Article:
Eiben, E, Ganian, R, Hamm, T et al. (1 more author) (2023) Parameterized Complexity of Envy-Free Resource Allocation in Social Networks. Artificial Intelligence, 315. 103826. ISSN 0004-3702

https://doi.org/10.1016/j.artint.2022.103826

Parameterized Complexity of Envy-Free Resource Allocation in Social Networks

Eduard Eiben[a,*], Robert Ganian[b,*], Thekla Hamm[b,*], Sebastian Ordyniak[c,*]

[a]*Department of Computer Science, Royal Holloway, University of London, UK*
[b]*Algorithms and Complexity Group, TU Wien, Austria*
[c]*Department of Computer Science, The University of Sheffield, UK*

Abstract

We consider the classical problem of allocating indivisible 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 resources. In particular, we show that even very general fragments of the considered problems become tractable as long as the social network has constant treewidth or clique-width. We complement our results with matching lower bounds which show that our algorithms cannot be substantially improved.

Keywords: resource allocation, parameterized complexity, computational social choice, treewidth, clique-width

1. Introduction

Envy-freeness ranks among the most important fairness requirements in the classical resource allocation problem of distributing resources (or items) among agents [10, 9]. There has also been an extensive line of works studying envy-free allocations of indivisible resources in a more general setting where agents only directly compare themselves to a subset of other agents [5, 2, 13]. 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. Similar ideas were also pursued for the allocation of *divisible* resources, specifically the classic CAKE CUTTING problem [1, 3], where agents have preferences over different parts of a "cake" (representing

*Corresponding authors
Email addresses: eduard.eiben@rhul.ac.uk (Eduard Eiben), rganian@gmail.com (Robert Ganian), thamm@ac.tuwien.ac.at (Thekla Hamm), sordyniak@gmail.com (Sebastian Ordyniak)

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 a full allocation of resources that is considered "envy-free" by each agent[1]. More specifically we study two well-established notions of envy and resulting variants of the problem:

(1) in GRAPH ENVY-FREE ALLOCATION (GEFA) each agent a is satisfied if it does not envy any of its neighbors—this notion of *graph envy-freeness* models situations where agents do not know or care about the total available number of resources. This variant was studied, e.g., by Beynier et al. [5] and Bredereck et al. [13].

(2) in GRAPH PROPORTIONAL ENVY-FREE ALLOCATION (GPEFA) each agent must not only be envy-free of its neighbors, but must also view its allocated resources as being "non-locally" proportional to the total number 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. GPEFA was proposed and studied by Aziz et al. [2].

Contribution. Unsurprisingly, both GEFA and GPEFA are NP-complete, and in fact remain NP-complete even on severely restricted instances [13]. For example, the well-known PARTITION PROBLEM can be encoded by GEFA and GPEFA on a complete bidirected social network with 2 agents using the same valuation. GEFA and GPEFA 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, GPEFA 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 [19, 16, 33] 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 GEFA and GPEFA would be. First of all, since both problems remain NP-complete even on very simple social networks (e.g., edgeless graphs for GPEFA;

[1] A formal definition of the problem is provided in Subsection 2.4.

Param. for A and R	Structural param. for G: tw(G)	cw(G)	–
–	paraNP-h	paraNP	NP-h
$\|A\|$	paraNP-h ([29])	paraNP-h	paraNP-h
$\|R\|$	FPT (Cor. 7)	XP	XP (brute force), W[1]-h ([13])
$\|T_A\|$	paraNP-h (Thm. 11)	paraNP-h	paraNP-h
$\|T_R\|$	XP (Thm. 1), W[1]-h	W[1]-h	paraNP-h
$\|A\| + \|R\|$	P	P	P (brute force)
$\|A\| + \|T_R\|$	XP	XP	XP
$\|T_A\| + \|R\|$	FPT	FPT (Cor. 18)	XP
$\|T_A\| + \|T_R\|$	XP, W[1]-h (Thm. 9)	XP (Thm. 12), W[1]-h	paraNP-h (Cor. 10)

Table 1: Overview of the parameterized complexity of GPEFA and GEFA under combinations of the structural parameters treewidth and cliquewidth of the social network, and the number of agents and resources and their types as parameters respectively. NP-h, paraNP-h and W[1]-h abbreviate hardness for the classes NP, paraNP and W[1] respectively. All stated results can easily be derived from the underlined ones using the relationships between the considered parameters. Note that there are several natural cases of parameter combinations left open. Specifically, it would be interesting to determine whether the XP-results for the parameter $|A|+|T_R|$ can be improved to FPT-results. The same is also left open for the parameter $|T_A|+|R|$ on arbitrary graphs, and the parameter $|R|$ on graphs of bounded clique-width. Maybe even more interesting from an algorithmic point of view, is the question whether there is an XP-algorithm parameterized by $|T_R| + \text{cw}(G)$.

see Theorem 11) 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 there are practical settings, in which one 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 GEFA and GPEFA 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 resource, 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 *resource-* and *agent-types*, which contain resources and agents that are indistinguishable from each other based on any valuation. Indeed, the numbers of resource- 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 [11, 23, 12, 32, 36].

Results. Our first result focuses on tree-like social networks, specifically social networks of constant *treewidth* tw [35, 19]. Specifically, we show that GEFA and GPEFA 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 programming procedure, which constructs a partial solution using so called *records*—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 9) 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 (Corollary 10 and Theorem 11) 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 a well-motivated graph parameter (see, for instance, the survey by Marx [30] and the result of Thorup linking treewidth and control flow graphs [37]), 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 GEFA and GPEFA are both in XP parameterized by $\text{cw}(G)+|T_A|+|T_R|$ (Theorem 12). 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.

An overview of the parameterized complexity classification of GPEFA and GEFA with respect to all combinations of the most important parameters in our work, i.e., $\text{tw}(G)$, $\text{cw}(G)$, $|A|$, $|R|$, $|T_A|$ and $|T_R|$, is given in Table 1. While in terms of these parameters our new reduction in Theorem 11 is not strictly speaking necessary to obtain this classification, the reduced instance satisfies other additional properties. Specifically, while the reduction from PARTITIONING described earlier also implies the NP-hardness of GPEFA and GEFA for a constant number of agent-types, Theorem 11 shows this, even for unary encodings of the valuations.

As our final result, we tackle the question of whether one can strengthen the polynomial-time result of Theorem 1 to fixed-parameter tractability for the problem by using a stronger[2] 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 resources assigned to any agent). Specifically, we combine integer linear programming with exhaustive branching to show that both GEFA and GPEFA are FPT when parameterized by the size of a minimum vertex cover of G, $|T_R|$, and the bundle-size (Theorem 19).

2. 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. For functions two $f, g : X \to \mathbb{N}$ from a set X to the set of natural numbers, we consider their sum, as defined in the natural way for each entry,

[2]A parameter A is stronger than a parameter B, if for every instance the value of the parameter B is bounded by (a function of) the value of the parameter A; i.e., treewidth is a stronger parameter than clique-width, because every graph with constant treewidth has also constant clique-width. In the literature, one sometimes also uses the equivalent phrase "treewidth is *dominated* by clique-width".

i.e., $f + g : X \to \mathbb{N}$ is given by $(f + g)(x) = f(x) + g(x)$ for $x \in X$. Similarly, we define $f \leq g$ to be true, whenever $f(x) \leq g(x)$ for all $x \in X$, and $f = g$ to be true, whenever $f(x) = g(x)$ for all $x \in X$. We refer to the handbook by Diestel [18] 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.

2.1. Parameterized Complexity

In parameterized algorithmics [19, 16, 33] the running-time of an algorithm is studied with respect to a parameter $k \in \mathbb{N}_0$ and input size n. The basic idea is to find a parameter that describes the structure of the instance such that the combinatorial explosion can be confined to this parameter. In this respect, the most favorable complexity class is FPT (*fixed-parameter tractable*) which contains all problems that can be decided by an algorithm running in time $f(k) \cdot n^{\mathcal{O}(1)}$, where f is a computable function. Algorithms with this running-time are called *fixed-parameter algorithms*. A less favorable outcome is an XP *algorithm*, which is an algorithm running in time $\mathcal{O}(n^{f(k)})$; problems admitting such algorithms belong to the class XP.

To obtain our lower bounds, we will need the notion of a *parameterized reduction* and the complexity class W[1]. A parameterized reduction is a variant of the standard polynomial-time reduction which retains bounds on the parameter, and W[1]-hardness rules out the existence of fixed-parameter algorithms under the Exponential Time Hypothesis [25]. The class W[1] captures parameterized intractability and contains all problems that are FPT-reducible to INDEPENDENT SET (parameterized by solution size). Parameterized problems which remain NP-hard even when the parameter is set to a specific constant are said to be paraNP-hard.

2.2. 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 $\mathrm{tw}(G)$, 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 [27]: A tree-decomposition (T, χ) is a *nice tree-decomposition* of a graph G if the tree T is rooted at node r, $\chi(r) = \emptyset$, and each node of T is of one of the following four types:

1. a *leaf node*: a node t having no children and $|\chi(t)| = 1$;
2. an *introduce node*: a node t having exactly one child t', and $\chi(t) = \chi(t') \cup \{v\}$ for a node v of G;

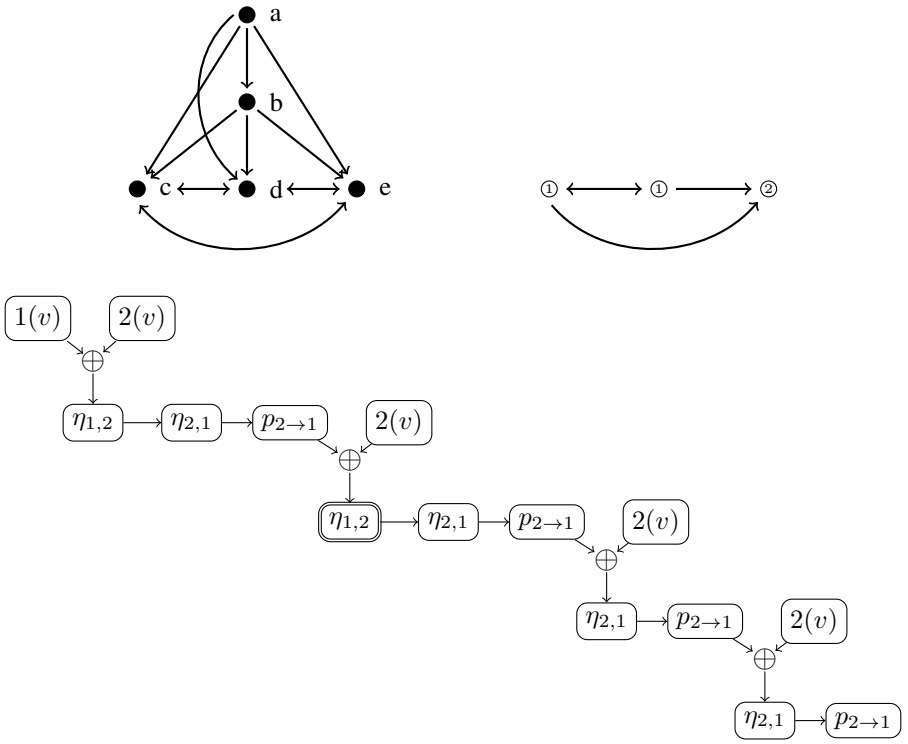


Figure 1: **Top left:** A directed graph G with $\mathrm{cw}(G) = 2$. **Bottom:** Depiction of an expression tree that witnesses that $\mathrm{cw}(G) = 2$. The expression tree corresponds to the expression $p_{2\to1}(\eta_{2,1}(2(v) \oplus p_{2\to1}(\eta_{2,1}(2(v) \oplus p_{2\to1}(\eta_{2,1}(\eta_{1,2}(2(v) \oplus p_{2\to1}(\eta_{2,1}(\eta_{1,2}(2(v) \oplus 1(v)))))))))))$ which uses the two initial 2-graphs $1(v)$ and $2(v)$ as basis. **Top right:** Directed graph resulting from the evaluation of the given expression tree up to and including the step $\eta_{1,2}$ with double borders. Here labels are drawn into the vertices.

3. a *forget node*: a node t having exactly one child t', and $\chi(t) = \chi(t') \setminus \{v\}$ for a node v of G;

4. a *join node*: a node t having 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 $\mathcal{O}(\mathrm{tw}(G) \cdot |V(G)|)$ many nodes and optimal width is fixed-parameter tractable, and FPT-approximation algorithms with better running times are also available [27, 7, 8].

2.3. 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 a 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 iterative 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 a directed edge (arc) from each vertex labeled by i to each vertex labeled by j ($i \neq j$; denoted by $\eta_{i,j}$).

Many graph classes are known to have constant clique-width; examples include all graph classes of constant treewidth [15] and co-graphs [15].

A *k-expression tree* [14] 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. An example is provided in Figure 1.

2.4. Problem Statement

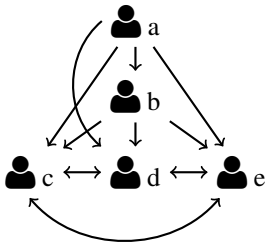
Let A be a set of agents, R be a set of resources, and G be a directed graph with vertex set A. A *preference function* (or *valuation function*) for an agent $a \in A$ is a function $\tau_a : 2^R \to \mathbb{N}$. Throughout the paper we will assume that preference functions are additive, i.e., $\tau_a(R')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

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

For brevity and ease of presentation, the term "proportionality" will always refer to the non-local notion of proportionality defined above. Note that this notion is different from the notion of "local proportionality" studied in some works [4]. We can now formalize our problems of interest:

GRAPH PROPORTIONAL ENVY-FREE ALLOCATION (GPEFA)	
Input:	A set A of agents, a set R of resources, 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 a graph proportional envy-free allocation?



	τ_a	τ_b	τ_c	τ_d	τ_e
$\mathbf{1}_1$	0	0	5	0	5
$\mathbf{1}_2$	0	0	5	0	5
$\mathbf{2}_1$	1	0	5	1	5
$\mathbf{2}_2$	1	0	5	1	5
$\mathbf{3}_1$	2	0	2	2	2
$\mathbf{3}_2$	2	0	2	2	2

Figure 2: An instance of GEFA with agents $A = \{a, b, c, d, e\}$, resources $R = \{\mathbf{1}_1, \mathbf{1}_2, \mathbf{2}_1, \mathbf{2}_2, \mathbf{3}_1, \mathbf{3}_2\}$, preference functions as given in the table on the right and G as depicted on the left. Note that $\mathbf{1}_1$ and $\mathbf{1}_2$, as well as $\mathbf{2}_1$ and $\mathbf{2}_2$, and $\mathbf{3}_1$ and $\mathbf{3}_2$ have the same resource-type. Similarly a and d, as well as c and e have the same agent-type.
In this example $\mathrm{tw}(G) = 4$ (a tree decomposition witnessing this is a single bag containing all vertices, moreover as all vertices are pairwise connected they must occur in a bag together), and $\mathrm{cw}(G) = 2$ (as G can be constructed from the two initial 2-graphs $1(v)$ and $2(v)$ as described in Figure 1).

GRAPH ENVY-FREE ALLOCATION (GEFA) is defined analogously, with the sole distinction that we ask for a graph 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.

2.5. *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 resources r and r' have the same *resource-type* if they are equally valued by any agent, i.e., if $\tau_a(r) = \tau_a(r')$ for every $a \in A$. Key notions that were introduced so far are illustrated in Figure 2.

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

We also call sets of resources $R' \subseteq R$ *bundles*. In the context of resource-types, we will also represent bundles by *bundle functions* $\mathrm{bfun} : T_R \to \mathbb{N}$, where $\mathrm{bfun}(t_\mathrm{r})$ is equal to the number of resources of resource-type t_r in the bundle represented by the function, for every $t_\mathrm{r} \in T_R$. We use $\mathrm{BFUN}(R')$ to denote the bundle function representing the bundle R' and conversely we denote by $\mathrm{BUN}(\mathrm{bfun})$ a bundle corresponding to the bundle function bfun. Moreover, we denote by $\mathcal{B} = \{\, \mathrm{BFUN}(R') \mid R' \subseteq R \,\}$, the set of all possible bundle functions.

Our results will mainly be concerned with establishing the tractability of GPEFA and GEFA under the combination of (1) a graph parameter that restricts the structure of the network and (2) the number of agent-types or resource-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.

Lastly, while our interest in the considered graph parameters is partly motivated by their fundamental nature, we also illustrate some natural examples where these parameters may be small:

- Consider a situation where we modify the basic envy-free allocation problem (in which all agents could envy each other) by partitioning agents into disjoint "families" whereas members of the same family will not envy each other. This results in an instance of GEFA where the graph is obtained by taking a bidirectional clique and then complementing all arcs between vertices in each family. It is a simple exercise to show that such graphs have clique-width at most 2 (one can adapt the standard clique construction by introducing whole families together instead of adding vertices one by one). In particular, suppose we want to build the 2-expression tree for an instance with 3 families A, B, and C whose sets of members are given by $\{a_1, \ldots, a_{|A|}\}$, $\{b_1, \ldots, b_{|B|}\}$, and $\{c_1, \ldots, c_{|C|}\}$, respectively. Then, the 3-expression tree is given by:

  $$p_{2\to 1}(\eta_{1,2}(\eta_{2,1}((p_{2\to 1}(\eta_{1,2}(\eta_{2,1}(1(A)) \oplus (2(B))))) \oplus (2(C)))))$$

  Here, the expression $i(A)$ (and similarily the expressions $i(B)$ and $i(C)$) introduce all members of the family A using label i, i.e., $i(A)$ is a shortcut for the expression $i(a_1) \oplus i(a_2) \oplus \cdots \oplus i(a_{|A|})$.

- Imagine that agents represent employees in a company who are organized in a hierarchical manner, and agents can only envy agents on the same or lower levels of the company hierarchy (it is common that superiors receive more resources than subordinates). This situation is captured by instances of GEFA where agents are partitioned into hierarchical "levels" $1, \ldots, q$ and agents on level $i \in [q]$ envy all agents on level j where $i \leq j \leq q$. Again, it is not difficult to see that the resulting directed graphs have clique-width at most 2 (one can construct each level individually as a bidirectional clique, and then introduce the cliques iteratively from the lowest to the highest level). This construction can also be extended to tree-like hierarchies. In particular, suppose we want to construct an instance with three levels $A = \{a_1, \ldots, a_{|A|}\}$, $B = \{b_1, \ldots, b_{|B|}\}$, and $C = \{c_1, \ldots, c_{|C|}\}$ with A being the highest level in the hierarchy. Then, our first step is to build a 2-expression that introduces all members of a level and the edges between. This is done by the expression $L(Q, i)$ (given below), where Q is a set containing all members of the level Q and i is a label. The expression, which is given below for the case that $Q = \{q_1, q_2, q_3\}$, introduces the q_i's and the edges between them and ensures that all q_i's have label i at the end of the expression.

  $$p_{1\to i}(p_{2\to 1}(\eta_{2,1}(\eta_{1,2}(p_{2\to 1}(\eta_{2,1}(\eta_{1,2}(1(q_1) \oplus 2(q_2)))) \oplus 2(q_3)))))$$

  We are now ready to give our 2-expression for our instance with 3 levels.

  $$p_{2\to 1}(\eta_{1,2}(p_{2\to 1}(\eta_{1,2}(L(C, 1) \oplus L(B, 2))) \oplus L(A, 2)))$$

- It is known that in many cases, real-world networks are structurally sparse and often have, e.g., *bounded expansion* [17], which is a general (more general than treewidth) and powerful measure of graph sparsity. While it is not known whether more general measures of sparsity can be algorithmically exploited for GRAPH (PROPORTIONAL) ENVY-FREE ALLOCATION, the notion of treewidth studied here can be seen as a more restrictive measure of graph sparsity [31].

- Finally, consider a situation where employees on the same position in a company receive certain benefits, but the benefits each employee receives is hidden (this is often the case, e.g., for employee salaries). However, the benefits of k employees were accidentally revealed to everyone, and hence all employees could now hypothetically envy any of the k employees whose benefits are known. This can be modeled as an instance of GRAPH (PROPORTIONAL) ENVY-FREE ALLOCATION where the graph has a vertex cover of size k (in particular, the k agents whose resources were revealed would form the vertex cover).

3. Allocating Resources on Tree-Like Networks

As our first result, we show that instances with a constant number of resource-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. GRAPH (PROPORTIONAL) ENVY-FREE ALLOCATION *is in* XP *parameterized by the treewidth of the social network and number of resource-types.*

For the remainder of this section, let $\mathcal{I} = (A, R, (\tau_a)_{a \in A}, G)$ be an instance of GRAPH (PROPORTIONAL) ENVY-FREE ALLOCATION. Since a nice tree-decomposition of a graph can be computed efficiently [27, 7, 8], it suffices to solve the problem when a minimum-width nice tree-decomposition of G is provided as part of the input.

Lemma 2. GRAPH (PROPORTIONAL) ENVY-FREE ALLOCATION *can be solved in time at most* $|R|^{\mathcal{O}(|T_R| \cdot \mathrm{tw}(G))}|A|$ *if a minimum-width nice tree-decomposition of* G *is provided as part of 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 graph proportional 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 $(\alpha, \tilde{u}, \beta)$, where:

- $\alpha : \chi(t) \to \mathcal{B}$ is a function that encodes an allocation for every agent in $\chi(t)$ via the corresponding bundle functions,
- $\tilde{u} \in \mathcal{B}$ corresponds to the bundle function of the bundle containing all resources already assigned to the agents in $\chi(T_t)$,

- $\beta : \chi(t) \to \mathcal{B}$ is a function that for each agent $a \in \chi(t)$, provides the bundle function of the bundle containing all resources 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 GEFA).

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

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

(R2) $\tilde{u} = \text{BFUN}(\pi(\chi(T_t)))$,

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

(R4) π is graph 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 GPEFA) 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 $\tau_a(\pi(a)) \geq \frac{\tau_a(R \setminus \pi(N_G^+(a)))}{|A \setminus N_G^+(a)|}$.

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 graph proportional envy-free allocation if and only if $\mathcal{R}(r)$, for the root r of T, contains a record $(\alpha, \tilde{u}, \beta)$ such that $\tilde{u} = \text{BFUN}(R)$; note that because $\chi(r) = \emptyset$ the functions α and β are empty functions and moreover (R5) is satisfied for all agents in the instance. Moreover, once we have computed the set of records for all nodes, a straightforward application of standard techniques [19] can be used to obtain a graph proportional 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 $\mathcal{O}(|\mathcal{B}|)$.*

Proof. Let $\chi(t) = \{a\}$. Then $\mathcal{R}(l)$ contains all records $(\alpha, \tilde{u}, \beta)$ such that:

- $\alpha(a) \in \mathcal{B}$,
- $\tilde{u} = \alpha(a)$, and
- $\beta(a) = \text{BFUN}(\emptyset)$,

The correctness of this construction of $\mathcal{R}(l)$ follows immediately from the definition of the records and the running-time is dominated by the number of possible choices for $\alpha(a)$, i.e., $|\mathcal{B}|$. □

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 $\mathcal{O}(|\mathcal{R}(t')||\mathcal{B}||T_R|\text{tw}(G))$.*

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. We say that a record $(\alpha', \tilde{u}', \beta') \in \mathcal{R}(t')$ and a bundle function $\mathrm{bfun}_a : T_R \to \mathbb{N}$ are *compatible* if they satisfy the following properties:

- there are still sufficiently many resources available for agent a, i.e., $\tilde{u}' + \mathrm{bfun}_a \leq \mathrm{BFUN}(R)$,
- no agent in $\chi(t')$ envies the agent a, i.e., for every $a' \in \chi(t')$ with $a \in N_G^+(a')$, it holds that $\tau_{a'}(\mathrm{BUN}(\alpha'(a'))) \geq \tau_{a'}(\mathrm{BUN}(\mathrm{bfun}_a))$,
- agent a does not envy any of its neighbors in $\chi(t')$, i.e.,

  $$\tau_a(\mathrm{BUN}(\mathrm{bfun}_a)) \geq \tau_a(\mathrm{BUN}(\alpha(a')))$$

  for every $a' \in \chi(t') \cap N_G^+(a)$;

Then, for every record $(\alpha', \tilde{u}', \beta') \in \mathcal{R}(t')$ and compatible bundle function $\mathrm{bfun}_a \in \mathcal{B}$, we construct $\mathcal{R}(t)$ to contain the record $(\alpha, \tilde{u}, \beta)$ such that:

- α is the extension of α' that allocates bfun_a to a, i.e., $\alpha(a) = \mathrm{bfun}_a$ and $\alpha(a') = \alpha'(a')$ for every $a' \in \chi(t')$,
- $\tilde{u} = \tilde{u}' + \mathrm{bfun}_a$,
- $\beta(a)$ is assigned to the sum of allocations assigned to all out-neighbors of a in $\chi(T_t)$ (which must be contained in $\chi(t')$), i.e., $\beta(a) = \sum_{a' \in N_G^+(a) \cap \chi(t')} \alpha'(a')$,
- for every $a' \in \chi(t')$ we need to add bfun_a to $\beta'(a')$ if a is an out-neighbor of a'. That is, we define $\beta(a')$ by setting either:
  - $\beta(a') = \beta'(a')$, if $a \notin N_G^+(a')$, or
  - $\beta(a') = \beta'(a') + \mathrm{bfun}_a$, otherwise;

Note that all neighbors of a in $\chi(T_t)$ are inside of $\chi(t')$. This is because by the properties of tree decompositions, both a and any of its out-neigbors u in $\chi(T_t)$ have to occur in $\chi(s)$ together for some $s \in V(T)$. If s is not a node of T_t, then $u \in \chi(t)$ must hold, as $u \in \chi(T_t)$ and otherwise $\{s' \in V(T) \mid u \in \chi(s')\}$ is not connected contradicting the properties of tree decompositions. If this s is in T_t, then it can only be equal to t, as t is the only node of T_t whose associated bag contains a. In both cases $u \in \chi(t)$.

Informally, the correctness of this construction of $\mathcal{R}(t)$ follows from this simple fact and the definition of the records.

More formally, we have to show that $(\alpha, \tilde{u}, \beta) \in \mathcal{R}(t)$ if and only if there is a record $(\alpha', \tilde{u}', \beta') \in \mathcal{R}(t')$ and a bundle function $\mathrm{bfun}_a : T_R \to \mathbb{N}$ that are compatible with each other and $(\alpha, \tilde{u}, \beta)$ is obtained from $(\alpha', \tilde{u}', \beta')$ and bfun_a as described above. So suppose that $(\alpha, \tilde{u}, \beta) \in \mathcal{R}(t)$. Because $(\alpha, \tilde{u}, \beta) \in \mathcal{R}(t)$ there is an allocation $\pi : \chi(T_t) \to 2^R$ satisfying (R1)–(R5). Let $(\alpha', \tilde{u}', \beta')$ be the record defined by setting:

- $\alpha'(a') = \alpha(a')$ for every $a' \in \chi(t')$,

- $\tilde{u}' = \tilde{u} - \alpha(a)$, and
- $\beta'(a') = \beta(a')$ for every $a' \in \chi(t')$ with $a \notin N_G^+(a')$ and $\beta'(a') = \beta(a') - \alpha(a)$, otherwise.

Then, $(\alpha, \tilde{u}, \beta)$ can be obtained from $(\alpha', \tilde{u}', \beta')$ and $\mathrm{bfun}_a = \alpha(a)$ using the above construction and because $(\alpha, \tilde{u}, \beta)$ is valid, $(\alpha', \tilde{u}', \beta')$ and $\mathrm{bfun}_a = \alpha(a)$ are compatible. Finally, the restriction $\pi' : \chi(T_{t'}) \to 2^R$ of π to $\chi(T_{t'})$ witnesses that $(\alpha', \tilde{u}', \beta') \in \mathcal{R}(t')$.

Towards showing the reverse direction, suppose that there is a record $(\alpha', \tilde{u}', \beta') \in \mathcal{R}(t')$ and a bundle function $\mathrm{bfun}_a : T_R \to \mathbb{N}$ that are compatible with each other and let $(\alpha, \tilde{u}, \beta)$ be the record obtained from $(\alpha', \tilde{u}', \beta')$ and bfun_a as described above. Because $(\alpha', \tilde{u}', \beta') \in \mathcal{R}(t')$ there is an allocation $\pi' : \chi(T_{t'}) \to 2^R$ satisfying (R1)–(R5). But then because $(\alpha', \tilde{u}', \beta')$ and bfun_a are compatible and all out-neighbors of a in $\chi(T_t)$ are in $\chi(t')$, we obtain that the allocation $\pi : \chi(T_t) \to 2^R$ obtained from π' and bfun_a by setting $\pi(a') = \pi(a')$ for every $a' \in \chi(T_{t'})$ and $\pi(a) = \mathrm{bfun}_a$ witnesses that $(\alpha, \tilde{u}, \beta) \in \mathcal{R}(t)$.

Finally, the running-time of the procedure follows because there are $|\mathcal{R}(t')||\mathcal{B}|$ compatible pairs of records in $\mathcal{R}(t')$ and bundle functions bfun_a, and checking the above properties for each of them takes time $\mathcal{O}(|T_R|\mathrm{tw}(G))$. □

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 $\mathcal{O}(|\mathcal{R}(t')||R|)$.*

Proof. Let a be the unique agent in $\chi(t') \setminus \chi(t)$. Informally, a record in $\mathcal{R}(t)$ is obtained from the restriction of a record r' in $\mathcal{R}(t')$ to $\chi(t)$—whereas in case of GPEFA one also checks that the record r' satisfies (R5), i.e., proportionality, for the agent a. More formally, we say that a record $(\alpha', \tilde{u}', \beta') \in \mathcal{R}(t')$ satisfies proportionality (for the agent a) if:

$$\tau_a(\alpha'(a)) \geq \frac{\tau_a(R \setminus \mathbf{BUN}(\beta'(a)))}{|A \setminus N_G^+(a)|}$$

Moreover, we define the restriction $(\alpha, \tilde{u}, \beta)$ of a record $(\alpha', \tilde{u}', \beta') \in \mathcal{R}(t')$ to $\chi(t)$ by setting: $\alpha(a) = \alpha'(a)$, $\tilde{u} = \tilde{u}'$, and $\beta(a) = \beta'(a)$ for every $a \in \chi(t)$. Then, the set $\mathcal{R}(t)$ contains the restriction of every record in $\mathcal{R}(t')$ that satisfies proportionality. The correctness follows immediately from the definition of the records together with the fact that the forgotten agent a has no neighbors in $G \setminus \chi(T_t)$. Finally, the running-time follows because we can check whether a record in $\mathcal{R}(t')$ satisfies proportionality in time $\mathcal{O}(|R|)$. □

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 $\mathcal{O}(|\mathcal{R}(t_1)||\mathcal{R}(t_2)||T_R|\mathrm{tw}(G))$.*

Proof. We say that two records $(\alpha_1, \tilde{u}_1, \beta_1) \in \mathcal{R}(t_1)$ and $(\alpha_2, \tilde{u}_2, \beta_2) \in \mathcal{R}(t_2)$ are *compatible* if they satisfy the following properties:

- $\alpha_1 = \alpha_2$, i.e., all agents in the bag $\chi(t)$ are assigned the equivalent bundles in terms of resource types,

- $\tilde{u}_1 + \tilde{u}_2 - \sum_{a \in \chi(t)} \alpha_1(a) \leq \text{BFUN}(R)$, i.e., there are sufficient resources of every type available;

Then, for every two compatible records $(\alpha_1, \tilde{u}_1, \beta_1) \in \mathcal{R}(t_1)$ and $(\alpha_2, \tilde{u}_2, \beta_2) \in \mathcal{R}(t_2)$, the set $\mathcal{R}(t)$ contains the record $(\alpha, \tilde{u}, \beta)$ satisfying:

(J1) $\alpha = \alpha_1 = \alpha_2$,

(J2) $\tilde{u} = \tilde{u}_1 + \tilde{u}_2 - \sum_{a \in \chi(t)} \alpha(a)$,

(J3) $\beta(a) = \beta_1(a) + \beta_2(a) - (\sum_{a' \in N_G^+(a) \cap \chi(t)} \alpha(a'))$ for every $a \in \chi(t)$;

Informally, the correctness of the construction follows from the definition of the records and the fact that there are no edges between agents in $\chi(T_{t_1}) \setminus \chi(t)$ and agents in $\chi(T_{t_2}) \setminus \chi(t)$.

More formally, we need to show that a record $(\alpha, \tilde{u}, \beta) \in \mathcal{R}(t)$ if and only if there are two compatible records $(\alpha_i, \tilde{u}_i, \beta_i) \in \mathcal{R}(t_i)$ satisfying (J1)–(J3).

So suppose that $(\alpha, \tilde{u}, \beta) \in \mathcal{R}(t)$ and let $\pi : \chi(T_t) \to 2^R$ be an allocation witnessing this. Let $\pi_i : \chi(T_{t_i}) \to 2^R$ be the restriction of π to $\chi(T_{t_i})$ for every $i \in \{1, 2\}$. Then, π_i witnesses that the record $(\alpha_i, \tilde{u}_i, \beta_i)$ is in $\mathcal{R}(t_i)$, where $\alpha_i(a) = \pi_i(a)$, $\beta_i(a) = \pi(N_G^+(a) \cap \chi(T_{t_i}))$ for every $a \in \chi(t_i)$ and $\tilde{u}_i = \pi_i(\chi(T_{t_i}))$. Moreover, $(\alpha_1, \tilde{u}_1, \beta_1)$ and $(\alpha_2, \tilde{u}_2, \beta_2)$ are compatible because π_1 and π_2 are restrictions of π only overlapping in $\chi(t)$. Finally, it is straightforward to verify that $(\alpha, \tilde{u}, \beta)$ and the records $(\alpha_i, \tilde{u}_i, \beta_i)$ satisfy (J1)–(J3).

Towards showing the reverse direction, let $(\alpha_i, \tilde{u}_i, \beta_i) \in \mathcal{R}(t_i)$ for $i \in \{1, 2\}$ be two compatible records and let $(\alpha, \tilde{u}, \beta)$ be the record obtained from them that satisfies (J1)–(J3). Let $\pi_i : \chi(T_{t_i}) \to 2^R$ be an allocation witnessing that $(\alpha_i, \tilde{u}_i, \beta_i) \in \mathcal{R}(t_i)$. Then, because $(\alpha_1, \tilde{u}_1, \beta_1)$ and $(\alpha_2, \tilde{u}_2, \beta_2)$ are compatible, the function $\pi : \chi(T_t) \to 2^R$ defined by setting $\pi(a) = \pi_1(a) = \pi_2(a)$ for every $a \in \chi(t)$ and $\pi(a) = \pi_1(a)$ for every $a \in \chi(T_{t_i}) \setminus \chi(t_i)$ is a well-defined allocation. Moreover, because there are no edges between $\chi(T_{t_1}) \setminus \chi(t)$ and $\chi(T_{t_2}) \setminus \chi(t)$ and because $(\alpha, \tilde{u}, \beta)$ satisfies (J1)–(J3), we obtain that π witnesses that $(\alpha, \tilde{u}, \beta) \in \mathcal{R}(t)$.

Finally, the running-time can be obtained by observing that there are at most $|\mathcal{R}(t_1)||\mathcal{R}(t_2)|$ compatible pairs of records and for each of them, we require time at most $\mathcal{O}(|T_R|\text{tw}(G))$. □

We are now ready to establish our main theorem.

Proof of Lemma 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 $(\alpha, \tilde{u}, \beta)$ such that $\tilde{u} = \text{BFUN}(R)$. Note that the correctness of the algorithm follows from 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 $\text{tw}(G) \cdot |A|$ [16, 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. □

From the running time of Theorem 1 and the fact that $|T_R| \leq |R|$, we also obtain an FPT result when we parameterize by the number of resources instead of the number of resource-types.

Corollary 7. GRAPH (PROPORTIONAL) ENVY-FREE ALLOCATION *is in* FPT *parameterized by the treewidth of the social network and number of resources.*

4. 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 GEFA and GPEFA are W[1]-under this parameterization.

In the following we denote edges in an undirected graph as sets of their two endpoints. This mean an undirected edge between vertices u and v of G is denoted by $\{u, v\}$.

The problem we reduce from is defined as follows.

EQUITABLE COLORING	
Input:	A graph G, an integer $q \in \mathbb{N}$.
Question:	Is there an *equitable coloring* of G, i.e. a function $\psi : V(G) \to [q]$ such that for every $\{u, v\} \in E(G)$ it holds $\psi(u) \neq \psi(v)$ and for all $i, j \in [q]$ it holds $\lvert\psi^{-1}(i)\rvert - \lvert\psi^{-1}(j)\rvert \leq 1$?

Fact 8 ([20])**.** EQUITABLE COLORING *is* W[1]*-hard parameterized by* $\mathrm{tw}(G)+q$*, even when restricted to cases where* $|V(G)|$ *is divisible by* q*.*

Theorem 9. GRAPH (PROPORTIONAL) ENVY-FREE ALLOCATION *is* W[1]*-hard parameterized by treewidth, number of resource-types, and number of agent-types, and bundle-size.*

Proof. The proof is very similar for both GPEFA and GEFA.

Given an instance (G, q) of EQUITABLE COLORING in which $|V(G)|$ is a multiple of q, we will construct an instance of $\mathcal{I} = (A, R, (\tau_a)_{a \in A}, G')$ of GRAPH (PROPORTIONAL) 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 $e = \{u, v\} \in E(G)$:
  - a set $A_e^u = \{\, a_{e,u}^i \mid i \in [q] \,\}$,
  - a set $A_e^v = \{\, a_{e,v}^i \mid i \in [q] \,\}$,
  - a set $A_e = \{\, a_e^{i,j} \mid i, j \in [q], i \neq j \,\}$,
- a set A_d of $10q^2(|V(G)| + |E(G)|)$ agents, and
- a single agent a_d.

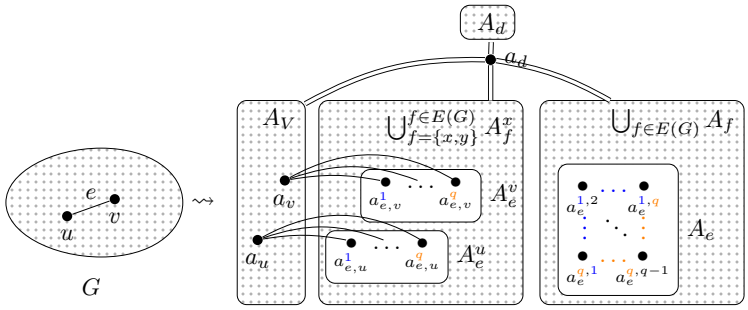


Figure 3: Social network in proof of Theorem 9; specifically for an edge $e = \{u, v\}$. Double lines denote complete connectivity and edges exist between vertices of the middle and right rectangle if they share a same-color (upper) index.

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:

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

An illustration of the graph is given in Figure 3. Note that A_d is an independent set.

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

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

Note that the number of resources is the same as the number of agents, so in any graph proportional envy-free solution, every agent is assigned at least 1 object.

Preferences. We have the following types of agents (see also Table 2):

1. For $a \in A_V$:
   - If $r \in \{r_d\} \cup \bigcup_{c\in[q]} R_c$, we let $\tau_a(r) = 1$.
   - If $r \in R \setminus \bigcup_{c\in[q]} R_c$, we let $\tau_a(r) = 0$.

Agent Type \ Resource Type	R_c	$R_{c'}$	$R_\square$	R_E	D	U	R_d	r_d
A_V	1	1	0	0	0	0	0	1
$a^c_{e,u}$	2	0	3	1	0	0	0	1
A_e	0	0	0	2	3	1	0	1
A_d	0	0	0	0	0	0	1	1
a_d	0	0	0	0	0	0	0	1

Table 2: The table of preferences of agents for resources depending on the agent and resource-type. The agent type $a^c_{e,u}$ represents an agent in A^u_e associated with the color $c \in [q]$. The resource-type R_c represent the resource-type associated with the color that matches the color of $a^c_{e,u}$, $R_{c'}$ represent the resource type associated with any other color that is $c' \in ([q] \setminus \{c\})$. Note that only agents $a^c_{e,u}$ in some of the sets A^u_e, for some edge $e \in E(G)$ and some vertex $u \in e$ have a different preference for R_c and $R_{c'}$.

2. For each $c \in [q]$, $v \in V(G)$, $e = \{u, v\} \in E(G)$ and $a = a^c_{e,v}$:
   - 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 $\tau_a(r) = 1$.
   - Else we let $\tau_a(r) = 0$. Note that this includes $r \in R_{c'}$ for $c' \neq c$.
3. For $a \in A_e$ for some $e \in E(G)$:
   - If $r \in D$, we let $\tau_a((r)) = 3$.
   - If $r \in R_E$, $c \in [r]$, we let $\tau_a(r) = 2$.
   - If $r \in \{r_d\} \cup U$, we let $\tau_a(r) = 1$.
   - Else we let $\tau_a(r) = 0$.
4. for $a \in A_d$:
   - If $r \in \{r_d\} \cup R_d$, we let $\tau_a((r)) = 1$.
   - Else we let $\tau_a(r) = 0$.
5. for $a = a_d$:
   - If $r = r_d$, we let $\tau_a((r)) = 1$.
   - Else we let $\tau_a(r) = 0$.

This concludes the construction. 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 GRAPH (PROPORTIONAL) 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 resource $r \in S$. Now, let $\psi : V(G) \to [q]$ be an equitable coloring of G. We define an graph proportional envy-free allocation $\pi : A \to 2^R$ as follows:

- For $v \in V(G)$, we let $\pi(a_v) \in R_{\psi(v)}$. This is possible as in an equitable coloring of an instance in which $|V(G)|$ is divisible by q, at most $\frac{|V(G)|}{q}$ vertices in $V(G)$ that share the same color, and by construction each $R_{\psi(v)}$ contains sufficiently many resources.

- For $v \in V(G)$, $e = \{u, v\} \in E(G)$ and $i \in [q]$, we let $\pi(a^i_{e,v}) \in R_E$ if $i \neq \psi(v)$ and $\pi(a^i_{e,v}) \in R_\square$ otherwise. This is possible as $\sum v \in V(G) \sum_{e=\{u,v\} \in E(G)} 1 = \sum v \in V(G) \deg_G(v) = 2|E(G)|$ and there is exactly one $i \in [q]$ with $i = \psi(v)$. By construction both R_E and $R_\square$ are appropriately sized.

- For $e = \{u, v\} \in E(G)$ and $i, j \in [q]$ with $i \neq j$, we let $\pi(a^{i,j}_e) \in U$ if $i = \psi(u)$ and $j = \psi(v)$ and we let $\pi(a^{i,j}_e) \in D$ otherwise. This is possible as $\sum e = \{u, v\} \in E(G) \sum_{i \neq j \in [q]} 1 = |E(G)|(q^2 - q)$ and for each edge $e = \{u, v\} \in E(G)$ there is exactly one combination of i and j such that $\psi(u) = i$ and $\psi(v))j$. By construction both U and D are appropriately sized.

- For $a \in A_d$, we let $\pi(a) \in R_d$. This is trivially possible as by construction $|A_d| = |R_d|$.

- We let $\pi(a_d) = \{r_d\}$.

The graph envy-freeness of the allocation π is straightforward to verify. For the proportionality, note that every agent a values her assigned resource at least 1. Moreover, $\tau_a(A) \leq \max\{3 \cdot |R \setminus R_d|, |R_d|\} \leq 10q^2(|V(G)| + |E(G)|)$ and $|A \setminus N^+_G(a)| \geq |A_d| = 10q^2(|V(G)| + |E(G)|)$. Hence, the allocation is proportional.

For the backward direction, let π be a graph envy-free allocation (note that every graph proportional envy-free allocation is also graph envy-free). We will show that the agents in A_V are assigned precisely the resources in $\bigcup_{c \in [q]} R_c$ such that the coloring ψ defined as

$$\psi(v) = c \text{ if and only if } \pi(a_v) \in R_c$$

is an equitable coloring of G by q colors. Note that a_d is adjacent to every other agent and and she desires only r_d. Hence, a_d is assigned the resource r_d by graph envy-freeness of π. Moreover, every other agent values r_d with value 1, hence every agent has to be assigned an resource in $R \setminus \{r_d\}$ that she values at least 1. Since the number of resources is the same as the number of agents, it is easy to see that every agent is assigned precisely one resource.

Now observe that agents in A_V value only the resources in $\bigcup_{c \in [q]} R_c$. Similarly, every agent in A_d is assigned in resource in R_d. Now let $v \in V(G)$, $e \in E(G)$ be an edge incident with v, and let $c \in [q]$ be such that $\pi(a_v) \in R_c$. It follows from graph envy-freeness that $a^c_{e,v} \in R_\square$, because all resources in R_c are already assigned to agents in A_V and if we assigned to $a^c_{e,v}$ anything else, she would envy the agent a_c for an resource in R_c. As there are precisely $2|E(G)|$ resources in $R_\square$ and agents in A^v_e value only resources in $R_\square \cup R_E \cup \bigcup_{c \in [q]} R_c$ it follows that all the other agents in A^v_e are assigned resources in R_E and agents in A_e are assigned resources either in D or in U. Furthermore, as argued above, there is exactly one pair $i, j \in [q]$ for each $e = \{u, v\}$ such that both $\pi(a^i_{e,u})$ and $\pi(a^j_{e,v})$ contain an resource in $R_\square$ and it is

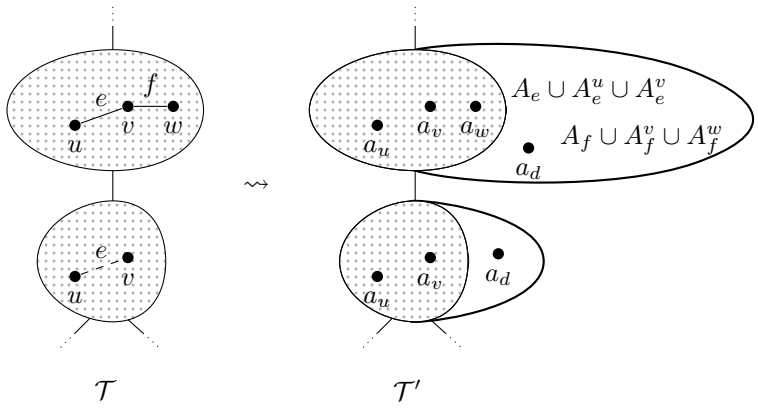


Figure 4: Illustration for the construction of a tree decomposition $\mathcal{T}'$ (right) of G' from a tree decomposition $\mathcal{T}$ (left) of G in the proof of Theorem 9 witnessing the desired treewidth bound. In this snippet, the construction is indicated for $\phi(e) = \phi(f)$ as the top depicted bag. Note that it is also valid to choose $\phi(e)$ to be the bottom depicted bag. The dotted pattern indicates parts of the bags which are in a one-to-one correspondence within the two tree decompositions.

precisely when $\pi(a_u) \in R_i$ and $\pi(a_v) \in R_j$. In all other cases at least one neighbor of $a_e^{i,j}$ is assigned an resource in R_E and by graph envy-freeness $\pi(a_e^{i,j}) \in D$. Therefore, in order to assign all resources in U, we have to assign precisely one resource from U to an agent in A_e for each e. Moreover, it has to be the agent $a_e^{i,j}$ such that $\pi(a_u) \in R_i$ and $\pi(a_v) \in R_j$. Note, that it follows that $i \neq j$ and assigning to $v \in V(G)$ color c if $\pi(a_v) \in R_c$ is a proper coloring of G. Finally, it is also an equitable coloring, because the sizes of R_c for all $c \in [q]$ are precisely $\frac{|V(G)|}{q}$.

It remains to show that the parameters are small. Clearly, the number of resource-types is $q + 6$, the number of agent-types is $q + 4$, and the maximum bundle-size is 1. To show that treewidth remains small let $\mathcal{T} = (T, \chi)$ be a tree-decomposition of G of width $\mathrm{tw}(G)$.

Now we construct tree-decomposition $\mathcal{T}' = (T', \chi')$ of G' as follows (see Figure 4 for a schematic illustration). We let $T' = T$. Now let $\phi : E(G) \to T$ be an arbitrary function such that for every edge $e \in E(G')$, $e \subseteq \chi(\phi(e))$; that is, ϕ assigns an edge e a node $t \in T$ such that $u, v \in \chi(t)$. For each $t \in V(T)$, we let $\chi'(t) = \{\, a_d \,\} \cup \{\, a_v \mid v \in$

$\chi(t)\,\} \cup \{\, a \mid \exists e = \{u, v\} \in \phi^{-1}(t) \quad a \in A_e \cup A_e^u \cup A_e^v \,\}$. In this way

$$\begin{aligned}
|\chi'(t)| &= 1 + |\{\, a_v \mid v \in \chi(t)\,\}| + |\{\, a \mid a \in A_e \cup A_e^u \cup A_e^v; e = \{u, v\} \in \phi^{-1}(t)\,\}| \\
&= 1 + |\chi(t)| + \sum_{e \in \phi^{-1}(t)} (|A_e| + |A_e^u| + |A_e^v|) \\
&= 1 + |\chi(t)| + \sum_{e \in \phi^{-1}(t)} (q(q-1) + 2q) \\
&= 1 + |\chi(t)| + \sum_{e \in \phi^{-1}(t)} (q(q-1) + 2q) \\
&= 1 + |\chi(t)| + |\phi^{-1}(t)|(q(q-1) + 2q) \\
&\leq 2 + \mathrm{tw}(G) + \binom{\mathrm{tw}(G)+1}{2}(q^2 + 2q).
\end{aligned}$$

Hence the size of each bag of the decomposition $\mathcal{T}'$ is bounded by a function of the parameter. Finally we show that $\mathcal{T}'$ satisfies the properties of a tree-decomposition.

First note that all edges in G' are between vertices in $\{a_u, a_v\} \cup A_e \cup A_e^u \cup A_e^v$ for some $e = \{u, v\} \in E(G)$. And by the construction of $\mathcal{T}'$ there is a node t such that $\{a_u, a_v\} \cup A_e \cup A_e^u \cup A_e^v \subseteq \chi'(t)$ and the property (P1) is satisfied. Furthermore, only the vertices in A_V are in more that one bag and for every $v \in V(G)$ it holds that $a_v \in \chi'(t)$ if and only if $v \in \chi(t)$. Since $\mathcal{T}$ is a tree-decomposition, the property (P2) holds as well. □

Note that in the proof of Theorem 9, the number of resource-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 (this can be easily shown by adding an independent set of size $2n$ to an instance of 3-COLORING, so any 3-coloring of the original graph can be extended to a coloring where every color set contains precisely n colors). Therefore, starting from an instance of EQUITABLE COLORING with 3 colors and using the reduction given in the proof of Theorem 9, we get the following theorem:

Corollary 10. GRAPH (PROPORTIONAL) ENVY-FREE ALLOCATION *is* NP*-hard even when restricted to instances with a constant number of resource-types, number of agent-types, and bundle-size.*

Our last lower-bound result is a counterpart to Corollary 10 showing that Theorem 1 cannot be strengthened by omitting the number of resource-types from the parameterization. Indeed, we even show that a structural restriction stronger than treewidth, namely parameterizing by the *vertex cover number* is insufficient to obtain tractability.

Theorem 11. GRAPH (PROPORTIONAL) ENVY-FREE ALLOCATION *is* NP*-hard even for treewidth and vertex cover number* 0 *(resp.* 1 *for* GEFA*) and one agent-type.*

Proof. The reduction is nearly identical to the one given in a previous work by Bliem, Bredereck and Niedermeier [6, Theorem 3]. Their result showed that a modification of GPEFA on complete graphs is W[1]-hard parameterized by the number of agents,

even when there is only one agent-type. We will now give the reduction and point out the differences. The reduction is from the following NP -hard problem.

UNARY BIN PACKING	
Input:	Positive integers $w_1, \ldots, w_m, b, C$ encoded in unary.
Question:	Is there an assignment of m resources with weights $w_1, \ldots, w_m$ to at most b bins such that none of the bins exceeds weight capacity C

As in the reduction by Bliem, Bredereck and Niedermeier [6], we use the set of resources $R = \{r_1, \ldots, r_m\} \cup \{d_1, \ldots, d_q\}$, where $q = b \cdot C - \sum_{1 \leq i \leq m} w_i$ is the total amount of "unused capacity" and the set of agents $A = \{a_1, \ldots, a_b\}$ such that each agent a_i has a preference function $\tau_{a_i}(d_j) = 1$, $j \in [q]$, and $\tau_{a_i}(r_j) = w_j$, $j \in [m]$. Our choice of the social network depends on whether we consider GPEFA or GEFA.

We start with the reduction for GPEFA. In this case we set the social network G to be edgeless.

The forward direction is exactly the same as in [6]. We assign the resources according to the bin packing and fill the assignment for each agent with enough resources in $D = \{d_1, \ldots, d_q\}$ such that $\tau_{a_i}(\pi(a_i)) = C$ for all $i \in [b]$. It is easy to see that the assignment is graph proportional envy-free.

For the backward direction, we will show that in any graph proportional envy-free allocation π it holds $\tau_{a_i}(\pi(a_i)) = C$. From completeness of π (i.e., because every resource is mapped to an agent) it is then straightforward to assign the resource j to bin i if and only if $r_j \in \pi(a_i)$. As the graph is edgeless it follows from proportionality for every agent a that $\tau_a(\pi(a)) \geq \frac{\sum_{r \in R} \tau_a(r)}{b}$. As the preference function is the same for all agents and since for each agent $a \in A$ we have $\sum_{r \in R} \tau_a(r) = b \cdot C$, this is possible only if all agents are assigned resources of total value C.

For GEFA we consider as social network G a star with center a_1 in which all edges are bidirected.

The forward direction is again exactly same as in the previous related reduction [6]. We assign the resources according to the bin packing and fill the assignment for each agent with enough resources in $D = \{d_1, \ldots, d_q\}$ such that $\tau_{a_i}(\pi(a_i)) = C$ for all $i \in [b]$. It is easy to see that the assignment is graph envy-free.

For the backward direction, note that for any $S \subseteq R$ and $i, j \in [b]$ it holds $\tau_{a_i}(S) = \tau_{a_j}(S)$. Therefore from graph envy-freeness it follows that a_1 values the assignments of all agents the same. From completeness of π it follows that every agent is assigned resources of total value C. □

5. 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 12. GRAPH (PROPORTIONAL) ENVY-FREE ALLOCATION *is in* XP *parameterized by clique-width of the social network, number of resource-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 GRAPH (PROPORTIONAL) ENVY-FREE ALLOCATION. It is known that an approximate k-expression tree can be computed in fixed-parameter time even for digraphs [34, 26, 22], and so it suffices to solve the problem when a k-expression tree for G is provided in the input.

Lemma 13. GRAPH (PROPORTIONAL) ENVY-FREE ALLOCATION *can be solved in time at most* $\mathcal{O}(|R|^{|T_R|(2k|T_A|+1)}k^2|A|)$ *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 GPEFA.

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

- $\alpha_{\min} : \Omega_t \times T_A \to \mathcal{B}$ is a function that for every $\omega \in \Omega_t$ and every $t_{\mathrm{a}} \in T_A$ provides the bundle function of a bundle with minimum value w.r.t. $\tau_{t_{\mathrm{a}}}$ 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 $t_{\mathrm{a}} \in T_A$ provides the bundle function of a bundle with maximum value w.r.t. $\tau_{t_{\mathrm{a}}}$ allocated to any agent with label ω in G_t.
- $\tilde{u} : \Omega_t \to \mathcal{B}$ is a function that for every label $\omega \in \Omega_t$ provides the bundle containing all resources 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 GEFA, it suffices to merely remember the bundle function of all resources assigned so far.
- (can be omitted for GEFA) β is a function that maps every tuple $(\omega \in \Omega_t, t_{\mathrm{a}} \in T_A)$ to $(a, \mathrm{bfun}, \mathrm{bfun}^+)$, where:
  - a is an agent of type t_{a} with label ω in G_t that maximizes:

$$\tau_{t_{\mathrm{a}}}(R) - \tau_{t_{\mathrm{a}}}(\pi(a))|A \setminus N_G^+(a)| - \tau_{t_{\mathrm{a}}}(\pi(N_G^+(a) \cap A_t))$$

    for an allocation π of G_t.

– bfun is equal to the bundle function of agent a, and
– bfun^+ is equal to the bundle function of the bundle containing all resources already assigned to all out-neighbors of a in G_t.

The semantics of a record are as follows. We say that a record $(\alpha_{\min}, \alpha_{\max}, \tilde{u}, \beta)$ 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 $t_\mathrm{a} \in T_A$, it holds that $\alpha_{\min}(\omega, t_\mathrm{a}) = \text{BFUN}(\pi(a))$, 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 $\omega \in \Omega_t$, $t_\mathrm{a} \in T_A$, it holds that $\alpha_{\max}(\omega, t_\mathrm{a}) = \text{BFUN}(\pi(a))$, 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 $\tilde{u}(\omega) = \text{BFUN}(\pi(A_t^\omega))$.

(R4) π is graph 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 GPEFA, and is also the most involved. For every $\omega \in \Omega_t$, $t_\mathrm{a} \in T_A$, it holds that $\beta(\omega, t_\mathrm{a}) = (a, \mathrm{bfun}_a, \mathrm{bfun}_a^+)$, where a is an agent of type t_a with label ω in G_t that maximizes:

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

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 any 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 $(a, \mathrm{bfun}_a, \mathrm{bfun}_a^+)$), 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 the root r of T satisfies the following: $\mathcal{R}(r)$ contains a record $(\alpha_{\min}, \alpha_{\max}, \tilde{u}, \beta)$ such that $\sum_{\omega \in \Omega_r} \tilde{u}(\omega) = \text{BFUN}(R)$ and, for the case of GPEFA, additionally:

$$\tau_{t_\mathrm{a}}(R) - \tau_{t_\mathrm{a}}(\mathrm{bfun}_a)|A \setminus N_G^+(a)| \leq \tau_{t_\mathrm{a}}(\mathrm{bfun}_a^+)$$

for every $\omega \in \Omega_r$ and every $t_\mathrm{a} \in T_A$, where $\beta(\omega, t_\mathrm{a}) = (a, \mathrm{bfun}_a, \mathrm{bfun}_a^+)$.

We note that, using the same standard techniques as for treewidth, one can always reconstruct an allocation from the records. Hence, to conclude the proof it suffices to compute $\mathcal{R}(t)$ via leaf-to-root dynamic programming. The following four lemmas show how this can be achieved for all of the four types of nodes in a k-expression tree.

Lemma 14. *Let $\ell \in V(T)$ be a leaf node of the form $\omega(a)$. Then $\mathcal{R}(\ell)$ can be computed in time $\mathcal{O}(|\mathcal{B}|)$.*

Proof. Let t_a be the agent-type of a. Then, for every bundle function (corresponding to an allocation of resources to a) $\mathrm{bfun}_a \in \mathcal{B}$, $\mathcal{R}(\ell)$ contains a record $(\alpha_{\min}, \alpha_{\max}, \tilde{u}, \beta)$ such that:

- $\alpha_{\min}(\omega, t_a) = \mathrm{bfun}_a$,
- $\alpha_{\max}(\omega, t_a) = \mathrm{bfun}_a$,
- $\tilde{u}(\omega) = \mathrm{bfun}_a$,
- $\beta(\omega, t_a) = (a, \mathrm{bfun}_a, \text{BFUN}(\emptyset))$.

The correctness follows immediately from the semantics of the records and the running-time is dominated by the number of possible choices for bfun_a, which is $|\mathcal{B}|$. □

Lemma 15. *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 $\mathcal{O}(|\mathcal{R}(t_1)||\mathcal{R}(t_2)|k|T_A||T_R|)$.*

Proof. We say that two records $(\alpha^1_{\min}, \alpha^1_{\max}, \tilde{u}^1, \beta^1) \in \mathcal{R}(t_1)$ and $(\alpha^2_{\min}, \alpha^2_{\max}, \tilde{u}^2, \beta^2) \in \mathcal{R}(t_2)$ are *compatible* if $\sum_{\omega \in \Omega_t} \tilde{u}^1(\omega) + \tilde{u}^2(\omega) \leq \text{BFUN}(R)$. Then, for every two compatible records $(\alpha^1_{\min}, \alpha^1_{\max}, \tilde{u}^1, \beta^1) \in \mathcal{R}(t_1)$ and $(\alpha^2_{\min}, \alpha^2_{\max}, \tilde{u}^2, \beta^2) \in \mathcal{R}(t_2)$, $\mathcal{R}(t)$ contains the record $(\alpha_{\min}, \alpha_{\max}, \tilde{u}, \beta)$ such that:

- For every $\omega \in \Omega_t$ and $t_\mathrm{a} \in T_A$ either: $\alpha_{\min}(\omega, t_\mathrm{a}) = \alpha^1_{\min}(\omega, t_\mathrm{a})$, if $\tau_{t_\mathrm{a}}(\alpha^1_{\min}(\omega, t_\mathrm{a})) \leq \tau_{t_\mathrm{a}}(\alpha^2_{\min}(\omega, t_\mathrm{a}))$, or $\alpha_{\min}(\omega, t_\mathrm{a}) = \alpha^2_{\min}(\omega, t_\mathrm{a})$, otherwise. Note that if the functions $\alpha^i_{\min}$ are undefined for particular values of ω and t_a, we assume that their value is equal to $\text{BFUN}(R)$.
- For every $\omega \in \Omega_t$ and $t_\mathrm{a} \in T_A$ either: $\alpha_{\max}(\omega, t_\mathrm{a}) = \alpha^1_{\max}(\omega, t_\mathrm{a})$, if $\tau_{t_\mathrm{a}}(\alpha^1_{\max}(\omega, t_\mathrm{a})) \geq \tau_{t_\mathrm{a}}(\alpha^2_{\max}(\omega, t_\mathrm{a}))$, or $\alpha_{\max}(\omega, t_\mathrm{a}) = \alpha^2_{\max}(\omega, t_\mathrm{a})$, otherwise. Note that if the functions $\alpha^i_{\max}$ are undefined for particular values of ω and t_a, we assume that their value is equal to $\text{BFUN}(\emptyset)$.
- $\tilde{u}(\omega) = \tilde{u}^1(\omega) + \tilde{u}^2(\omega)$ for every $\omega \in \Omega_t$,
- For every $\omega \in \Omega_t$ and $t_\mathrm{a} \in T_A$, we distinguish two cases: We set $\beta(\omega, t_\mathrm{a}) = \beta^1(\omega, t_\mathrm{a})$, if

$$\begin{aligned}\tau_{t_\mathrm{a}}(R) - \tau_{t_\mathrm{a}}(\mathrm{bfun}_{a^1})|A \setminus N^+_G(a^1)| - \tau_{t_\mathrm{a}}(\mathrm{bfun}^+_{a^1}) \geq \\ \tau_{t_\mathrm{a}}(R) - \tau_{t_\mathrm{a}}(\mathrm{bfun}_{a^2})|A \setminus N^+_G(a^2)| - \tau_{t_\mathrm{a}}(\mathrm{bfun}^+_{a^2})\end{aligned}$$

  where $\beta^1 = (a^1, \mathrm{bfun}_{a^1}, \mathrm{bfun}^+_{a^1})$ and $\beta^2 = (a^2, \mathrm{bfun}_{a^2}, \mathrm{bfun}^+_{a^2})$. Otherwise, we set $\beta(\omega, t_\mathrm{a}) = \beta^2(\omega, t_\mathrm{a})$.

The correctness of the construction is rather obvious because G_t is obtained as the disjoint union of G_{t_1} and G_{t_2} and therefore there are no interactions between agents in the two parts. For instance, the definition of $\alpha_{\min}(\omega, t_a)$ for every $\omega \in \Omega_t$ and $t_a \in T_A$ follows because the agent a in A_t of type t_a and label ω that minimizes $\tau_a(\pi(a))$ is either in A_{t_1}, in which case we set $\alpha_{\min}(\omega, t_a) = \alpha^1_{\min}(\omega, t_a)$, or in A_{t_2}, in which case we set $\alpha_{\min}(\omega, t_a) = \alpha^2_{\min}(\omega, t_a)$.

Finally, the running-time follows because there are at most $|\mathcal{R}(t_1)||\mathcal{R}(t_2)|$ pairs of such records and for every such pair the time required is at most $\mathcal{O}(k|T_A||T_R|)$. □

Lemma 16. *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 $\mathcal{O}(|\mathcal{R}(t')|k|T_A||T_R|)$.*

Proof. For every record $(\alpha'_{\min}, \alpha'_{\max}, \tilde{u}', \beta') \in \mathcal{R}(t')$, $\mathcal{R}(t)$ contains the record $(\alpha_{\min}, \alpha_{\max}, \tilde{u}, \beta)$ such that:

- For every $\omega \in \Omega_t$ and $t_a \in T_A$ either:
  - $\alpha_{\min}(\omega, t_a) = \alpha'_{\min}(\omega, t_a)$, if $\omega \notin \{i, j\}$,
  - $\alpha_{\min}(\omega, t_a) = \text{BFUN}(R)$, if $\omega = i$, or
  - if $\omega = j$, then either $\alpha_{\min}(\omega, t_a) = \alpha'_{\min}(j, t_a)$, if $\tau_{t_a}(\alpha'_{\min}(i, t_a)) \geq \tau_{t_a}(\alpha'_{\min}(j, t_a))$, or $\alpha_{\min}(\omega, t_a) = \alpha'_{\min}(i, t_a)$, otherwise.

  Note that if the function $\alpha'_{\min}$ is undefined for particular values of ω (only possible for $\omega = j$), we assume that its value is equal to $\text{BFUN}(R)$.

- For every $\omega \in \Omega_t$ and $t_a \in T_A$ either:
  - $\alpha_{\max}(\omega, t_a) = \alpha'_{\max}(\omega, t_a)$, if $\omega \notin \{i, j\}$,
  - $\alpha_{\max}(\omega, t_a) = \text{BFUN}(\emptyset)$, if $\omega = i$, or
  - if $\omega = j$, then either $\alpha_{\max}(\omega, t_a) = \alpha'_{\max}(j, t_a)$, if $\tau_{t_a}(\alpha'_{\min}(i, t_a)) \leq \tau_{t_a}(\alpha'_{\min}(j, t_a))$, or $\alpha_{\max}(\omega, t_a) = \alpha'_{\max}(i, t_a)$, otherwise. Note that if the function $\alpha'_{\max}$ is undefined for particular values of ω (only possible for $\omega = j$), we assume that its value is equal to $\text{BFUN}(\emptyset)$.

- $\tilde{u}(\omega) = \tilde{u}'(\omega)$ for every $\omega \in \Omega_t$,

- For every $\omega \in \Omega_t$ and $t_a \in T_A$ either:
  - $\beta(\omega, t_a) = \beta'(\omega, t_a)$, if $\omega \notin \{i, j\}$,
  - $\beta(\omega, t_a) = \text{undef}$, if $\omega = i$, or
  - if $\omega = j$, then we distinguish two cases:
    We set $\beta(\omega, t_a) = \beta'(i, t_a)$, if

$$\tau_{t_a}(R) - \tau_{t_a}(\text{bfun}_{a^1})|A \setminus N^+_G(a^1)| - \tau_{t_a}(\text{bfun}^+_{a^1}) \geq \tau_{t_a}(R) - \tau_{t_a}(\text{bfun}_{a^2})|A \setminus N^+_G(a^2)| - \tau_{t_a}(\text{bfun}^+_{a^2})$$

    where $\beta(i, t_a) = (a^1, \text{bfun}_{a^1}, \text{bfun}^+_{a^1})$ and $\beta(j, t_a) = (a^2, \text{bfun}_{a^2}, \text{bfun}^+_{a^2})$. Otherwise, we set $\beta(\omega, t_a) = \beta'(j, t_a)$.

The correctness of the construction follows using very similar arguments as for the disjoint union node in Lemma 15. In essence, relabeling the label i to j means that A_t^j is obtained as the disjoint union of $A_{t'}^i$ and A_t^j. Furthermore, it means that A_t^i is empty and that nothing changes for the agents with any label other than i or j. Therefore, $\alpha_{\min}(\omega, t_a)$, $\alpha_{\max}(\omega, t_a)$, and $\beta(\omega, t_a)$ remain unchanged whenever $\omega \notin \{i, j\}$. Moreover, if $\omega = i$, then $\alpha_{\min}(\omega, t_a)$, $\alpha_{\max}(\omega, t_a)$, and $\beta(\omega, t_a)$ are reset to their respective undefined value, i.e., BFUN(R), BFUN($\emptyset$), or undef, respectively. Finally, if $\omega = j$ then $\alpha_{\min}(\omega, t_a)$, $\alpha_{\max}(\omega, t_a)$, and $\beta(\omega, t_a)$ are obtained as in Lemma 15 by considering the disjoint union of the labels i and j. For instance, the definition of $\alpha_{\min}(j, t_a)$ for every $t_a \in T_A$ follows because the agent a in A_t^j of type t_a that minimizes $\tau_a(\pi(a))$ is either in $A_{t'}^i$, in which case we set $\alpha_{\min}(j, t_a) = \alpha'_{\min}(i, t_a)$, or in $A_{t'}^j$, in which case we set $\alpha_{\min}(j, t_a) = \alpha'_{\min}(j, t_a)$.

Finally, the running-time follows because there are at most $|\mathcal{R}(t')|$ records in $\mathcal{R}(t')$ and for every such record the time required is at most $\mathcal{O}(k|T_A||T_R|)$. □

Lemma 17. *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 $\mathcal{O}(|\mathcal{R}(t')|k|T_A||T_R|)$.*

Proof. Consider a record $R = (\alpha'_{\min}, \alpha'_{\max}, \tilde{u}', \beta') \in \mathcal{R}(t')$. We say that R is *extendable* if:

$$\tau_{t_a}(\alpha'_{\min}(i, t_a)) \geq \tau_{t_a}(\alpha'_{\max}(j, t_a))$$

for every $t_a \in T_A$. Informally, R is extendable if no agent of label i envies an agent with label j, which implies that R still satisfies condition (R4) for a valid record after all edges from agents with label i to agents with label j are added.

Then, for every extendable record $R' = (\alpha'_{\min}, \alpha'_{\max}, \tilde{u}', \beta') \in \mathcal{R}(t')$, $\mathcal{R}(t)$ contains the record $R = (\alpha_{\min}, \alpha_{\max}, \tilde{u}, \beta)$ such that:

- $\alpha_{\min} = \alpha'_{\min}$, $\alpha_{\max} = \alpha'_{\max}$, $\tilde{u} = \tilde{u}'$,
- 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 $\beta(\omega, t_a) = (a, \text{bfun}_a, \text{bfun}_a^+ + \tilde{u}'(j))$, where $\beta'(\omega, t_a) = (a, \text{bfun}_a, \text{bfun}_a^+)$, otherwise.

The correctness of the construction can be seen as follows. First the condition that R' is extendable ensures that R satisfies condition (R4). Moreover, since no new agents are added and we do not change the bundles (or equivalently bundle functions) for any agents, it holds that $\alpha_{\min} = \alpha'_{\min}$, $\alpha_{\max} = \alpha'_{\max}$, and $\tilde{u} = \tilde{u}'$. Moreover, $\beta(\omega, t_a) = \beta'(\omega, t_a)$ for every label $\omega \in \Omega_t \setminus \{i\}$ and $t_a \in T_A$ because nothing changes for the corresponding agents. Finally, if $\omega = i$ we need to update $\beta(\omega, t_a)$ to take into account that all agents with label j are now additional out-neighbors of all agents with label i. That is, we need to add the bundle function $\tilde{u}'(j)$ for the bundle of resources assigned to all agents of label j to the bundle function bfun_a^+ for the bundle of resources assigned to the out-neighbors of agent a, where $\beta'(\omega, t_a) = (a, \text{bfun}_a, \text{bfun}_a^+)$.

Finally, the running-time follows because there are at most $|\mathcal{R}(t')|$ records in $\mathcal{R}(t')$ and for every such record the time required is at most $\mathcal{O}(|T_A||T_R|)$. □

We are now ready to prove our main theorem.

Proof of Lemma 13. 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 $\mathcal{R}(r) \neq \emptyset$. Note that the correctness of the algorithm follows from Lemmas 14, 15, 16, and 17. The running-time of the algorithm is at most the number of nodes of T, i.e., at most $k^2|A|$, times the maximum time required to compute $\mathcal{R}(t)$ for any of the four node types of a k-expression tree, which because of lemmas 14, 15, 16, and 17 is at most $\mathcal{O}(|\mathcal{R}(t)|^2 k|T_A||T_R|)$. Because $|\mathcal{R}(t)| \leq |R|^{|T_R|(4|T_A|+1)}$, we obtain an upper bound of $\mathcal{O}(|R|^{|T_R|(2k|T_A|+1)}(k^2|A|)$ for the total running-time of the algorithm. □

From the running time of Theorem 12 and the fact that $|T_R| \leq |R|$, we also obtain an FPT result when we parameterize by the number of resources instead of the number of resource-types.

Corollary 18. GRAPH (PROPORTIONAL) ENVY-FREE ALLOCATION *is in* FPT *parameterized by clique-width of the social network, number of resources, and number of agent-types.*

Towards Fixed Parameter Tractability. Theorem 9 excludes the existence of a fixed-parameter algorithm for GRAPH (PROPORTIONAL) 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 [21], 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 GRAPH (PROPORTIONAL) ENVY-FREE ALLOCATION. Recall that, based on Theorem 11, it is not possible to achieve tractability by restricting the structure of the graph alone, at least not in any natural way—and to achieve our result, we parameterize by the number of resource-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 9.

Theorem 19. GRAPH (PROPORTIONAL) ENVY-FREE ALLOCATION *is in* FPT *parameterized by vertex cover number of the social network, number of resource-types, and bundle-size.*

Proof. Let $\mathcal{I} = (A, R, (\tau_a)_{a \in A}, G)$ be an instance of GRAPH (PROPORTIONAL) ENVY-FREE ALLOCATION. It is well known that we can compute a minimum-size vertex cover $X \subseteq A$ of G, i.e., $|X| = \text{vcn}(G)$, in FPT time w.r.t. the size of the vertex cover [16]. 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 resource-types) of at most k resources 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 GRAPH (PROPORTIONAL) ENVY-FREE ALLOCATION for $\mathcal{I}$.

Because for any $a \in A \setminus X$, $N_G^+(a) \subseteq X$, graph envy-freeness at a only speaks about a and its preference of resources assigned to vertices in X. These are already

fixed by π. We compute (up to resource-type) for each $a \in A \setminus X$ all bundles R_a of at most k elements such that $\sum_{t_r \in T_R} \tau_a(t_r)\text{BFUN}(R_a)[t_r] \geq \tau_a(\pi(x))$ for all $x \in N_G^+(a)$ in time $|T_R|^k \cdot |X|$. Denote these bundles by $\mathfrak{B}_a$. By construction, the conditions for graph envy-freeness will be satisfied by π for $a \in A \setminus X$, if and only if $\pi(a) \in \mathfrak{B}_a$.
For each $a \in X$ we can check graph 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.) Moreover for $a \in A \setminus X$ we can remove bundles R_a from $\mathfrak{B}_a$ whenever $\tau_a(R_a) < \frac{\tau_a(R \setminus \pi(N_G^+(a)))}{|A \setminus N_G^+(a)|}$. In this way we ensure that the graph envy-freeness condition will be satisfied by π for all of A, if and only if for all $a \in A \setminus X$, $\pi(a) \in \mathfrak{B}_a$.

If required, in a second similar step, we restrict the $\mathfrak{B}_a$ to ensure proportionality: For $a \in A \setminus X$, $N_G^+(a) \subseteq X$ and thus $\pi(N_G^+(a))$ is already defined by π. We restrict $\mathfrak{B}_a$ to all bundles $R' \in \mathfrak{B}_a$ with $\tau_a(R') \geq \frac{\tau_a(R \setminus \pi(N_G^+(a)))}{|A \setminus N_G^+(a)|}$ in time $|T_R|^k \cdot |X|$. By construction, the conditions for graph proportional envy-freeness will be satisfied by π for $a \in A \setminus X$, if and only if $\pi(a) \in \mathfrak{B}_a$.
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 $R' \in \mathfrak{B}_a$ 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) GRAPH PROPORTIONAL ENVY-FREE ALLOCATION.

We do so by considering an integer linear program with a number of variables that we can bound in terms of vcn(G), $|T_R|$ and k. For this we group the agents $a \in A \setminus X$ according to their respective $\mathfrak{B}_a$. That is, we say $a, a' \in A \setminus X$ are in the same *group* if $\mathfrak{B}_a = \mathfrak{B}_{a'}$. Because there are (up to resource-type) at most $|T_R|^k$ bundles of k resources and each $\mathfrak{B}_a$ is a set of such bundles, the number of groups can be bounded by $2^{|T_R| \cdot k}$. Let $\mathcal{G}_1, \ldots \mathcal{G}_z$ with $z \leq 2^{|T_R| \cdot k}$ be an enumeration of all the groups of agents in $A \setminus X$. We denote by $\mathfrak{B}_\mathcal{G}$ the union $\bigcup_{a \in \mathcal{G}} \mathfrak{B}_a$. Now for each group $\mathcal{G}$, each of its bundles $R_G \in \mathfrak{B}_\mathcal{G}$ 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 equivalent resource-types. On these variables we solve the integer linear program with the following constraints:

Integrality constraints:

For $\mathcal{G} \in \{\mathcal{G}_1, \ldots, \mathcal{G}_z\}$, $R_\mathcal{G} \in \mathfrak{B}_\mathcal{G}$ and $X' \subseteq X$, $x_{\mathcal{G},R,X'} \in \mathbb{N}_0$;
These integrality constraints ensure that the number of agents in each group are assigned a specific bundle are natural numbers.

Network conformity constraints:

For $X' \subseteq X$ and $\mathcal{G} \in \{\mathcal{G}_1, \ldots, \mathcal{G}_z\}$, $\sum_{R_\mathcal{G} \in \mathfrak{B}_\mathcal{G}} x_{\mathcal{G},R_\mathcal{G},X'} = |\{a \in A \mid N_G^-(a) = X'\} \cap \mathcal{G}|$;
These network conformity constraints do not ensure any property of an allocation corresponding to the solution of the ILP, but rather ensure that the variables of the ILP encode the desired information. Recall that we want the variable $x_{\mathcal{G},R_\mathcal{G},X'}$ to encode how many agents $a \in \mathcal{G}$ in group $\mathcal{G}$ with $N_G^-(a) = X'$ will be assigned bundle $R_\mathcal{G}$. This means we need to ensure that each agent $a \in \mathcal{G}$ in group $\mathcal{G}$ with $N_G^-(a) = X'$ is counted exactly once in $\sum_{R_\mathcal{G} \in \mathfrak{B}_\mathcal{G}} x_{\mathcal{G},R_\mathcal{G},X'}$. The number of such agents is obviously $|\{a \in A \mid N_G^-(a) = X'\} \cap \mathcal{G}|$.

Resource constraints:
For $t_{\mathrm{r}} \in T_R$, $\sum_{X' \subseteq X} \sum_{\mathcal{G} \in \{\mathcal{G}_1,\ldots,\mathcal{G}_z\}} \sum_{R_\mathcal{G} \in \mathfrak{B}_\mathcal{G}} \mathrm{BFUN}(R_\mathcal{G})[t_{\mathrm{r}}] \cdot x_{\mathcal{G},R_\mathcal{G},X'} + \mathrm{BFUN}(\pi(X))[t_{\mathrm{r}}] = \mathrm{BFUN}(R)[t_{\mathrm{r}}]$;
These resource constraints ensure that all resources are assigned to agents and also that the given resources of each resource-type are not exceeded.

Proportionality constraints:
For $a \in X$, $\sum_{\mathcal{G} \in \{\mathcal{G}_1,\ldots,\mathcal{G}_z\}} \sum_{R_\mathcal{G} \in \mathfrak{B}_\mathcal{G}} \sum_{X' \subseteq X \setminus \{a\}} \tau_a(R_\mathcal{G}) \cdot x_{\mathcal{G},R_\mathcal{G},X'} \leq |A \setminus N_G^+(a)| \cdot \tau_a(\pi(a)) - \sum_{a' \in X \setminus N_G^+(a)} \tau_a(\pi(a'))$.
These proportionality constraints ensure that a solution of the ILP corresponds to a proportional allocation.

A graph proportional envy-free allocation π^* for $\mathcal{I}$ infers a solution to this ILP in the branch where π coincides with π^* restricted to X: This can be seen by setting $x_{\mathcal{G},R_\mathcal{G},X'} = |\{a \in \mathcal{G} \mid N_G^-(a) = X' \wedge \mathrm{BFUN}(\pi^*(a)) = \mathrm{BFUN}(R_\mathcal{G})\}|$ and verifying the constraints of the ILP.

Conversely, given a solution to the ILP, i.e., the $x_{\mathcal{G},R_\mathcal{G},X'}$ are assigned values that satisfy all constraints, the network conformity constraints guarantee that we find a bijection φ between A and $\{(\mathcal{G}, R_\mathcal{G}, X', \ell) \mid \mathcal{G} \in \{\mathcal{G}_1,\ldots,\mathcal{G}_z\} \wedge R_\mathcal{G} \in \mathfrak{B}_\mathcal{G} \wedge X' \subseteq X \wedge \ell \in [x_{\mathcal{G},R_\mathcal{G},X'}]\}$ such that φ maps an agent a to a tuple with this agent's group at the first and $N_G^-(a)$ at the third component of $\varphi(a)$. In turn the resource constraints guarantee that we find disjoint bundles that realize for all groups $\mathcal{G}$ of agents and all $X' \subset X$, $x_{\mathcal{G},R_\mathcal{G},X'}$-many bundles that are equivalent to $R_\mathcal{G}$ up to resource-types. In combination this implies we find disjoint bundles for each agent that conform to the group of that agent. By previous considerations this implies a graph envy-free allocation, that is also proportional for $A \setminus X$. It is straightforward to check that the proportionality constraints encode exactly proportionality on X in our ILP.

Lenstra's celebrated result [28] states that an ILP can be solved in FPT time parameterized by the number of its variables which is in our case $2^{|T_R| \cdot k} |T_R|^k \cdot 2^{\mathrm{vcn}(G)}$. This concludes the proof. □

6. 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 constant treewidth or clique-width in combination with a constant number of resource-types or a constant number of resource- and agent-types respectively. In this sense this work shows the potential of exploiting structural properties of the social network for resource allocation with social networks, for example in some of the situations presented in Subsection 2.5.

On the other hand, if one wishes to lift these algorithms to fixed-parameter ones, our results rule this out for any parameterization by a combination of our main parameters whenever both $|A|$ and $|R|$ are large. This situation is likely to happen for resource allocation problems that may arise in, e.g., food banks [38]. In this way our work

points to the necessity of identifying additional parameters that allow us to tackle resource allocation in these situations. The fixed-parameter algorithm from Theorem 19 which additionally uses bundle-size can be understood as a very preliminary step in this direction. More work in this direction is desirable, but is likely to depend more strongly on the context of and restrictions given in specific applications of GPEFA and GEFA.

Of course, another immediate direction for theoretical future work would target the gaps in the parameterized complexity classification of GPEFA and GEFA that are left open by our work (see Table 1 for an overview with respect to $\mathrm{tw}(G)$, $\mathrm{cw}(G)$, $|A|$, $|T_A|$, $|R|$ and $|T_R|$). Here, a question we find particularly interesting is the following:
Q1: Are there XP algorithms for GPEFA and GEFA parameterized by $|T_R| + \mathrm{cw}(G)$?

It would also be natural to study classic resource allocation problems without social networks parameterized by the number of resource-types and agent-types. A question which we would like to emphasize here is:
Q2: Are the envy-free/proportional versions of resource allocation without a social network FPT when parameterized by the number of resource-types?
Note that this is the same as asking for a fixed-parameter algorithm for GEFA on cliques or GPEFA on edgeless graphs parameterized by the number of resource-types. Hence **Q2** might be a first step toward resolving **Q1**. This is closely connected to similar questions about BIN PACKING. Specifically, when in **Q2** we restrict ourselves to instances with one agent type, we essentially arrive at the question whether BIN PACKING[3] is in FPT parameterized by the number of different values of integers in the input. This problem is known to be in XP [24], but the existence of a fixed-parameter algorithm remains an open question. Note also that if the number of agents is considered as an additional parameter, then the problem becomes FPT [12].

Acknowledgments. Robert Ganian and Thekla Hamm acknowledge support from the Austrian Science Fund (FWF, Projects P31336: **NFPC** and Y1329: **ParAI**). Thekla Hamm is also supported by FWF project W1255-N23. Sebastian Ordyniak acknowledges support from the Engineering and Physical Sciences Research Council (EPSRC, project EP/V00252X/1).

References

[1] Rediet Abebe, Jon M. Kleinberg, and David C. Parkes. Fair division via social comparison. In Kate Larson, Michael Winikoff, Sanmay Das, and Edmund H. Durfee, editors, *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2017, São Paulo, Brazil, May 8-12, 2017*, pages 281–289. ACM, 2017.

[2] Haris Aziz, Sylvain Bouveret, Ioannis Caragiannis, Ira Giagkousi, and Jérôme Lang. Knowledge, fairness, and social constraints. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on*

[3]The integers are given by the valuations of every agent for each resource, the number of bins is simply the number of agents and the capacity is the valuation of all R divided by the number of agents.

Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018, pages 4638–4645. AAAI Press, 2018.

[3] Xiaohui Bei, Youming Qiao, and Shengyu Zhang. Networked fairness in cake cutting. In Carles Sierra, editor, *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 3632–3638. ijcai.org, 2017.

[4] Xiaohui Bei, Xiaoming Sun, Hao Wu, Jialin Zhang, Zhijie Zhang, and Wei Zi. Cake cutting on graphs: A discrete and bounded proportional protocol. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 2114–2123. SIAM, 2020.

[5] Aurélie Beynier, Yann Chevaleyre, Laurent Gourvès, Ararat Harutyunyan, Julien Lesca, Nicolas Maudet, and Anaëlle Wilczynski. Local envy-freeness in house allocation problems. *Autonomous Agents and Multi-Agent Systems*, 33(5):591–627, 2019.

[6] Bernhard Bliem, Robert Bredereck, and Rolf Niedermeier. Complexity of efficient and envy-free resource allocation: Few agents, resources, or utility levels. In Subbarao Kambhampati, editor, *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 102–108. IJCAI/AAAI Press, 2016.

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

[8] Hans L. Bodlaender, Pål Grønås Drange, Markus S. Dregi, Fedor V. Fomin, Daniel Lokshtanov, and Michal Pilipczuk. A $c^k n$ 5-approximation algorithm for treewidth. *SIAM J. Comput.*, 45(2):317–378, 2016.

[9] Sylvain Bouveret, Yann Chevaleyre, and Nicolas Maudet. Fair allocation of indivisible goods. In *Handbook of Computational Social Choice*, pages 284–310. 2016.

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

[11] Simina Brânzei, Yuezhou Lv, and Ruta Mehta. To give or not to give: Fair division for single minded valuations. In Subbarao Kambhampati, editor, *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 123–129. IJCAI/AAAI Press, 2016.

[12] Robert Bredereck, Andrzej Kaczmarczyk, Dusan Knop, and Rolf Niedermeier. High-multiplicity fair allocation: Lenstra empowered by n-fold integer programming. In Anna Karlin, Nicole Immorlica, and Ramesh Johari, editors, *Proceedings of the 2019 ACM Conference on Economics and Computation, EC 2019, Phoenix, AZ, USA, June 24-28, 2019*, pages 505–523. ACM, 2019.

[13] Robert Bredereck, Andrzej Kaczmarczyk, and Rolf Niedermeier. Envy-free allocations respecting social networks. *Artif. Intell.*, 305:103664, 2022.

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

[15] Bruno Courcelle and Stephan Olariu. Upper bounds to the clique width of graphs. *Discrete Applied Mathematics*, 101(1-3):77–114, 2000.

[16] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

[17] Erik D. Demaine, Felix Reidl, Peter Rossmanith, Fernando Sánchez Villaamil, Somnath Sikdar, and Blair D. Sullivan. Structural sparsity of complex networks: Bounded expansion in random models and real-world graphs. *J. Comput. Syst. Sci.*, 105:199–241, 2019.

[18] Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

[19] Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer Verlag, 2013.

[20] Michael R. Fellows, Fedor V. Fomin, Daniel Lokshtanov, Frances Rosamond, Saket Saurabh, Stefan Szeider, and Carsten Thomassen. On the complexity of some colorful problems parameterized by treewidth. *Information and Computation*, 209(2):143–153, 2011.

[21] Michael R. Fellows, Daniel Lokshtanov, Neeldhara Misra, Frances A. Rosamond, and Saket Saurabh. Graph layout problems parameterized by vertex cover. In Seok-Hee Hong, Hiroshi Nagamochi, and Takuro Fukunaga, editors, *Algorithms and Computation, 19th International Symposium, ISAAC 2008, Gold Coast, Australia, December 15-17, 2008. Proceedings*, volume 5369 of *Lecture Notes in Computer Science*, pages 294–305. Springer, 2008.

[22] Robert Ganian, Petr Hlinený, and Jan Obdrzálek. Better algorithms for satisfiability problems for formulas of bounded rank-width. *Fundam. Inform.*, 123(1):59–76, 2013.

[23] Robert Ganian, Sebastian Ordyniak, and C. S. Rahul. Group activity selection with few agent types. In Michael A. Bender, Ola Svensson, and Grzegorz Herman, editors, *27th Annual European Symposium on Algorithms, ESA 2019,*

September 9-11, 2019, Munich/Garching, Germany, volume 144 of *LIPIcs*, pages 48:1–48:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[24] Michel X. Goemans and Thomas Rothvoss. Polynomiality for bin packing with a constant number of item types. *J. ACM*, 67(6):38:1–38:21, 2020.

[25] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001.

[26] Mamadou Moustapha Kanté and Michaël Rao. The rank-width of edge-coloured graphs. *Theory Comput. Syst.*, 52(4):599–644, 2013.

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

[28] Hendrik W. Jr. Lenstra. Integer programming with a fixed number of variables. *Math. Oper. Res.*, 8(4):538–548, 1983.

[29] R. J. Lipton, E. Markakis, E. Mossel, and A. Saberi. On approximately fair allocations of indivisible goods. In *Proceedings of the 5th ACM Conference on Electronic Commerce*, EC '04, page 125?131, New York, NY, USA, 2004. Association for Computing Machinery.

[30] Dániel Marx. Can you beat treewidth? *Theory of Computing*, 6:85–112, 2010.

[31] Jaroslav Nešetřil and Patrice Ossona de Mendez. *Sparsity - Graphs, Structures, and Algorithms*, volume 28 of *Algorithms and Combinatorics*. Springer, 2012.

[32] Trung Thanh Nguyen and Jörg Rothe. Approximate pareto set for fair and efficient allocation: Few agent types or few resource types. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 290–296. ijcai.org, 2020.

[33] Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford Lecture Series in Mathematics and its Applications. Oxford University Press, Oxford, 2006.

[34] Sang-il Oum and Paul D. Seymour. Approximating clique-width and branch-width. *J. Comb. Theory, Ser. B*, 96(4):514–528, 2006.

[35] Neil Robertson and Paul D. Seymour. Graph minors. I. Excluding a forest. *J. Comb. Theory, Ser. B*, 35(1):39–61, 1983.

[36] Tammar Shrot, Yonatan Aumann, and Sarit Kraus. On agent types in coalition formation problems. In Wiebe van der Hoek, Gal A. Kaminka, Yves Lespérance, Michael Luck, and Sandip Sen, editors, *9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010), Toronto, Canada, May 10-14, 2010, Volume 1-3*, pages 757–764. IFAAMAS, 2010.

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

[38] Toby Walsh. Allocation in practice. In Carsten Lutz and Michael Thielscher, editors, *KI 2014: Advances in Artificial Intelligence - 37th Annual German Conference on AI, Stuttgart, Germany, September 22-26, 2014. Proceedings*, volume 8736 of *Lecture Notes in Computer Science*, pages 13–24. Springer, 2014.