graph G' is bidirectional, and whenever we say that G' contains edge $\{a_1, a_2\}$ we mean that it contains both arcs (a_1, a_2) and (a_2, a_1). The graph G' contains the following edges:

- For every $i \neq j \in [q]$, $v \in V(G)$, and $e = \{u, v\} \in E(G)$ edges: $\{a_v, a_{e,v}^i\}$, $\{a_{e,v}^i, a_e^{i,j}\}$, and $\{a_{e,v}^i, a_e^{j,i}\}$.
- An edge between a_d and every other agent.

An illustration of the graph is given in Figure 1.

Items. The set R is the union of the following sets.

- For each color $c \in [q]$ a set R_c of $\frac{|V(G)|}{q}$ items.
- A set $R_\square$ of $2 \cdot |E(G)|$ items.
- A set R_E of $2(q-1) \cdot |E(G)|$ items.
- A set D of $(q^2 - q - 1) \cdot |E(G)|$ items.
- A set U of $\cdot |E(G)|$ items.
- A set R_d of $10q^2(|V(G)| + |E(G)|)$ items.
- A single item r_d.

Preferences. If we do not explicitly specify the value of an item $r \in R$ for an agent $a \in A$, we assume $\tau_a(r) = 0$. We have the following types of agents:

1. For $a \in A_V$:
   - If $r \in \{r_d\} \cup \bigcup_{c \in [q]} R_c$, we let $\tau_a(r) = 1$.
2. For each $c \in [q]$, $v \in V(G)$, $e = \{u, v\} \in E(G)$ and $a = a_{e,v}^c$:
   - If $r \in R_\square$, we let $\tau_a(r) = 3$. If $r \in R_c$, we let $\tau_a(r) = 2$. If $r \in \{r_d\} \cup R_E$, we let $\tau_a(r) = 1$.
3. For $a \in A_e$ for some $e \in E(G)$:
   - If $r \in D$, we let $\tau_a((r)) = 3$. If $r \in R_E$, we let $\tau_a(r) = 2$. If $r \in \{r_d\} \cup U$, we let $\tau_a(r) = 1$.
4. for $a \in A_d$:
   - If $r \in \{r_d\} \cup R_d$, we let $\tau_a((r)) = 1$.
5. for $a = a_d$:
   - If $r = r_d$, we let $\tau_a((r)) = 1$.

This concludes the construction. It is not difficult to verify that parameters are bounded by function of $\text{tw}(G) + q$. We will now show that (G, q) is a YES-instance of EQUITABLE COLORING if and only if $\mathcal{I} = (A, R, (\tau_a)_{a \in A}, G')$ is a YES-instance of (LOCALLY) ENVY-FREE ALLOCATION.

For simplicity, in the rest of the proof we will use $\pi(a) \in S$ as a shorthand for $\pi(a) = \{r\}$ for an arbitrary item $r \in S$. Now, let $\chi : V(G) \to [q]$ be an equitable coloring of G. We define an envy-free allocation $\pi : A \to 2^R$ as follow:

- For $v \in V(G)$, we let $\pi(a_v) \in R_{\chi(v)}$.
- For $v \in V(G)$, $e = \{u, v\} \in E(G)$, $i \in [q]$ we let $\pi(a_{e,v}^i) \in R_E$ if $i \neq \chi(v)$ and $\pi(a_{e,v}^i) \in R_\square$ otherwise.
- For $e = \{u, v\} \in E(G)$, $i, j \in [q]$, $i \neq j$, we let $\pi(a_e^{i,j}) \in U$ if $i = \chi(u)$ and $j = \chi(v)$ and we let $\pi(a_e^{i,j}) \in D$ otherwise.
- For $a \in A_d$, we let $\pi(a) \in R_d$.
- We let $\pi(a_d) = \{r_d\}$.

For the backward direction, let π be a locally envy-free allocation. First, it is easy to observe that a_d has to be assigned the item r_d, which forces to assign agents in A_V precisely the items in $\bigcup_{c \in [q]} R_c$ and agents in A_d the items in R_d. This clearly induces coloring with same number of vertices per color. To show that this assignment gives a proper coloring, we show that items in $R_\square$ have to be assigned to agents $a_{e,v}^i$ such that $\pi(a_v) \in R_i$ and we have to assign precisely one item in U to agents A_e for each edge $e = \{u, v\}$; both neighbors (not counting a_d) of these agents have to be assigned an item in $R_\square$. $\square$

Note that in the proof of Theorem 7, the number of item-types and number of agent-types is bounded by a function of the number of colors q and is independent of treewidth of G. Moreover, the bundle-size is 1. Furthermore, EQUITABLE COLORING is NP-hard already for 3 colors. Therefore, starting from an instance of EQUITABLE COLORING with 3 colors and using the reduction given in the proof of Theorem 7, we get the following theorem:

Theorem 8. (LOCALLY) ENVY-FREE ALLOCATION *is* NP*-hard even when restricted to instances with a bounded number of item-types, number of agent-types, and bundle-size.*

Our last lower-bound result is a counterpart to Theorem 8 showing that Theorem 1 cannot be strengthened by omitting the number of item-types from the parameterization.

Theorem 9. (LOCALLY) ENVY-FREE ALLOCATION *is* NP*-hard even for treewidth* 0 *(resp.* 1 *for* LEFA*) and one agent-type.*

Sketch of Proof. The reduction is nearly identical to the one given in a previous work by Bliem, Bredereck and Niedermeier (2016, Theorem 3). Their result showed that a modification of EFA on complete graphs is W[1]-hard parameterized by the number of agents, even when there is only one agent-type. The only difference in the reduction is that the graph G is either empty (EFA) or a star whose center is the first agent (LEFA). $\square$

Dealing with Dense Networks

A limitation of Theorem 1 is that it requires the social network to be sparse. In this section, we will show that the graph parameter clique-width can be used instead of treewidth as long as the number of agent-types is also bounded. Since complete bidirected graphs have a clique-width of 2, this setting also generalizes the (well-studied) problem of allocating resources to agents without a network.

Theorem 10. (LOCALLY) ENVY-FREE ALLOCATION *is in* XP *parameterized by clique-width of the social network, number of item-types, and number of agent-types.*

For the remainder of this subsection, let $\mathcal{I} = (A, R, (\tau_a)_{a \in A}, G)$ be the given instance of (LOCALLY) ENVY-FREE ALLOCATION. It is known that an approximate k-expression tree can be computed in fixed-parameter time even for digraphs (Oum and Seymour 2006; Kanté and Rao 2013; Ganian, Hlinený, and Obdrzálek 2013), and so it suffices to solve the problem when a k-expression tree for G is provided in the input.

Theorem 11. (LOCALLY) ENVY-FREE ALLOCATION *can be solved in time $\mathcal{O}(|R|^{|T_R|(2k|T_A|+1)}k^2|A|)$ when a k-expression tree T of G is provided as part of the input.*

Let t be a node of T, and recall that t could be one of the following four types of nodes: $i(v)$, $\oplus$, $\eta_{i,j}$ or $p_{i \to j}$. Let T_t be the subtree of T rooted at t, and let G_t be the k-graph defined by the k-expression tree T_t; furthermore, let Ω_t denote the set of labels used in G_t, A_t denote the set of agents in G_t, and let A_t^ω denote the set of all agents in G_t with label ω. For instance, if r is the root of T then $G_r = G$, and for each leaf t in T it holds that G_t is a graph with a single labeled agent.

The high-level idea of the algorithm is similar to the idea behind our algorithm for treewidth, i.e., the aim is to compute a set of records for every node $t \in V(T)$ (in a leaf-to-root fashion), where each record represents a set of partial solutions for G_t. However, the records and computations required here are significantly more complex than those used for treewidth—and this is *especially* the case for EFA.

A *record* for a node $t \in V(T)$ is a tuple $(\alpha_{\min}, \alpha_{\max}, \vec{u}, \beta)$, where:

- $\alpha_{\min} : \Omega_t \times T_A \to \mathcal{B}$ is a function that for every $\omega \in \Omega_t$ and every $t_a \in T_A$ provides the bundle with minimum value w.r.t. τ_{t_a} allocated to any agent of type t_a and label ω in G_t.
- $\alpha_{\max} : \Omega_t \times T_A \to \mathcal{B}$ is a function that for every $\omega \in \Omega_t$ and every $t_a \in T_A$ provides the bundle with maximum value w.r.t. τ_{t_a} allocated to any agent with label ω in G_t.
- $\vec{u} : \Omega_t \to \mathcal{B}$ is a function that for every label $\omega \in \Omega_t$ provides the bundle containing all items already assigned to all agents with label ω in G_t. Note that the distinction between different labels is only necessary when considering proportionality—for LEFA, it suffices to merely remember the bundle of all items assigned so far.
- (can be omitted for LEFA) β is a function that maps every tuple $(\omega \in \Omega_t, t_a \in T_A)$ to $(a, \vec{b}, \vec{b}^+)$, where (informally) a is an agent that maximizes the distance to satisfying proportionality, $\vec{b}$ is the bundle assigned to a, and $\vec{b}^+$ is the bundle containing all items assigned to all out-neighbors of a in A_t.

The semantics of a record are as follows. We say that a record $(\alpha_{\min}, \alpha_{\max}, \vec{u}, \beta)$ for a node $t \in V(T)$ is *valid* if there is an allocation $\pi : A_t \to 2^R$ satisfying:

(R1) For every $\omega \in \Omega_t$ and $t_a \in T_A$, it holds that $\alpha_{\min}(\omega, t_a) = \vec{b}(\pi(a))$, where a is an agent of type t_a with label ω minimizing $\tau_a(\pi(a))$ among all agents in G_t of type t_a and label ω.

(R2) For every $\omega \in \Omega_t$, $t_a \in T_A$, it holds that $\alpha_{\max}(\omega, t_a) = \vec{b}(\pi(a))$, where a is an agent with label ω in G_t maximizing $\tau_a(\pi(a))$ among all agents with label ω in G_t.

(R3) For every $\omega \in \Omega_t$, it holds that $\vec{u}(\omega) = \vec{b}(\pi(A_t^\omega))$.

(R4) π is locally envy-free on the instance induced by the agents in A_t, i.e., for all $a \in A_t$, it holds that $\tau_a(\pi(a)) \geq \tau_a(\pi(a'))$ for every $a' \in N_G^+(a) \cap A_t$,

(R5) This condition only applies for EFA, and is also the most involved. For every $\omega \in \Omega_t$, $t_a \in T_A$, it holds that $\beta(\omega, t_a) = (a, \vec{b}_a, \vec{b}_a^+)$, where a is an agent of type t_a with label ω in G_t that maximizes:

$$\tau_{t_a}(R) - \tau_{t_a}((\pi(a)))|A \setminus N_G^+(a)| - \tau_{t_a}(\pi(N_{G_t}^+(a)))$$

Note that in R5, the value of the equation equals the value that is still required, i.e., still needs to be distributed among the out-neighbors of a in $G_t \setminus A_t$, for agent a to satisfy proportionality. Since a maximizes the required value (among all agents with label ω and type t_a), this implies that once we added sufficient value among the out-neighbors of a to satisfy proportionality for a, all agents with label ω and type t_a satisfy proportionality. Also note that it would be sufficient to only store the required value for a (instead of the triple $(a, \vec{b}_a, \vec{b}_a^+)$), however, then our algorithm would only be efficient for instances with a unary encoding of the valuations.

For a node $t \in V(T)$ we denote by $\mathcal{R}(t)$ the set of all valid records for t. Then $\mathcal{I}$ is a YES-instance if and only if $\mathcal{R}(r)$, for the root r of T, contains a record $(\alpha_{\min}, \alpha_{\max}, \vec{u}, \beta)$ such that $\sum_{\omega \in \Omega_r} \vec{u}(\omega) = \vec{b}(R)$ and, for the case of EFA, additionally $\tau_{t_a}(R) - \tau_{t_a}(\vec{b}_a)|A \setminus N_G^+(a)| \leq \tau_{t_a}(\vec{b}_a^+)$, where $\beta(\omega, t_a) = (a, \vec{b}_a, \vec{b}_a^+)$ for every $\omega \in \Omega_t$ and every $t_a \in T_A$. To conclude the proof, it now suffices to show how to compute our records $\mathcal{R}(t)$ in a leaf-to-root fashion.

Lemma 12. *Let $\ell \in V(T)$ be a leaf node of the form $\omega(a)$. Then $\mathcal{R}(\ell)$ can be computed in time $\mathcal{O}(|\mathcal{B}|)$.*

Sketch of Proof. Let t_a be the agent-type of a. Then, for every $\alpha_a \in \mathcal{B}$, $\mathcal{R}(t)$ contains the record $(\alpha_{\min}, \alpha_{\max}, \vec{u}, \beta)$ such that $\alpha_{\min}(\omega, t_a) = \alpha_a$, $\alpha_{\max}(\omega, t_a) = \alpha_a$, $\vec{u}(\omega) = \alpha_a$, and $\beta(\omega, t_a) = (a, \alpha_a, \vec{b}(\emptyset))$. ☐

Lemma 13. *Let $t \in V(T)$ be a disjoint union node with children t_1 and t_2. Then $\mathcal{R}(t)$ can be computed from $\mathcal{R}(t_1)$ and $\mathcal{R}(t_2)$ in time $\mathcal{O}(|\mathcal{R}(t_1)||\mathcal{R}(t_2)|k|T_A||T_R|)$.*

Sketch of Proof. Informally, for every two records $(\alpha^1_{\min}, \alpha^1_{\max}, \vec{u}^1, \beta^1) \in \mathcal{R}(t_1)$ and $(\alpha^2_{\min}, \alpha^2_{\max}, \vec{u}^2, \beta^2) \in \mathcal{R}(t_2)$, $\mathcal{R}(t)$ contains the record $(\alpha_{\min}, \alpha_{\max}, \vec{u}, \beta)$ such that (1) $\alpha_{\min}(\omega, t_a)$ is equal to $\alpha^1_{\min}(\omega, t_a)$ if $\alpha^1_{\min}(\omega, t_a) \leq \alpha^2_{\min}(\omega, t_a)$ or equal to $\alpha^2_{\min}(\omega, t_a)$, otherwise, (2) the case for $\alpha_{\max}(\omega, t_a)$ is similar to (1) only that we now take the maximum, (3) $\vec{u}(\omega) = \vec{u}^1(\omega) + \vec{u}^2(\omega)$, and (4) $\beta(\omega, t_a)$ is either equal to either $\beta^1(\omega, t_a)$ or $\beta^2(\omega, t_a)$ depending on which of the associated agents requires more to satisfy proportionality. ☐

Lemma 14. *Let $t \in V(T)$ be a relabeling node of the form $p_{i \to j}$ with child t'. Then $\mathcal{R}(t)$ can be computed from $\mathcal{R}(t')$ in time $\mathcal{O}(|\mathcal{R}(t')|k|T_A||T_R|)$.*

Sketch of Proof. Informally, relabeling label i to label j can be seen as taking the disjoint union of G^i_t and G^j_t and leaving everything else untouched. Consequently, the main complications for this case have already been addressed in Lemma 13. ☐

Lemma 15. *Let $t \in V(T)$ be an add-edge node with child t' of the form $\eta_{i,j}$. Then $\mathcal{R}(t)$ can be computed from $\mathcal{R}(t')$ in time $\mathcal{O}(|\mathcal{R}(t')|k|T_A||T_R|)$.*

Sketch of Proof. For every record $(\alpha'_{\min}, \alpha'_{\max}, \vec{u}', \beta') \in \mathcal{R}(t')$ that for every $t_a \in T_A$ satisfies $\tau_{t_a}(\alpha'_{\min}(i, t_a)) \geq \tau_{t_a}(\alpha'_{\max}(j, t_a))$ (ensuring that no agent of label i envies an agent of label j), $\mathcal{R}(t)$ contains the record $(\alpha_{\min}, \alpha_{\max}, \vec{u}, \beta)$ such that: $\alpha_{\min} = \alpha'_{\min}$, $\alpha_{\max} = \alpha'_{\max}$, $\vec{u} = \vec{u}'$, and for every $\omega \in \Omega_t$ and $t_a \in T_A$, either: $\beta(\omega, t_a) = \beta'(\omega, t_a)$, if $\omega \neq i$, or $\beta(\omega, t_a) = (a, \vec{b}_a, \vec{b}^+_a + \vec{u}(\omega))$, where $\beta'(\omega, t_a) = (a, \vec{b}_a, \vec{b}^+_a)$, otherwise. ☐

The proof of Theorem 11 now follows in a similar fashion as the proof of Theorem 2 for treewidth.

Towards Fixed Parameter Tractability. Theorem 7 excludes the existence of a fixed-parameter algorithm for (LOCALLY) ENVY-FREE ALLOCATION parameterized by the treewidth and a number of additional parameters of the instance under standard complexity assumptions. Hence it is natural to consider more restrictive parameterizations of the social network. Here we consider, as has been done for other difficult problems (Fellows et al. 2008), the vertex cover number as a stronger structural restriction.

By using the vertex cover number as our network (graph) parameter, we obtain a fixed-parameter algorithm for (LOCALLY) ENVY-FREE ALLOCATION. Recall that, based on Theorem 9, it is not possible to achieve tractability by restricting the structure of the graph alone—and to achieve our result, we parameterize by the number of item-types (analogously as in Theorem 1) and additionally by the bundle-size; here, the use of a stronger structural restriction allows us to circumvent the lower bound given by Theorem 7.

Theorem 16. (LOCALLY) ENVY-FREE ALLOCATION *is in* FPT *parameterized by vertex cover number of the social network, number of item-types, and bundle-size.*

Proof. Let $\mathcal{I} = (A, R, (\tau_a)_{a \in A}, G)$ be an instance of (LOCALLY) ENVY-FREE ALLOCATION. It is well known that we can compute a minimum-size vertex cover $X \subseteq A$ of G in FPT time w.r.t. the size of the vertex cover (Cygan et al. 2015). Note that because X is a vertex cover of G, for any $a \in A \setminus X$, $N^+_G(a) \subseteq X$ and $N^-_G(a) \subseteq X$.

Obviously we can branch on disjoint assignments (up to item-types) of at most k items to agents in X in time at most $|T_R|^{k \cdot \text{vcn}(G)}$. Denote this partial assignment by π. In every branch we attempt to extend π to $A \setminus X$ and make sure π is a solution to (LOCALLY) ENVY-FREE ALLOCATION for $\mathcal{I}$.

For each $a \in X$ we can check local envy-freeness of π in the current branch w.r.t to some other $a' \in X$ explicitly in time $|X|$. (If it is violated, we abandon the current branch.)

Because for any $a \in A \setminus X$, $N^+_G(a) \subseteq X$, envy-freeness at a only speaks about a and its preference of items assigned to vertices in X. These are already fixed by π. We compute (up to item-type) for each $a \in A \setminus X$ all bundles R_a of at most k elements such that $\tau_a(R_a) \geq \tau_a(\pi(x))$ for all $x \in N^+_G(a)$ and $\tau_a(R_a) \geq \frac{\tau_a(R \setminus \pi(N^+_G(a)))}{|A \setminus N^+_G(a)|}$ in time $|T_R|^k \cdot |X|$. Denote these bundles by $\mathcal{B}_a$. By construction, the conditions for envy-freeness will be satisfied by π for $a \in A \setminus X$, if and only if $\pi(a) \in \mathcal{B}_a$. Note that proportionality for agents in X is not ensured by this.

It remains to find assignments for each $a \in A \setminus X$ to bundles $R' \in \mathcal{B}_a$ in a way that they are pairwise disjoint and also disjoint to the fixed assignment of π on X, as well as guaranteeing proportionality for X under this assignment, in the case we are considering (non-locally) ENVY-FREE ALLOCATION. We do so by considering an integer linear program with a number of variables that we can bound in terms of $\text{vcn}(G)$, $|T_R|$ and k. For this we group the agents $a \in A \setminus X$ according to their respective $\mathcal{B}_a$. That is, we say $a, a' \in A \setminus X$ are in the same *group* if $\mathcal{B}_a = \mathcal{B}_{a'}$. Because there are (up to item-type) at most $|T_R|^k$ bundles of k items and each $\mathcal{B}_a$ is a set of such bundles, the number of groups can be bounded by $2^{|T_R| \cdot k}$. Let $\mathcal{G}_1, \dots \mathcal{G}_z$ with $z \leq 2^{|T_R| \cdot k}$ be an enumeration of all the groups of agents in $A \setminus X$. By $\mathcal{B}_\mathcal{G}$ we denote $\mathcal{B}_a$ for the agents $a \in \mathcal{G}$ in group $\mathcal{G}$. Now for each group $\mathcal{G}$, each of its bundles $R_G \in \mathcal{B}_\mathcal{G}$ and each $X' \subseteq X$, we introduce an integer variable $x_{\mathcal{G}, R_\mathcal{G}, X'}$ that will encode how many agents $a \in \mathcal{G}$ in group $\mathcal{G}$ with $N^-_G(a) = X'$ will be assigned bundle $R_\mathcal{G}$ up to item-types. We now solve the integer linear program with the following constraints:

Integrality constraints: For $\mathcal{G} \in \{\mathcal{G}_1, \dots, \mathcal{G}_z\}$, $R_\mathcal{G} \in \mathcal{B}_\mathcal{G}$ and $X' \subseteq X$,$x_{\mathcal{G}, R, X'} \in \mathbb{N}_0$;

Network conformity constraints: For $X' \subseteq X$ and $\mathcal{G} \in \{\mathcal{G}_1, \dots, \mathcal{G}_z\}$, $\sum_{R_\mathcal{G} \in \mathcal{B}_\mathcal{G}} x_{\mathcal{G}, R_\mathcal{G}, X'} = |(N^+_G(X') \setminus X) \cap \mathcal{G}|$;

Resource constraints: For $t_r \in T_R$, $\sum\limits_{X' \subseteq X} \sum\limits_{\mathcal{G} \in \{\mathcal{G}_1, \dots, \mathcal{G}_z\}} \sum\limits_{R_\mathcal{G} \in \mathcal{B}_\mathcal{G}} \vec{b}(R_\mathcal{G})[t_r] \cdot x_{\mathcal{G}, R_\mathcal{G}, X'} + \vec{b}(\pi(X))[t_r] = \vec{b}(R)[t_r]$;

Proportionality constraints: For $a \in X$, $\sum\limits_{\mathcal{G} \in \{\mathcal{G}_1, \dots, \mathcal{G}_z\}} \sum\limits_{R_\mathcal{G} \in \mathcal{B}_\mathcal{G}} \sum\limits_{X' \subseteq X \setminus \{a\}} \tau_a(R_\mathcal{G}) \cdot x_{\mathcal{G}, R_\mathcal{G}, X'} \leq$

$|A \setminus N_G^+(a)| \cdot \tau_a(\pi(a)) - \sum_{a' \in X \setminus N_G^+(a)} \tau_a(\pi(a')).$

One can easily see that a solution to (LOCALLY) ENVY-FREE ALLOCATION for $\mathcal{I}$ infers a solution to the ILP and conversely construct from a solution to the ILP an extension of π to $A \setminus X$ such that π is a solution to (LOCALLY) ENVY-FREE ALLOCATION for $\mathcal{I}$. Lenstra's celebrated result (1983) states that an ILP can be solved in FPT time parameterized by the number of its variables which is in our case $2^{|T_R| \cdot k}|T_R|^k \cdot 2^{\text{vcn}(G)}$. This concludes the proof. $\square$

Concluding Remarks

We initiated the study of resource allocation problems with social networks under natural restrictions to the networks and valuation functions. Our main results are polynomial-time algorithms for instances whose social networks have bounded treewidth or clique-width.

For future work, it would be interesting to study resource allocation problems without social networks parameterized by the number of item-types and agent-types. For instance, it is an interesting open question (which is also closely connected to similar questions about BIN PACKING), whether the envy-free versions of resource allocation are FPT when parameterized by the number of item-types.

Acknowledgments. Robert Ganian and Thekla Hamm acknowledge support from the Austrian Science Fund (FWF, Project P31336: **NFPC**). Thekla Hamm is co-funded by FWF project W1255-N23.

References

Abebe, R.; Kleinberg, J. M.; and Parkes, D. C. 2017. Fair division via social comparison. In *Proceedings of AAMAS 2017*, 281–289.

Aziz, H.; Bouveret, S.; Caragiannis, I.; Giagkousi, I.; and Lang, J. 2018. Knowledge, fairness, and social constraints. In *Proceedings of AAAI 2018*, 4638–4645.

Bei, X.; Qiao, Y.; and Zhang, S. 2017. Networked fairness in cake cutting. In *Proceedings of IJCAI 2017*, 3632–3638.

Beynier, A.; Chevaleyre, Y.; Gourvès, L.; Harutyunyan, A.; Lesca, J.; Maudet, N.; and Wilczynski, A. 2019. Local envy-freeness in house allocation problems. *Autonomous Agents and Multi-Agent Systems* 33(5):591–627.

Bliem, B.; Bredereck, R.; and Niedermeier, R. 2016. Complexity of efficient and envy-free resource allocation: Few agents, resources, or utility levels. In Kambhampati, S., ed., *Proceedings of IJCAI 2016*, 102–108. IJCAI/AAAI Press.

Bodlaender, H. L.; Drange, P. G.; Dregi, M. S.; Fomin, F. V.; Lokshtanov, D.; and Pilipczuk, M. 2016. A c^{k} n 5-approximation algorithm for treewidth. *SIAM J. Comput.* 45(2):317–378.

Bodlaender, H. L. 1996. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.* 25(6):1305–1317.

Bouveret, S., and Lang, J. 2008. Efficiency and envy-freeness in fair division of indivisible goods: Logical representation and complexity. *J. Artif. Intell. Res.* 32:525–564.

Bouveret, S.; Chevaleyre, Y.; and Maudet, N. 2016. Fair allocation of indivisible goods. In *Handbook of Computational Social Choice*. 284–310.

Brânzei, S.; Lv, Y.; and Mehta, R. 2016. To give or not to give: Fair division for single minded valuations. In *Proceedings of IJCAI 2016*, 123–129.

Bredereck, R.; Kaczmarczyk, A.; and Niedermeier, R. 2018. Envy-free allocations respecting social networks. In *Proceedings of AAMAS 2018*, 283–291.

Courcelle, B.; Makowsky, J. A.; and Rotics, U. 2000. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory Comput. Syst.* 33(2):125–150.

Cygan, M.; Fomin, F. V.; Kowalik, L.; Lokshtanov, D.; Marx, D.; Pilipczuk, M.; Pilipczuk, M.; and Saurabh, S. 2015. *Parameterized Algorithms*. Springer.

Diestel, R. 2012. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer.

Downey, R. G., and Fellows, M. R. 2013. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer Verlag.

Fellows, M. R.; Lokshtanov, D.; Misra, N.; Rosamond, F. A.; and Saurabh, S. 2008. Graph layout problems parameterized by vertex cover. In *ISAAC*.

Fellows, M. R.; Fomin, F. V.; Lokshtanov, D.; Rosamond, F.; Saurabh, S.; Szeider, S.; and Thomassen, C. 2011. On the complexity of some colorful problems parameterized by treewidth. *Information and Computation* 209(2):143–153.

Ganian, R.; Hlinený, P.; and Obdrzálek, J. 2013. Better algorithms for satisfiability problems for formulas of bounded rank-width. *Fundam. Inform.* 123(1):59–76.

Ganian, R.; Ordyniak, S.; and Rahul, C. S. 2019. Group activity selection with few agent types. In *Proceedings of ESA 2019*. To appear.

Kanté, M. M., and Rao, M. 2013. The rank-width of edge-coloured graphs. *Theory Comput. Syst.* 52(4):599–644.

Kloks, T. 1994. *Treewidth: Computations and Approximations*. Berlin: Springer Verlag.

Lenstra, H. W. J. 1983. Integer programming with a fixed number of variables. *Math. Oper. Res.* 8(4):538–548.

Marx, D. 2010. Can you beat treewidth? *Theory of Computing* 6:85–112.

Niedermeier, R. 2006. *Invitation to Fixed-Parameter Algorithms*. Oxford Lecture Series in Mathematics and its Applications. Oxford: Oxford University Press.

Oum, S., and Seymour, P. D. 2006. Approximating clique-width and branch-width. *J. Comb. Theory, Ser. B* 96(4):514–528.

Robertson, N., and Seymour, P. D. 1983. Graph minors. i. excluding a forest. *J. Comb. Theory, Ser. B* 35(1):39–61.

Thorup, M. 1998. All structured programs have small tree-width and good register allocation. *Inf. Comput.* 142(2):159–181.