Almost Envy-Free allocations in multigraphs

Minas Marios Sotiriou
7115142300016

Examination committee:
Alkmini Sgouritsa, Department of Informatics, Athens University of Economics and Business.
Evangelos Markakis, Department of Informatics, Athens University of Economics and Business.
Giorgos Christodoulou, School of Informatics, Aristotle University of Thessaloniki.
Aris Pagourtzis, School of Electrical and Computer Engineering, National Technical University of Athens.

Supervisor:
Alkmini Sgouritsa, Assistant Professor, Department of Informatics, Athens University of Economics and Business.
Co-supervisor:
Evangelos Markakis, Professor, Department of Informatics, Athens University of Economics and Business.

ABSTRACT

This thesis studies the problem of "fairly" dividing goods in several agents. The study of the problem started in the late 40's for divisible goods with the "cake cutting" problem. There are multiple notions of fairness when referring to the field of Fair Division. One notion is *envy-free allocations*, where nobody envies what is allocated to any other agent. Envy-freeness has been extensively studied in settings with divisible resources. This thesis focuses on settings with indivisible resources, where envy-freeness cannot be guaranteed. Instead, our focus revolves around a fairness criterion approximating envy-free allocations: that is envy-free up to any good (EFX) allocations, i.e., no agent envies any proper subset of the goods given to any other agent. The existence or not of EFX allocations is a major open problem in Fair Division, and there are only positive results for special cases.

Christodoulou et al. 2023 introduced a setting with a restriction on the agents' valuations according to a graph structure: the vertices correspond to agents and the edges to goods, and each vertex/agent has zero marginal value (or in other words, they are indifferent) for the edges/goods that are not adjacent to them. The existence of EFX allocations has been shown for simple graphs when agents have general monotone valuations by Christodoulou et al. 2023, and for multigraphs for restricted additive valuations by Kaviani et al. 2024.

In this thesis, we push the state-of-the-art further, and show that the EFX allocations always exists in *multigraphs* and for *general monotone valuations* if any of the following three conditions hold: either (a) the multigraph is bipartite, or (b) each agent has at most $\lceil \frac{n}{4} \rceil - 1$ neighbors, where n is the total number of agents, or (c) the shortest cycle with non-parallel edges has length at least 6.

ΣΥΝΟΨΗ

Η διπλωματική μελετά το πρόβλημα της «δίκαιης» κατανομής αγαθών σε πράκτορες. Η μελέτη του προβλήματος ξεκίνησε στα τέλη της δεκαετίας του 1940 για αγαθά που μπορούν να διαχωριστούν με το πρόβλημα «cake cutting». Υπάρχουν πολλές έννοιες δίκαιου διαμοιρασμού αγαθών όταν αναφερόμαστε στον τομέα της Δίκαιης Κατανομής (Fair Division). Μία έννοια είναι οι κατανομές χωρίς φθόνο (envy-free allocations), όπου κανείς δεν ζηλεύει ό,τι έχει κατανεμηθεί σε οποιονδήποτε άλλο πράκτορα. Η απουσία φθόνου έχει μελετηθεί εκτενώς σε περιβάλλοντα με πόρους που μπορούν να διαχωριστούν. Η διπλωματική επικεντρώνεται σε διαμοιρασμούς αγαθών που δεν μπορούν να διαχωριστούν, όπου η απουσία φθόνου δεν μπορεί να εγγυηθεί. Αντίθετα, εστιάζουμε γύρω από ένα κριτήριο δικαιοσύνης που προσεγγίζει τις κατανομές χωρίς φθόνο: τους διαμοιρασμούς envy-free up to any good (EFX), δηλαδή, κανένας πράκτορας δεν ζηλεύει κανένα γνήσιο υποσύνολο των αγαθών που δίνονται σε οποιονδήποτε άλλο πράκτορα. Η ύπαρξη ή μη των διαμοιρασμών EFX αποτελεί ένα σημαντικό ανοικτό πρόβλημα στη Δίκαιη Κατανομή, και υπάρχουν μόνο θετικά αποτελέσματα για ειδικές συνθήκες.

Οι Christodoulou et al. 2023 εισήγαγαν μία συνθήκη με περιορισμό στις αποτιμήσεις των πρακτόρων σύμφωνα με μια δομή γραφήματος: οι κορυφές αντιστοιχούν σε πράκτορες και οι ακμές σε αγαθά, και κάθε κορυφή/πράκτορας έχει μηδενική αξία (ή αλλιώς είναι αδιάφορος) για τις ακμές/αγαθά που δεν είναι γειτονικές με αυτήν. Η ύπαρξη των κατανομών EFX έχει αποδειχθεί για γραφήματα όταν οι πράκτορες έχουν γενικές μονότονες αποτιμήσεις από τους Christodoulou et al. 2023 και για πολυγραφήματα για περιορισμένες προσθετικές αποτιμήσεις από τους Kaviani et al. 2024.

Στην παρούσα διατριβή, δείχνουμε ότι οι κατανομές EFX υπάρχουν πάντα σε πολυγραφήματα και για γενικές μονότονες αποτιμήσεις αν ισχύει τουλάχιστον μία από τις παρακάτω τρεις συνθήκες: είτε (α) το πολυγράφημα είναι διμερές, είτε (β) κάθε πράκτορας έχει το πολύ $\lceil \frac{n}{4} \rceil - 1$ γείτονες, όπου n είναι ο συνολικός αριθμός των πρακτόρων, είτε (γ) ο συντομότερος κύκλος με μη παράλληλες ακμές έχει μήκος τουλάχιστον 6.

CONTENTS

CHAPTER 1

INTRODUCTION

We study a problem of "fairly" dividing indivisible goods to many agents. The question of how to divide resources to several agents in a fair way dates back to the ancient times, e.g., dividing land, and it raised important research questions since the late 40's [37]. One prominent notion in fair division is *envy-free allocations*, where nobody envies what is allocated to any other agent, which was formally introduced a bit later [24, 23, 39]. Initially, the problem was studied under the scope of divisible resources, where envy-free allocations are known to always exist [7].

The focus of this thesis is on indivisible goods, with multiple applications, such as dividing inheritance, and assigning courses to students [13]. The non-profit website Spliddit (`http://www.spliddit.org/`) provides mechanisms for several such applications. It is easy to see that envy-free allocations are not guaranteed to exist; for instance consider two agents and one indivisible good, then whoever gets the good is envied by the other agent. This example demonstrates how strong the requirement of completely envy-freeness is for the scenario of indivisible goods.

This has led to the study of two basic relaxations of envy-freeness, namely envy-freeness up to one good (EF1) [12] and envy-freeness up to any good (EFX) [15]. EF1 is a weaker notion than EFX, and it is guaranteed to always exist and can be found in polynomial time [31]. One the other hand, it is not known if EFX allocations are guaranteed to exist in general, and it has been characterized as "Fair Division's Most Enigmatic Question" [36]. EFX allocations are known to exist for special cases: e.g., for 2 agents with general monotone valuations [35], for 3 agents with additive valuations or a slightly more general class [17, 2], for agents with at most three types of additive valuations [38], and for many agents with identical monotone valuations [35], or with additive valuations where each agent is restricted to have one of the two fixed values for each good [6].

Surprisingly, it was recently shown that EFX allocations need not exists in the case with chores, i.e., negatively valued items [21]. This is the first result of non-existence of EFX for monotone valuation functions, and the construction requires only 3 agents and 6 goods. This is an interesting separation between goods and chores, as for the case of goods it is known that EFX allocations are guaranteed to exist when the number of goods are at most 3 more than the number of agents [32].

Unfortunately, little is known for the case with multiple agents and multiple goods;

additionally to the works that have been already mentioned [6, 35], EFX allocations are known to exist when agents' preference follow a lexicographic order defined by their preference over singletons [27], and when the valuations have dichotomous marginals, i.e., the marginal value when a good is added to a set is either 0 or 1 [8]. All those works consider high restrictions and resemblance on the agents valuations. Towards broadening our understanding for the case of multiple agents and goods, Christodoulou et al. [20] introduced a setting that is related to our work, where the valuations are defined based on a graph: given a graph, the agents correspond to the vertices of the graph, and the goods to the edges. Then, each agent is indifferent for the goods/edges that are not adjacent to them. In [20], they showed that EFX allocations always exist on graphs.

The motivation in the multigraph setting is, similarly to [20], the division of territories between nations, areas of interest between neighboring countries and more generally division of geographic settings. Another application is to allocate available space for research teams and collaborators, which is always a challenging task, and becomes even more difficult when there are multiple conflicts for available areas. It was recently showed that EFX allocation always exists in multigraphs when all agents have restricted additive valuations [30], i.e., each good g has a fixed value v_g, and each agent may value the good by v_g or not value it at all, in which case he has value 0. We generalize this result with respect to the valuation functions, where we use general monotone valuations, on the expense of having restrictions on the multigraph, where either the multigraph is bipartite, or each agent has at most $\lceil \frac{n}{4} \rceil - 1$ neighbors, where n is the total number of agents, or the shortest cycle with no parallel edges has length at least 6.

Two other papers independently and in parallel showed EFX allocations involving multigraphs [1, 11]. We next discuss their results and the comparison to ours. [1] independently showed that EFX allocations exist for bipartite multigraphs, which coincides with our first result. They further showed the existence of EFX allocations for multicycles with additive valuations. Our first and third results (for bipartite multigraphs and graphs with cycles of non-parallel edges of length at least 6) include all multicycles apart from the ones of length 3 and 5; we note however that we consider the more general monotone valuations. [11] independently showed the existence of EFX allocations in bipartite multigraphs when agents have cancelable valuations and in multi-trees for general monotone valuations. Our first result generalizes those two results. Moreover, they showed that multigraphs with chromatic number t admit EFX allocations when the shortest cycle using non-parallel edges is of length at least $2t - 1$; this result holds when agents has cancelable valuations. Note that the multigraphs with chromatic number 2 is just the bipartite multigraphs. When the chromatic number is at least 4, we improve the requirement for shortest cycles using non-parallel edges, from length at least 7 (and greater depending on the chromatic number of the multigraph) to 6 (no matter the chromatic number), and furthermore our results is more general as it applies to general monotone valuations. On the other hand, for multigraphs with chromatic number 3, [11] showed the existence of EFX allocation (for cancelable valuations) for an improved length of 5 (comparing to 6 in our result) for the shortest cycle using non-parallel edges.

Thesis roadmap. The structure of the thesis from this point on is the following: **An Overview of the recent literature** which discusses related work to EFX and the different recent directions of the field. After that a **Background** chapter concerning definitions, basic widespread algorithms in the literature, variant definitions of Envy-Free allocation with important results related to the notion of EFX. The **Background** chapter aims to provide useful tools and approaches that help digest the original re-

sults of the thesis, described in the chapter **Almost Envy-allocations in multigraphs**. **Our contribution** is a construction of EFX allocations if one of the three conditions hold: either (a) the multigraph is bipartite, or (b) each agent has at most $\lceil \frac{n}{4} \rceil - 1$ neighbors, where n is the total number of agents, or (c) the shortest cycle with non-parallel edges has length at least 6. We describe the results in such a way to showcase a 3 Step framework for EFX allocations on multigraphs. Finally, **Conclusion** contains future directions and a discussion of our original results.

CHAPTER 2

OVERVIEW OF THE RECENT LITERATURE

The scope of the thesis is the fairness criterion EFX, we present an overview of the recent advancements in the field of EFX and approximate EFX allocation of indivisible items for non strategic agents in an offline setting.
We need to note that it is not shown that EFX allocations do not exist in specific settings. A counter-example to the existence of EFX allocations for non-negatively valued items would be a breakthrough to the field.
Complete EFX allocations. There are positive results for restricted cases. It was shown that EFX allocations always exist when all agents share a valuation function [35]. The restriction on the valuation functions of agents are still pretty strict and revolve around additive valuation function. Some notable cases in which complete EFX allocations exists is for dichotomous valuations i.e. each good adds 0 or 1 to their value, when goods have one of two possible values, when agents have in total at most 3 different additive valuations, and when there is a common lexicographic preferences over items [8], [6] [27], [38]. If the restriction are on the number of agents or goods, there are some brighter results. For two agents with different valuations there is always an EFX allocation [35], while for three agents only when one of them has a valuation slightly more general than an additive one is shown that an EFX allocation exists [2]. Also for at most $n + 3$ goods, for n agents, there is always an EFX allocation [33].
Restriction of the relevant items per agent. These results refer to complete EFX allocations, but is an interesting approach in the restriction of agents' valuations. Closer to the original results of the thesis, in [20] was introduced the (p, q)-bounded setting, where an item is relevant to at most p agents, and has multiplicity q. In [30], they use a bid altered notation where q represents the maximum number of items that are relevant to a pair of agents. In both notations, the multigraph setting is equivalent to the $(2, \infty)$-bounded setting. The existence of EFX allocations in the $(2, 1)$-bounded setting, which is equivalent to simple graphs, has been studied for goods [20], and for mixed manna settings [41]. In the graph and multigraph setting the existence of EFX orientations, i.e., allocations where edges may only be allocated to one of the endpoints, has also been considered. In [20] they showed that EFX orientations need not exist by giving a counterexample in a K_4 graph, and they further showed that even deciding if there exists an EFX orientation is NP-complete; it was later shown that this result holds even if the vertex cover of the graph has size of 8, or in multigraphs with only 10 vertices [22].

In [40], they showed that EFX orientations are not guaranteed to exist in graphs with chromatic number greater than 3, and they always exist when the chromatic number is at most 2. Regarding multigraphs, a very recent work [28] showed that finding an EFX orientation is NP complete even for bipartite multigraphs.

Approximate EFX. Several relaxations of EFX have also been explored. One such relaxation is the approximate EFX, α-EFX, where no agent envies any strict subset of another agent's allocated set by more than an $\alpha \leq 1$ parameter. For subadditive valuations a $\frac{1}{2}$-EFX allocation is known [35]. The approximation was improved to $\phi - 1$, but for the more restricted class of additive valuations [5]. The approximation has been pushed further to $\frac{2}{3}$ for additive valuations under several restrictions [3, 34]. Related to our work, in [3] they showed $\frac{2}{3}$-EFX for additive valuations in the setting of multigraphs. For the $(\infty, 1)$ - bounded setting (according to [30]), $\frac{\sqrt{2}}{2}$-EFX allocations are guaranteed to exist even for subadditive valuations [30].

Partial EFX allocations. Another relaxation of EFX, introduced by Caragiannis et al. [14], is to consider partial EFX allocations, also known as EFX with charity. Chaudhury et al. [18] showed that EFX allocations always exist, even with general monotone valuations, if at most $n - 1$ goods are donated to charity, where n is the number of agents, and moreover nobody envies the charity. The size of the charity was then improved to $n - 2$, and to one for the case of 4 agents with additive valuations [10]. For the $(\infty, 1)$ - bounded setting (according to [30]), the size of the charity was reduced to $\lfloor \frac{n}{2} \rfloor - 1$ for general monotone valuations [30]. Finally, the number of unallocated goods was subsequently improved to sublinear by [19, 2, 9, 29], but for $(1 - \varepsilon)$-EFX, for any $\varepsilon \in (0, \frac{1}{2}]$.

CHAPTER 3

BACKGROUND

3.1 Preliminaries

3.1.1 Basic definitions

We give formal definitions for some general concepts used throughout the thesis.

A *graph* G is an ordered pair (V, E) where V is a finite set and $E \subseteq \binom{V}{2}$. The set V is called the *vertex set* of the graph and the set E is called the set of *edges* of the graph. Given a graph G we also denote its vertex set by $V(G)$ and its edge set by $E(G)$. For $x, y \in V(G)$. We say that the vertices x and y are *adjacent* or *neighbors* if $\{x, y\} \in E(G)$.

Neighborhood Let G be a graph and vertex $u \in V(G)$. The *neighborhood* of u is the size of the set of each neighbors denoted $N(u)$.

The same definitions stand for *multigraphs*. The only difference between multigraphs and graphs is that there can be multiple edges between two vertices.

For section 4 of this thesis, we will refer to edges of the multigraph as a set, because we can enumerate each edge.

Matching Let $G = (V, E)$ be a graph. A *matching* of G is a set $M \subseteq E$, such that for $\{a, b\}, \{c, d\} \in M$ then $a \neq c, d, b \neq c, d$.

Allocation. An *allocation* $\mathbf{X} = (X_1, \ldots, X_n)$ is a partition of a subset of E into n (disjoint) bundles $X_1, \ldots, X_n$, where each agent i receives X_i. We call an allocation *c*omplete if it's a partition of the set E. We call an allocation *partial* if it's a partition of a strict subset of E. We define *s*ocial welfare as :$\sum_{i=1}^{n} = v_i(X_i)$.

Definition 3.1 (Envy - EFX-satisfied.)**.** We say that an agent i *envies* another agent j given an allocation $\mathbf{X} = (X_1, \ldots, X_n)$, if $v_i(X_i) < v_i(X_j)$.
We say that an agent i is *EFX-satisfied* against agent j given an allocation $\mathbf{X} = (X_1, \ldots, X_n)$, if $v_i(X_i) \geq v_i(X_j \setminus \{g\})$, for all $g \in X_j$.

Definition 3.2 (EFX allocation.)**.** An allocation $\mathbf{X} = (X_1, \ldots, X_n)$ is *EFX* if for any pair of agents i, j it holds that:

$$v_i(X_i) \geq v_i(X_j \setminus \{g\}) \quad \forall g \in X_j$$

We consider settings where there is a set N of n agents and a set M of m indivisible goods, and each agent i has a valuation set function $v_i : 2^M \to \mathbb{R}$, over the sets of goods, i.e., by $v_i(S)$ we denote the valuation of agent i for the set S of goods. The valuation functions are considered to be monotone, i.e., for any $S \subseteq T \subseteq M$, it holds that $v_i(S) \leq v_i(T)$, and normalized, i.e., $v_i(\emptyset) = 0$.

3.1.2 Basic Algorithms

We will present two very basic and important algorithms that are used widely in modern literature, in further sections we will show some characteristic results that use at their core these algorithms. We will define the necessary definition of envy graph.

Definition 3.3. [Envy graph] Let **X** be an allocation, we define $E_{\mathbf{X}}$ to be graph called "envy graph" with vertices representing each agent and directed edges from a vertex representing an agent i to a vertex that represents an agent j, if: $v_i(X_i) < v_i(X_j)$.

Breaking cycles in Envy graph is a very useful and powerful tool, that changes an allocation into an allocation with greater or equal social welfare with an non envied agent.

Algorithm 1 Envy Cycle Elimination

Input: An EFX allocation **X**
Output: An EFX allocation with greater social welfare.

while $\exists$ a cycle in $E_{\mathbf{X}}$ **do**
  Let the cycle be $C = (v_1, v_2, \ldots, v_k, v_1)$
  $X_t = X_1$ (We "save" the bundles)
  for $i \in [k-1]$ **do**
    $X_{v_i} \leftarrow X_{v_{i+1}}$
  end for
  $X_{v_k} \leftarrow X_{v_1}$
end while

Claim 3.4. The algorithm Envy Cycle Elimination terminates and increases the social welfare.

Proof. The definition of the Envy graph states that there is a directed edge (i, j) between two agents if and only if agent i envies agent j. During the resolution of a cycle, every agent got a bundle that they envied, so they valued it more than their previous bundle. The algorithm terminates, since there are a finite number of cycles in a simple graph. □

We will proceed with a very simple algorithm called "round robin", given an order of the agents from 1 to n, we let each of them pick their most valued good. This algorithm will only be considered for agents with additive valuations.

3.2 Envy-Free relaxations

3.2.1 EF1

The first relaxation of envy based fairness criteria was EF1. We define what is an EF1 allocation:

Algorithm 2 Round Robin

Input: A set of players N and goods M, an allocation $\mathbf{X}$, a vector $order$ with size n that contains numbered agents from 1 to n, and a number of iterations T
Output: A new allocation.

Iterations ← T
index ← 1
while Iterations > 0 **do**
    $g = \text{argmax}_{k \in P}\, v_{order[index]}(k)$
    $X_{order[\text{index}]} \leftarrow X_{order[index]} \cup g$
    Iterations ← Iterations - 1
    index ← (index + 1)mod n
end while

Definition 3.5 (EF1 allocation)**.** An allocation $\mathbf{X} = (X_1, \ldots, X_n)$ is EF1, if for any agents i, j:

$$\exists g \in X_j : v_i(X_i) \leq v_i(X_j \setminus \{g\})$$

This notion of fairness was properly defined in [12], but there was already a polynomial algorithm in [31] that constructed an EF1 allocation for agents with general monotone valuations. We present the algorithm that Lipton et Al constructed. The algorithm creates an initial allocation where each agent has exactly one item. After the initial allocation, Envy Cycle Elimination is executed, this guarantees a non-envied vertex exists. The rest of the algorithm is an iteration of allocating one non-envied agent an additional item, running Envy Cycle Elimination, thus having a non-envied agent in the partial allocation, and repeating until all goods are allocated.

Algorithm 3 Envy Cycle Elimination

Input: A set of players N and goods M
Output: An EF1 allocation.

$P \leftarrow M$
Give each agent an arbitrary good. (for agent i, $|X_i| = 1$)
Construct envy graph E_X.
Run Envy Cycle Elimination.
while $P \neq \emptyset$ **do**
    Let i be a non envied agent (a source in the envy graph) and $g \in P$
    $X_i \leftarrow X_i \cup g$
    $P \leftarrow P \setminus \{g\}$
    Update envy graph E_X.
    Run Envy Cycle Elimination.
end while
return $(X_1, \ldots, X_n)$

Lemma 3.6. (shown in [31]) Algorithm 3 constructs an EF1 allocation.

Proof. We will prove this by induction. Before running the "while", each agent has one good, so the allocation is EF1. Our inductive hypothesis will be that in every iteration up to iteration k ($k < m$), we have a partial EF1 allocation. We will show that after iteration $k + 1$, we will have an EF1 allocation. Let i be the non envied agent

before adding to their bundle good g, notice that after adding good g to X_i agent i is EF1-satisfied against other agents, due to the inductive hypothesis and the fact that $v_i(X_i \cup \{g\}) \geq v_i(X_i)$. Let j be an agent different than i, j did not envy i before adding g, therefore we have that $v_j(X_j) \geq v_j(X_i)$, so by "ignoring" g, agent j will not envy agent i. After running Envy Cycle Elimination, every agent has increased their value, so their new bundles are EF1 satisfied against other agents. Notice that in each iteration a good is removed from the pool, so this procedure will terminate at most after m iterations. □

Lemma 3.7. (again shown in [31]) Algorithm 3 runs in polynomial time.

Proof. Constructing the initial graph and updating the envy graph takes $O(n^2)$ and $O(n)$ respectively. Finding a cycle in a graph takes $O(n^2)$ time complexity (vertices represent one agent each). After adding a good, at most n edges can be added in the graph, so a cycle removal is going to be required at most $O(mn)$ times. In total we have $O(mn^3)$ time complexity. □

Also Round Robin (Algorithm 2) executed for $|M| = m$ iterations provides an EF1 allocation for agents with additive valuations.

Claim 3.8. Algorithm 2 executed for m iterations, constructs an EF1 allocation.

Proof. Consider 2 agents i, j, we will show that i's bundle is EF1 satisfied against j's. If i "picks" an item before j in the order of the algorithm, then i will not envy j. If it's the other way around, ignore the first item j got in the algorithm, then we can consider the rest of the allocation as i picking before j so i will not envy j, if we remove the first item that j "picked" in the algorithm. □

3.2.2 EFX

EFX for agents with general monotone identical valuations and the cut and choose protocol

In [35], one key result is that agents that share valuation functions always admit EFX allocations. This result used in a black box manner can provide us with a simple procedure, called cut and choose, to prove the existence of EFX allocations for 2 agents with general monotone valuations.

The Leximin++ comparison operator compares allocations. Since we are referring to identical valuation function, we can assume a lexicographic order of the agents (each agent is assigned a unique integer from 1 to n). The comparison is based on an increasing ordering of the agents of the value of their allocated items, an allocation $\mathbf{X}$ is greater that an other allocation $\tilde{\mathbf{X}}$, if in the k-th position of each ordering of the allocations, the k-th agents has greater value for their items in allocation $\mathbf{X}$ than in $\tilde{\mathbf{X}}$, or if they have the same value in allocation $\mathbf{X}$, the k-th agent is allocated more items than in allocation $\tilde{\mathbf{X}}$.

We provide an algorithm for the comparison operator Leximin++.

We will take for granted that the Leximin++ operator provides a total ordering of the possible complete allocations. We will prove that a greatest allocation based on the leximin++ operator is EFX.

Claim 3.9. If there is no allocation greater, based on the leximin++ operator, than an allocation $\mathbf{X} = (X_1, \ldots, X_n)$, then allocation $\mathbf{X}$ is EFX.

Algorithm 4 Leximin++ Compare

Input: Two allocations $\mathbf{X}, \tilde{\mathbf{X}}$ that induce an ordering of bundles in increasing order
Output: The greater allocation based on the leximin++ total ordering.

We will denote v the identical valuation function for all agents.
$i = 1$
$O\mathbf{X} \leftarrow$ order of agents from 1 to n
$O\tilde{\mathbf{X}} \leftarrow$ order of agents from 1 to n
($O\mathbf{X}_j$ is agent j in the order, similarly for $O\tilde{\mathbf{X}}$)
while $i \leq n$ **do**
    $y \leftarrow O\mathbf{X}_i$
    $k \leftarrow O\tilde{\mathbf{X}}_i$
    if $v(X_y) < v(\tilde{X}_k)$ **then**
        return $\tilde{\mathbf{X}}$
    else if $v(X_y) > v(\tilde{X}_k)$ **then**
        return $\mathbf{X}$
    else
        if $|X_y| < |\tilde{X}_k|$ **then**
            return $\tilde{\mathbf{X}}$
        else if $|X_y| > |\tilde{X}_k|$ **then**
            return $\mathbf{X}$
        end if
    end if
    $i = i + 1$
end while

Proof. We will consider that allocation $\mathbf{X}$ is ordered, such that if $t_1 < t_2$ then $v_{t_1}(X_{t_1}) \leq v_{t_2}(X_{t_2})$. For the sake of contradiction, let's assume $\mathbf{X}$ is not EFX, we will construct an allocation $\tilde{\mathbf{X}}$, that is greater than $\mathbf{X}$ based on the leximin++ operator. We assumed $\mathbf{X}$ is not EFX, thus there are agents i, j such that $v_i(X_i) < v_i(X_j \setminus \{g\})$, for $g \in X_j$, let agent i to be the last agent in the ordered allocation for which EFX is violated, which suggests that if there are agents with equal value to i, i has the biggest size bundle. Let $\tilde{\mathbf{X}}$ be $\tilde{X}_i = X_i \cup \{g\}, \tilde{X}_j = X_j \setminus \{g\}, \tilde{X}_k = X_k, \forall k \neq i, j$. Let B be the agents in $\mathbf{X}$ that have a smaller index that i (agents with less valuable or smaller bundles). Since $\mathbf{X}_i \subset \tilde{\mathbf{X}}_i$, there are two cases:

1. $v_i(\tilde{\mathbf{X}}_i) > v_i(\mathbf{X}_i)$, then i is again after every vertex in B.
2. $v_i(\tilde{\mathbf{X}}_i) = v_i(\mathbf{X}_i)$, then i is again after every vertex in B, because i will have greater size than each agent in B.

That means that B is the same in both allocations, so the comparison from leximin++ will compare at least up to agent $|B|+1$ in the order. Next, consider agents with greater index than i in the ordering from $\mathbf{X}$, this means that $j \in A$. Agent i has index $|B| + 1$ in $\mathbf{X}$'s order, now there are two cases:

1. The agent with index $|B| + 1$ in $\tilde{\mathbf{X}}$ is also i, then $\tilde{\mathbf{X}}$ is greater than $\mathbf{X}$ since i has greater size in $\tilde{\mathbf{X}}$.
2. The agent with index $|B| + 1$ in $\tilde{\mathbf{X}}$ is agent $k \neq i$, $k \in A$, so $v_i(X_i) < v_i(\tilde{X}_k) = v_i(X_k)$, so again allocation $\tilde{\mathbf{X}}$ is greater than $\mathbf{X}$.

However, no allocation is greater than **X**, we arrived to a contradiction. □

The cut-and-choose protocol is simple, let two agents i, j with m items:

1. Let i cut the items, i pretends that j has the same valuation function as i and find an EFX allocation, we described that it exists through leximin++. Define those two bundles.
2. Let j pick their most valued bundle out of the two, and allocate it to them.
3. Agent i is allocated the rest of the goods.

Lemma 3.10. The cut-and-choose protocol constructs an EFX allocation for two agents with general monotone valuations.

Proof. Agent j does not envy the bundle of items that they did not pick, and both bundles are EFX-feasible for agent i. □

Given two agents i, j, there are two ways to "cut", one based on i's valuation function and one based on j's valuation function.

3.2.3 Approximate EFX

$\frac{1}{2}$-EFX for subadditive valuations

In [35] showed an algorithm for 1/2-EFX allocations for agents with subadditive valuations.

What the algorithm does is simple: In each iteration we give to an agent that is a source in the envy graph an item, if that violates our desired guarantee, we can show that giving the recently added item to the agent that violates that guarantee gives us a $1/2$-EFX partial allocation. Using induction we can prove that this algorithm provides us with an 1/2-EFX complete allocation. This algorithm will terminate, in [35] a potential function was provided that essentially says that in each iteration either the social welfare increases or an item is given. All the possible ways that this can happen are exponential but finite, in [16] it was shown how we can modify this algorithm into a simpler algorithm that runs in polynomial time. We will provide the proof by induction for the correctness of the original algorithm in [35].

Lemma 3.11. For subadditive valuations, algorithm 5 outputs an 1/2-EFX allocation.

Proof. We will prove this by induction. In the base case, we have an allocation where nobody gets a good, so we have an EFX allocation which is a 1/2-EFX allocation. The inductive hypothesis is that in the k-th iteration of the while loop, we had an EFX allocation, we will prove that in the next iteration the allocation will remain 1/2-EFX. If the condition of "if" clause is not true then the $k+1$ iteration outputs an 1/2-EFX allocation. We proceed by assuming that the algorithm entered in the "if" clause. Any agent other than s, i, have the same bundles, before the changes in the allocation no agent envied s' bundle and i's bundle has size 1, so trivially is EFX. So we proceed to show that agents s, i are 1/2-EFX satisfied. Again, s' bundle was 1/2-EFX feasible compared to all other agents. We need to show that agent i's bundle is 1/2-EFX feasible. We denote X_i the bundle that i had before giving them only g. We will make use of two inequalities. Firstly none envied s, so $v_i(X_i) > v_i(X_s)$. Secondly $v_i(X_i) < \frac{1}{2}(X_s \cup \{g\} \setminus \{a\}) < \frac{1}{2}(X_s \cup \{g\})$ because the condition of the "if" clause is true,

Algorithm 5 1/2-EFX for subadditive valuations

Input: A set of players N and goods M
Output: An 1/2 EFX allocation.

1: $i = 1$
2: $P = M$
3: Create an Envy graph E_X with no edges.
4: **for** $i \in N$ **do** $X_i \leftarrow \emptyset$
5: **end for**
6: **while** $P \neq \emptyset$ **do**
7: Let good $g \in P$
8: $P \leftarrow P \setminus \{g\}$
9: Let s be an non envied agent.
10: $X_s \leftarrow X_s \cup \{g\}$
11: **if** $\exists i \in N, a \in X_s$, such that $v_i(X_i) < \frac{1}{2} v_i(X_s \setminus \{a\})$ **then**
12: $P \leftarrow P \cup X_i$
13: $X_s \leftarrow X_s \setminus \{g\}$
14: $X_i \leftarrow \{g\}$
15: **end if**
16: Update Envy graph E_X.
17: Run EnvyCycleElimination($X_1, \ldots, X_n$) (Algorithm 1)
18: **end while**
19: **return** $(X_1, \ldots, X_n)$

and agents have monotone valuations. Due to subadditive valuations and the second inequality, we get:

$$v_i(X_i) < \frac{1}{2} v_i(X_s) + \frac{1}{2} v_i(g)$$

using the first inequality:

$$v_i(X_i) < \frac{1}{2} v_i(X_s) + \frac{1}{2} v_i(g) < \frac{1}{2} v_i(X_i) + \frac{1}{2} v_i(g)$$

and since we $v_i(X_i)$ which leads to:

$$v_i(g) > v_i(X_i)$$

So agent i did not envy agent s and increased the value of their bundle. So we have that the allocation is 1/2-EFX satisfied against all agents.
One small detail for the correctness of the algorithm is to show that we can always have a non envied agent in line 8. The algorithm Envy Cycle Elimination makes the envy graph of an allocation acyclic, so there is always a node that has no incoming edge in the envy graph, therefore none envies them (by the definition of the envy graph). □

$\phi - 1$-EFX for additive valuations

In [4], it was shown that there is always a $\phi - 1$-EFX allocation for agents with additive valuations, we provide the proof of the result based on the skeleton of the proof in [4]. In this subsection ϕ denotes the golden ratio ($\phi \approx 1.618$), one identity of the golden ration that is used widely in this result is that $\phi - 1 = \frac{1}{\phi}$.

Let's give a description of the proof, first a partial EFX allocation is constructed that allocates to agents either one or two items. The agents with two items are the less favored agents in a run of the Round Robin algorithm. Agents that get initially one item, value that item ϕ times from the other unallocated items in the Round Robin. After constructing the initial allocation, Envy Cycle Elimination is run and one of the remaining items is allocated to a non-envied agent. The last procedure is repeated until no unallocated items remain.

The initial allocation is done by two runs of the Round Robin algorithm. However the second run is performed in the less fortunate agents in the first run of Round Robin. Algorithm 6 differentiate which agents get better items running an iteration of Round Robin. We give those agents the higher value items (compared to each item), the not so fortunate agents will be given two items. We will assume that we assigned to each agent a unique number from 1 to n.

Algorithm 6 Association and assigning order

Input: A set of players N and goods M
Output: A vector $order$ that contains integers from 1 to n (referring to agents), a set of agents W.

$W = \emptyset, A = N, k = 1$
$m \leftarrow |M|$
while $A \neq \emptyset$ **do**
    Let i be an agent in A.
    $h_i = \text{argmax}_{g \in M}\, v_i(g)$
    $t_i = m - |M| + 1$
    $R \leftarrow (N \setminus (A \cup W)) \cup i$
    $j = \text{argmax}_{l \in R}\, v_i(h_l)$
    if $\phi \cdot v_i(h_i) < v_i(h_j)$ **then**
        $h_i \leftarrow h_j$
        $W \leftarrow W \cup \{i\}$
        $order[k] \leftarrow i$
        $k \leftarrow k + 1$
        $A \leftarrow (A \setminus \{i\}) \cup \{j\}$
    else
        $A \leftarrow A \setminus \{i\}$
        $M \leftarrow M \setminus h_i$
    end if
end while
$S \leftarrow N \setminus W$ sorted in increasing order based on t_i, for $i \in N \setminus W$
for $i \in S$ **do**
    $order[k] \leftarrow i$
    $k \leftarrow k + 1$
end for
return $(order, W)$

The Round Robin algorithm depletes items from set M, that was given as input.

After differentiating the two types of agents, a completed Algorithm that produces a complete allocation can be run. Algorithm 7 runs two instances of Round Robin, and, until no unallocated items remain, runs Envy Cycle Elimination then allocates to a non-envied agent an item and again runs Envy Cycle Elimination.

Algorithm 7 $\phi - 1$-EFX allocation

Input: A set of players N and goods M
Output: A $\phi - 1$-EFX allocation $\mathbf{X}$

1: $(order, k) \leftarrow$ Association and assigning order
2: $\mathbf{X} \leftarrow (X_1, \ldots, X_n)$, with $X_i = \emptyset \forall$ agents i.
3: Run Round Robin$(N, \mathbf{X}, M, order, n)$
4: $reverse\_order \leftarrow (order[n], \ldots, order[1])$
5: Run Round Robin$(N, \mathbf{X}, M, reverse_order, n - k)$.
6: Create Envy graph E_X.
7: Run Envy Cycle Elimination.
8: **while** $M \neq \emptyset$ **do**
9: Give an item to a non envied vertex.
10: Update Envy graph E_X.
11: Run Envy Cycle Elimination.
12: **end while**

Lemma 3.12. Algorithm 6 associates agent i with an item h_i depending on if they are in set W or not, such that:

1. $v_i(h_i) > \phi \cdot v_i(g), \forall i \in W, g \in M \setminus (\bigcup_{k=1}^{n} h_k)$
2. $\phi \cdot v_i(h_i) \geq v_i(h_j), \forall i, j \in N \setminus W$

Proof.

1. Algorithm 6 puts an agent i in set W, only in the last time they changed bundle, so for items non associated with some agent, the condition of the "if" is satisfied then agent i prefers every non associated item ϕ times more than theirs.
2. If $t_i > t_j$, then i picked first their item and so $v_i(h_i) \geq v_i(h_j)$ and trivially $\phi \cdot v_i(h_i) \geq v_i(h_j)$. If $t_i < t_j$ then $if \phi \cdot v_i(h_i) < v_i(h_j)$, i in the last iteration they were considered would be in set W, which leads us to a contradiction.

□

In the first run of Round Robin in algorithm 7, each agent i is given the h_i with the properties in lemma 3.12. In the second run of Round Robin we will denote the items that agents in W got as sh_i. The analysis proceeds with considering the last item that is allocated to each agent and the notation of h and sh items will be used.

Lemma 3.13. The allocation $\mathbf{X} = (X_1, \ldots, X_n)$ that algorithm 7 returns is $(\phi - 1)$-EFX.

Proof. Consider two agents i, j with $i \neq j$, we show that i is $(\phi - 1)$-EFX feasible for j's bundle. If j's bundle has size one, then it is trivially EFX for i. Assume that $|X_j| \geq 2$ and let l be the last item added to X_j. The last item is either added in the second Round Robin or was added right before running Envy Cycle Elimination. In case the item previously belonged to another agent (in case we switched bundles with the Envy Cycle Elimination algorithm), we will refer that agent as p and their bundle X_p^{prev}. There are 4 cases, depending on if i belongs to W and when l was added. When l was added i's bundle might be different, we will denote i's bundle right before l was added as X_i^{prev}. Envy Cycle Elimination does not give to an agent a bundle they value less, therefore $X_i \geq X_i^{prev}$.

1. $i \in W$, and l was added in Round Robin. We have that $X_i \geq h_i$, $X_j = \{h_j, sh_j\}$, $v_i(h_i) \geq v_i(h_j)$ and $v_i(h_i) \geq v_i(sh_j)$, so $v_i(X_i) \geq v_i(h_i) \geq v_i(\max\{h_j, sh_j\})$. This in turn means $v_i(X_i) \geq \max_{g \in X_j} v_i(X_j \setminus \{g\}) > \frac{1}{\phi} \max_{g \in X_j} v_i(X_j \setminus \{g\})$

2. $i \in W$, and l was added right before Envy Cycle Elimination. When l was added, p was non envied, so $v_i(X_i^{prev}) \geq v_i(X_p^{prev})$. Since l was not added in Round Robin then $v_i(X_i^{prev}) > v_i(h_i) > \phi v_i(l)$ due to lemma 3.12. With these inequalities in mind:

$$v_i(X_j) = v_i(X_p^{prev}) + v_i(l) \leq v_i(X_i^{prev}) + \frac{1}{\phi} v_i(X_i^{prev})$$

$$(1 + \frac{1}{\phi}) v_i(X_i^{prev}) = \phi \cdot v_i(X_i^{prev}) \leq \phi \cdot v_i(X_i)$$

3. $i \notin W$, and l was added in Round Robin. Agents $i, j \in N \setminus W$. If i picked before p in the first run of Round Robin, then the analysis of the first case suffices. Otherwise, agent i picked after p in the first Round Robin, but i picked first in the second Round Robin, so $v_i(sh_i) \geq v_i(sh_p)$. Since l was the last item added to the bundle j got, then $X_j = \{h_j, sh_j\}$. So:

$$v_i(X_i) \geq v_i(X_i^{prev}) \geq v_i(h_i) + v_i(sh_i)$$

Due to lemma 3.12 and the fact that i chose before p in the second Round Robin:

$$v_i(h_i) + v_i(sh_i) \geq \frac{1}{\phi} v_i(h_p) + v_i(sh_p) \geq \frac{1}{\phi} v_i(h_p) + \frac{1}{\phi} v_i(sh_p) = (\phi - 1) v_i(X_j)$$

4. $i \notin W$, and l was added right before Envy Cycle Elimination. Since p was non envied then $v_i(X_i^{prev}) \geq v_i(X_p^{prev})$. Since l was added right before Envy Cycle Elimination and the two choices of Round Robin, then $v_i(h_i) \geq v_i(sh_i) \geq v_i(l)$, and we get that:

$$2 \cdot v_i(l) \leq \min\{h_i, sh_i\} \Rightarrow v_i(l) \leq \frac{1}{2} \min\{h_i, sh_i\} \leq v_i(X_i^{prev})$$

Using these inequalities:

$$v_i(X_j) = v_i(X_p^{prev}) + v_i(l) \leq v_i(X_i^{prev}) + \frac{1}{2} v_i(X_i^{prev}) \leq \phi \cdot v_i(X_i)$$

Remember that $\phi \approx 1.618$ and that:

$$v_i(X_i) \geq \frac{1}{\phi} v_i(X_j) = (\phi - 1) v_i(X_j)$$

□

Claim 3.14. Algorithm 7 runs in polynomial time.

Proof. Algorithm 7 is just algorithm 3 with two runs of Round Robin. □

3.2.4 EFX allocations on graphs

In [20], for agents with valuations that can be represented as such: each good is represented as an edge and their endpoints as the only agents that are interested in the good, it was shown that there is always an EFX allocation. We will explain in a high level the proof roadmap and the challenges that arise in this setting and some core ideas that are present in the results about multigraphs as well.
The way the construction of the EFX allocation in graphs shares elements with the way we study the problem for multigraphs. One key concept they share, is the satisfaction of certain properties that imply that an EFX allocation can be constructed. Furthermore, they share two steps that guarantee some needed properties and then a final step for allocating any remaining, from the previous two steps, goods that have not been allocated yet.
The first step is a greedy algorithm that works like Round Robin, but agents pick in a chain like manner, e.g. if agent i chooses their most valued edge (i, j), then agent j will pick their most valued edge and so on. This algorithm guarantees two important properties:

1. We have an EFX orientation
2. Any agent prefers their allocated edge/good from their current relevant unallocated goods.

Proof. Each agent gets one good, so the first property is immediately guaranteed. Every agent gets their most or second most valued good, so any other goods have less value for them, hence the second property holds. □

After that a second algorithm proceeds, briefly described, any envied agent is offered all their neighboring unallocated edges (the unallocated goods that they are interested in), if they prefer those over their current single good, they release that good and they get the bundle of relevant goods, then in a cascading manner as in the first algorithm each agent is asked if they would prefer a new unallocated edge/good. This makes any previously envied agent non envied, because they release the good that created envy, and makes them non interested in what the other agents will do with any unallocated goods. If the envied agent prefers their one good over the others, no changes are made but a third property holds: Any envied agent, does not value their unallocated neighboring goods more that the one good they hold. The second algorithm also creates something interesting, any envied vertex that turns into a non envied one (their neighbors have relevant goods that they value more) and we can "park" to this agent any unallocated neighboring edges of an envied vertex. This is called a "safe" vertex in [20] and is a main obstacle in the multigraph setting. However, in the graph setting is an obstacle as well, the second described algorithm is not guaranteed to change any bundles. If no changes are made by the second algorithm, a case by case analysis shows that changing the allocation a bit two envied vertices will share a "safe" vertex.

"Safe" vertices allow to give edges/goods that are relevant to two envied vertices, the EFX criterion is satisfied only because the envied vertices have only one good, breaking that, can break the construction of the allocation.

This is why, we impose restrictions on the multigraph setting, we want to make sure that for any pair of envied vertices that share interest to an unallocated good, can "safely park" goods to another agent. The final step in graphs and multigraph setting is very similar: we do a final allocation of the remaining unallocated edges. In the graph setting, an edge between two non envied vertices can be given to any one endpoint,

an edge between an envied and a non-envied vertex is given to the non-envied vertex. The problematic edges are those between envied vertices, it is needed to give these edge to a "safe" vertex. This mindset transfers to the multigraph setting, but it's even more difficult, now we have at least two unallocated edges between two vertices and the "safe" vertices are not so easy to find.

CHAPTER 4

ALMOST ENVY-FREE ALLOCATIONS IN MULTIGRAPHS

This chapter is exclusively about the contribution of the original results of the thesis. We begin by describing in a high level what and how we achieve our results, while providing some necessary definition to proceed. We lead with the simpler setting of at most 2 parallel edges per two agents,that serves as a warm up that encapsulate the central ideas of our results, and then we generalize the results for many parallel edges.

4.1 Our results

We show that an EFX allocation exists for multigraphs when agents have general monotone valuations and:

- The graph is bipartite
- Each agent has at most $\lceil \frac{n}{4} \rceil - 1$ neighbors, where n is the total number of agents
- The shortest cycle with non-parallel edges of the multigraph is at least 6

Our results are summarized in the following three theorems:

Theorem 1. In bipartite multigraphs, an EFX allocation always exists.

Theorem 2. In multigraphs with at most $\lceil \frac{n}{4} \rceil - 1$ neighbors per agent, where n is the total number of agents, an EFX allocation always exists.

Theorem 3. In multigraphs, where the shortest cycle with non-parallel edges has length at least 6, an EFX allocation always exists.

The construction of EFX allocations follows the same skeleton for all our results, and so they are presented together. In order to keep the presentation smooth, we begin in Section 4.4 with a special case of multigraphs for which each pair of vertices is connected with *at most two* edges. As we discuss in Section 4.2, in the general case, we consider different ways to partition the common edges between two vertices. For simplicity of the presentation, we postpone this crucial technicality for later and consider

at most two parallel edges, for which there is a single way to be partitioned. This way we focus on highlighting first all the important ideas for constructing EFX allcations. In Section 4.5 we consider the general case where each pair of vertices is connected with multiple edges.

4.2 Our Techniques

Here we discuss our techniques in order to construct EFX allocations in multigraphs.

We make use of the cut-and-choose-based protocol of [35] for two agents: one agent cuts the set of goods into two bundles where he is EFX-satisfied with each of them (i.e., no matter which of the two bundles he receives, he does not envy the other bundle up to any good), and the other agent chooses his favorite bundle among those two. This simple protocol results in an EFX allocation for two agents, even if they have general monotone valuations. Note that there may be two different EFX allocations derived by the cut-and-choose protocol, depending on who "cuts", and moreover, only the agent who "cuts" might be envious of the other agent (i.e., the agent who chooses).

Remark 1. *We remark that according to the original definition of EFX in [15], where each agent i is not envious against any other agent j after the hypothetical removal of a* positive valued *good for i from the j's bundle, the cut-and-choose protocol provides a simple EFX allocation for multigraphs: for every pair of vertices i, j with common adjacent edges E_{ij}, use the cut-and-choose protocol to decide the allocation of E_{ij}. This allocation is in fact an* orientation*, i.e., each edge is given to one of its endpoints, and satisfies EFX. The reason is that i is indifferent about any other good that j receives apart from E_{ij}, and E_{ij} is allocated between i and j in such a way that i is envy free up to any positively valued good against j. However, following the traditional definition of EFX, the good that is hypothetical removed from j's bundle may be indifferent for i, and the local allocation of the cut-and-choose protocol is insufficient. Instead, we need to consider the whole multigraph more globally.*

Following the above remark, we make use of the cut-and-choose protocol in order to partition the set E_{ij} into two bundles, but we may consider two different partitions depending on which endpoint "cuts". The reason for that is so that we control the direction of envy, and manage to generalize the ideas of [20]. However, we also put the cut-and-choose protocol in use in a different way: if two agents do not agree on having the same cut, we use the EFX-cut of one of them in order to create a partition of *three* bundles where the two agents have different most preferred set. This tool was proven to be very useful for constructing the EFX allocation for Theorem 2, where we want to minimize the number of envied agents. In both approaches, one crucial condition that we always upkeep is that *we never allocate more than one bundles of the partition of E_{ij} to the same vertex.*

Our approach can be seen as a three-step procedure: i) We define an initial allocation where each agent receives exactly one bundle (derived from carefully constructed partitions) from the common edges with exactly one of his neighbors. In this step we guarantee some ground properties on the allocation. ii) We perform Algorithm 11 (a generalization of Algorithm 2 of [20]) in order to satisfy extra properties by preserving an EFX orientation, while ensuring that any non-envied agent has received exactly one adjacent bundle associated with *each* of their neighbors. At this step we are done with any orientation of the edges, whose allocation will not change in the next step. iii)

We appropriately allocate all the unallocated edges to non-envied vertices that are not endpoints of the edges, while preserving the allocation of Step 2 and the EFX guarantee.

Initial Allocation. One important novelty of our work is about the construction of the initial allocation in Step 1. In all our three results, we carefully define for each pair of vertices a partition of their common edges into 2 or 3 bundles, such that in the initial allocation the following two properties are satisfied:

1. it is an EFX orientation;
2. nobody prefers an unallocated bundle of edges (based on a defined partition) to what they have.

For showing Theorem 1, where the multigraph is bipartite, we use the cut-and-choose protocol for partitioning the edges, where the vertices on the one side EFX-cut and the vertices from the other side choose. Then, the vertices from the former side choose their favorite available bundle, resulting in a initial allocation with the following Property:

No two envied vertices are adjacent.

For Theorem 2, we construct an initial allocation with bounded envy as follows: we properly partition the parallel edges between any pair of vertices into 2 or 3 bundles; if there is a partition into two bundles that is a "cut" for both endpoints we keep this partition, otherwise we construct a partition into 3 bundles where the top preference for each endpoint is different (Lemma 4.22). Given those partitions, we create a special weighted bipartite graph between all vertices and the created bundles of edges, where each vertex is only connected with its most preferred and second most preferred bundle with appropriately weights. Then, the maximum matching in that graph induces an initial allocation that satisfies the following Property:

At most half of the agents are envied.

For Theorem 3, starting by any partial allocation that satisfies the above properties 1 and 2, we alter the allocation in order to ensure the following Property:

For each envied vertex there exists a non-envied vertex in short distance (at most 2).

Reducing envy from the initial allocation. In step 2, we run algorithm 11, by focusing on reducing the number of envied vertices and giving as many bundles as possible to non-envied vertices, by possibly changing the initial allocation, laying the groundwork for the allocation of the remaining edges.

Final allocation. In step 3, we have the necessary conditions to give the rest of the unallocated bundles. We assign each such bundle to a non-envied vertex that is not an endpoint to that bundle.

4.3 Further Preliminaries

Before we jump into the details of the results, we properly define the setting and how we express unallocated goods as unallocated edges.

Multigraph setting. Let $G = (V, E)$ be a multigraph with $|V| = n$ vertices and $|E| = m$ edges. In the multigraph setting, the agents correspond to the vertices of G, and the goods correspond to the edges of G. From now on we will refer to the agents as

vertices and to goods as edges. In this setting, the edges that are not adjacent to some vertex i are *irrelevant* to it, i.e., for any $S \subseteq E$ and any $g \in E$ that is not adjacent to i, $v_i(S \cup \{g\}) = v_i(S)$. We call an good *relevant* to i, if it is adjacent to it. We further call any two edges with the same endpoints *parallel*. We slightly abuse notation, so that if S is a set of bundles of edges, $v(S) = v(\cup_{B \in S} B)$. W.l.o.g., we assume that each vertex has degree at least 2, otherwise, if some vertex i has degree 1, meaning that agent i is interested only in one good g, we may allocate only g to i, resulting in i not envying anybody else, and the condition of EFX is satisfied for any other agent against i, since i is allocated a single good.

Unallocated edges. Given a partial allocation $\mathbf{X}$, an edge e is *unallocated* if for any X_i, $e \notin X_i$. We denote with $U(\mathbf{X})$ the set of unallocated edges in $\mathbf{X}$. For each agent i, we define $U_i(\mathbf{X})$ to be the set of all the unallocated edges that have i as one of the endpoints.

EFX-cut. We heavily rely on the cut-and-choose protocol [35] when considering the parallel edges E_{ij} between two vertices i and j: i "cuts" E_{ij} by partitioning it into two bundles such that whichever bundle of the two i receives, it is EFX-satisfied against the remaining bundle. We call such a partition an *EFX-cut* for i with respect to vertex j (we may omit referring to j if it is clear from the context). In the the cut-and-choose protocol [35], then j receives its most valued bundle and i the remaining bundle.

Orientation. An *orientation* is an allocation where each allocated edge is given to one of its endpoints.

At least degree 2 in G. W.l.o.g., we assume that each vertex has degree *at least* 2, otherwise, if some vertex i has degree 1, meaning that vertex i is interested only in one good g, we may allocate only g to i, resulting in i not envying anybody else, and the condition of EFX is satisfied for any other vertex against i, since i is allocated a single good.

Observation 4.1. Given an EFX orientation $\mathbf{X}$, if some vertex i is envied, then there exists a single vertex j that envies i, and X_i contains only edges relevant to j (j is one of the endpoints for all edges in X_i).

Proof. On the contrary, suppose that vertex j envies vertex i, and there exists edge $g \in X_i$ such that g is irrelevant to j. Then $v_j(X_i \setminus g) = v_j(X_i) > v_j(X_j)$, where the equality is by definition of irrelevant edges, and the inequality is because j envies i. This is a contradiction to the fact that $\mathbf{X}$ satisfies EFX. Therefore, all edges in X_i are relevant to j, and to i since $\mathbf{X}$ is an orientation. This in turns means that for any other vertex all edges in X_i are irrelevant, and therefore there is no other vertex envying i. □

4.4 At most 2 parallel edges

In this section we prove Theorems 1, 2 and 3 for the case where each pair of vertices are connected with *at most two* edges. Regarding Theorem 2, in this special case of at most 2 parallel edges, we manage to relax a bit the restriction to at most $\lfloor \frac{n}{4} \rfloor$ neighbors (instead of at most $\lceil \frac{n}{4} \rceil - 1$).

Edges e^1_{ij}, e^2_{ij}. For any pair of adjacent vertices i, j, we call e^1_{ij}, e^2_{ij}, the i's most and second most valued edge between vertices i, j, respectively. If there is only one edge between i and j, w.l.o.g., we may add a dummy edge as their least preferred edge. Moreover, we give the following two useful definitions.

Definition 4.2. Given an EFX allocation $\mathbf{X}$, for any envied vertex i, we denote the vertex that envies them as $p_i(\mathbf{X})$.

Definition 4.3. Given an EFX allocation $\mathbf{X} = (X_1, \ldots, X_n)$, and an *envied* vertex i, let $\hat{U}_i(\mathbf{X}) = U_i(\mathbf{X}) \cup \{e^2_{ip_i(\mathbf{X})}\}$. We define the most valued set of potentially unallocated non-parallel edges as

$$UNP_i(\mathbf{X}) \in \arg \max_{S \subseteq \hat{U}_i(\mathbf{X})} \{v_i(S) | \text{ any } e_1, e_2 \in S \text{ are not parallel}\},$$

where $UNP_i(\mathbf{X})$ is chosen to have the *maximum possible cardinality*. This is the best bundle of non-parallel edges that i could get if X_i would be given to $p_i(\mathbf{X})$ (who envies i), and $X_{p_i(\mathbf{X})}$ becomes available.

Definition 4.4. Given an EFX allocation $\mathbf{X} = (X_1, \ldots, X_n)$, and a *non-envied* vertex i, let $\hat{U}_i(\mathbf{X}) = U_i(\mathbf{X}) \cup X_i$. We define the most valued set of potentially unallocated non-parallel edges as

$$UNP_i(\mathbf{X}) \in \arg \max_{S \subseteq \hat{U}_i(\mathbf{X})} \{v_i(S) | \text{ any } e_1, e_2 \in S \text{ are not parallel}\}.$$

where $UNP_i(\mathbf{X})$ is chosen to have the *maximum possible cardinality*. This is the best bundles of non-parallel edges that i could get without changing the allocation to other vertices.

Remark 2. *Given an EFX allocation* $\mathbf{X}$*, suppose that some vertex* i *is assigned* $\text{UNP}_i(\mathbf{X})$*, and the allocation of the other vertices remain unchanged, apart maybe from vertex* $p_i(\mathbf{X})$ *in the case that* i *is envied; if* $p_i(\mathbf{X})$ *received* $e^2_{ip_i(\mathbf{X})}$ *in* $\mathbf{X}$*, and that edge belongs to* $\text{UNP}_i(\mathbf{X})$*, then it is removed from* $p_i(\mathbf{X})$*'s bundle and is given to* i *along with the* $\text{UNP}_i(\mathbf{X})$*. In the new allocation, there are no parallel edges adjacent to* i *that are both unallocated. The reason is that* $\text{UNP}_i(\mathbf{X})$ *has maximum value and cardinality, so one of the two parallel edges should belong to* $\text{UNP}_i(\mathbf{X})$.

We prove Theorems 1, 2 and 3 by constructions of an EFX allocation in three steps. The first two steps are dedicated to satisfy specific properties for the allocation, and the final step is just an assignment of the remaining unallocated edges. We discuss each step separately, and we first give the required properties for the first two steps.

Properties of the allocation $\mathbf{X}$ after the 1st step.

(1) A partial EFX orientation.

(2) For any vertex i and $e \in U(\mathbf{X})$, $v_i(X_i) \geq v_i(e)$.

(3.1) No two envied vertices are adjacent. (For Theorem 1)

(3.2) The number of the envied vertices is at most $\lfloor \frac{n}{2} \rfloor$. (For Theorem 2)

(3.3) For any envied vertex i, $p_i(\mathbf{X})$ or at least one neighbor of $p_i(\mathbf{X})$ is non-envied. (For Theorem 3)

Additional Property of the allocation $\mathbf{X}$ after the 2nd step.

(4) For any envied vertex i, $v_i(X_i) \geq v_i\left(UNP_i(\mathbf{X})\right)$, and any non-envied vertex $j, v_j(X_j) \geq v_j(U_j(\mathbf{X}))$.

4.4.1 Step 1 - Initial Allocation

This step is the initial allocation towards the EFX construction. For each of our results, in this warm up case, i.e., at most 2 parallel edges between any pair of vertices, we show an initial allocation that satisfies properties (1),(2) and respectively one of the properties (3.1),(3.2) and (3.3) for each of our main theorems.

Initial allocation for bipartite multigraphs

Here we describe the initial allocation for the case of bipartite multigraphs: starting from the empty allocation, each vertex in side A is allocated their most valued edge. Then each vertex in side B is allocated their most valued edge from the remaining unallocated edges. This procedure is formally described in Algorithm 8.

Algorithm 8 Initial partial allocation for Bipartite

Input: A bipartite multigraph $G = (V = A \cup B, E)$.
Output: $\mathbf{X}$ satisfying Properties (1), (2), and (3.1).

for every vertex i **do**
    $X_i \leftarrow \emptyset$
end for
for every $i \in A$ **do**
    $X_i \leftarrow e_i \in \arg\max_{e \in U_i(\mathbf{X})} \{v_i(e)\}$
end for
for every $j \in B$ **do**
    $X_j \leftarrow e_j \in \arg\max_{e \in U_j(\mathbf{X})} \{v_j(e)\}$
end for

Lemma 4.5. Algorithm 8 outputs allocation $\mathbf{X}$ that satisfies properties (1), (2) and (3.1).

Proof. Property (1) is trivially satisfied since in $\mathbf{X}$ every vertex receives a single edge adjacent to them. Property (2) is also satisfied since when Algorithm 8 updates the bundle allocated to any vertex i, they receive their most valued edge, so they prefer X_i to any unallocated edge.

For Property (3.1) note that all vertices in side A are assigned their most valued edge before assigning any edge to side B. Since the graph is bipartite, only vertices from side B may envy vertices from side A. Therefore, all envy vertices are on side A, and in turn, no two envied vertices are adjacent. □

Initial allocation for at most $\lfloor \frac{n}{4} \rfloor$ neighbors per vertex

The purpose in this case is to find an initial allocation with bounded number of envied vertices. For this, we carefully define a bipartite (simple) weighted graph $H(G) = (A \cup B, E_H)$ based on the given multigraph G, between the vertices and the edges of G. A maximum weighted matching of $H(G)$ gives an allocation where at least half of the vertices receive their most valued edge, and the rest receive their second most valued edge (so each of them envies at most one other vertex), limiting that way the number of envied vertices.

Definition 4.6. (Bipartite graph $H(G)$). Given a multigraph G, we define a simple weighted bipartite graph $H(G) = (A \cup B, E_H)$ as follows. The set A represents the

vertices of G, so for each vertex of G there exists exactly one vertex in A. The set B represents the edges of G; there exists exactly one vertex in B for each edge in G. We create the edge set E_H as follows: we connect each vertex i of A (i.e., each vertex of G) with the *two* vertices of B that represents i's most and second most valued edge in G (adjacent to i) [1]; we set the weights of those two edges we add in $H(G)$ to 1 and 0, respectively.

We prove in Lemma 4.8 that the allocation, let it be $\mathbf{X}_M$, denoted by the maximum A-perfect matching[2] in $H(G)$ satisfies properties (1),(2), and (3.2). We first show in Observation 4.7 that there is always an A-perfect matching.

Observation 4.7. There is always an A-perfect matching in $H(G)$.

Proof. In order to prove the existence of an A-perfect matching, we show that Hall's theorem [26] holds for any $S \subseteq A$. Let N_S be the set of neighbors of S. For the sake of contradiction, assume that $|N_S| < |S|$ for some $S \subseteq A$. Each node in S has degree exactly 2, by construction, so there are exactly $2|S|$ edges in $H(G)$ with endpoints in S. Since we assumed $|N_S| < |S|$, by the pigeonhole principle there is a vertex $j \in N_S$ with degree at least 3. However, each vertex in B, and therefore in N_S, has degree at most two, as it represents an edge in G and it may only be connected with its endpoint vertices in A. This is a contradiction to our assumption, so for any $S \subseteq A$, it holds that $|N_S| \geq |S|$; by Hall's theorem, there is always an A-perfect matching in $H(G)$. □

Lemma 4.8. $\mathbf{X}_M$ satisfies properties (1),(2), and (3.2).

Proof. Let M be the maximum weighted A-perfect matching from which $\mathbf{X}_M$ is derived. Every vertex is assigned exactly one edge that is adjacent to it, so Property (1) trivially holds. Regarding Property (2), note that every vertex gets either their first or second most preferred edge in the graph. If a vertex gets their most preferred edge, then Property (2) trivially holds for that vertex. If a vertex gets their second most preferred edge, then since $\mathbf{X}_M$ is denoted by a *maximum* A-perfect matching, their most preferred edge is allocated to another vertex, so Property (2) holds for that vertex as well.

Let K be a maximal alternating path or an alternating cycle of M. Then if we replace $K \cap M$ with $K \setminus M$ (so we alter the matching on K), we end up with another A-perfect matching. If $K \setminus M$ had greater weight than $K \cap M$ then we would create a matching with higher weight which is a contradiction. So in every alternating cycle or path K, the weight of $K \cap M$ is at least $\left\lceil \frac{|K|}{2} \right\rceil$.

Claim 4.9. Any vertex in side A belongs to exactly one maximal alternating path or cycle of M.

Proof. This is due to the degree of each vertex in A side, which is exactly 2. Let i be a vertex in A, if i is part of an alternating cycle, then since i has degree exactly 2, i cannot be a part of any other cycle or path. If i is part of a maximal alternating path, i cannot be one of the end vertices, since it has degree exactly 2, and for the same reason cannot be a part of any other cycle or path. □

Adding up over all alternating paths and cycles, we get that M has weight at least $\lceil \frac{n}{2} \rceil$, meaning that at least $\lceil \frac{n}{2} \rceil$ vertices receive their most preferred edge in $\mathbf{X}_M$. This in turn means that there are at most $\lfloor \frac{n}{2} \rfloor$ vertices receiving in $\mathbf{X}_M$ their second most

[1] Note that in Section 4.3 we argued that w.l.o.g., each vertex in G is assumed to have degree at least 2.
[2] An A-perfect matching is a matching where all vertices of side A are matched.

preferred edge. Those vertices are the only vertices that envy some other vertex, and they may envy at most one other vertex (the one that receives their most preferred edge). Therefore, there are at most $\lfloor \frac{n}{2} \rfloor$ envied vertices, and hence Property (3) is satisfied. □

Initial allocation for length of the shortest cycle with non-parallel edges at least 6

This initial allocation requires an allocation that satisfies properties (1) and (2) and then we change that allocation such that it additionally satisfies Property (3.3). We may use the initial allocation $\mathbf{X}_M$, however there is a very simple greedy algorithm that satisfies properties (1) and (2): iteratively let each vertex pick their most valued edge from its adjacent unallocated edges. We present this procedure in Algorithm 9

Algorithm 9 Initial Greedy allocation

Input: A multigraph $G = (V, E)$.
Output: **X** satisfying Properties (1) and (2).

1: **for** every vertex i **do**
2: $\quad X_i \leftarrow \emptyset$
3: **end for**
4: **for** every vertex i **do**
5: $\quad X_i \leftarrow e_i \in \arg\max_{e \in U_i(\mathbf{X})} \{v_i(e)\}$
6: **end for**

Lemma 4.10. Algorithm 9 satisfies properties (1) and (2).

Proof. Every vertex is allocated exactly one adjacent edge, therefore we have an EFX orientation, and Property (1) is satisfied. Each vertex gets (in its turn) their most valued adjacent unallocated edge. Since they are indifferent for non-adjacent edges, overall, after running Algorithm 9, they do not prefer any unallocated edge to their allocated edge, resulting in satisfaction of Property (2). □

Satisfying also Property (3.3). Let **X** be an allocation satisfying properties (1) and (2), e.g., the allocation derived after running Algorithm 9. We use Algorithm 10 to transform **X** so that it additionally satisfies Property (3.3).

The idea of this algorithm is that when for an envied vertex i, the vertex $j = p_i(\mathbf{X})$ is also envied, and so do all its neighbors, j releases its allocated edge which is given to the vertex that envies j. Then, j receives its most valued available edge, if any, and properties (1) and (2) still hold. Note that j becomes non-envied and all its neighbors were envied before, and they may or may not be envied after the reallocation; so, overall, the envy only reduces, and Property (3.3) is now satisfied for i.

Lemma 4.11. Algorithm 10 terminates and the allocation returned satisfies properties (1),(2) and (3.3).

Proof. We will show that if before each round of Algorithm 10 properties (1) and (2) were satisfied, those properties hold after the completion of that round (outer while loop), and moreover, that the number of envied vertices is reduced by at least one. This would mean that Algorithm 10 terminates and Properties (1),(2),(3.3) are satisfied; Property (3.3) should be trivially satisfied after the termination of Algorithm 10 due to the condition of the outer while loop.

It is easy to see that properties (1) and (2) are satisfied after any round of the outer while loop: Property (1) is satisfied since each vertex receives at most one of its adjacent

Algorithm 10 Initial Allocation for Theorem 3

Input: An allocation $\mathbf{X}$ satisfying Properties (1)-(2).
Output: An allocation satisfying Properties (1),(2),(3.3).

1: **while** $\exists$ envied vertex i for which Prop. (3.3) is not satisfied **do**
2: Let $j = p_i(\mathbf{X})$ and $k = p_j(\mathbf{X})$, then
3: $U(\mathbf{X}) \leftarrow U(\mathbf{X}) \cup X_k$
4: $X_k \leftarrow X_j$
5: $X_j \leftarrow \emptyset$
6: **while** $\exists$ vertex l and $e \in U_l(\mathbf{X})$ s.t. $v_l(e) > v_l(X_l)$ **do**
7: $U(\mathbf{X}) \leftarrow U(\mathbf{X}) \cup X_l$
8: $X_l \leftarrow \{e\}$
9: **end while**
10: **end while**

edges, and Property (2) is guaranteed by the inner while loop. Next we show that in each outer while loop, no non-envied vertex becomes envied, and at least one envied vertex, namely j, becomes unenvied; this means that Algorithm 10 terminates.

Suppose that at the beginning of the outer while loop that some vertex i is considered, properties (1) and (2) are satisfied, and vertex $j = p_i(\mathbf{X})$ and all its neighbors are envied, otherwise Property (3.3) would be satisfied for i. In lines 4, 5 j's bundle/edge is given to k, so k is still envied but the envy now comes from j. j is not envied anymore and it may only envy its neighbors, however those were envied before. Regarding the inner while loop, a vertex l may only change its assignment, after its neighbor releases an edge that l prefers, however, its neighbor released that edge because it received a better one; so, no new envy may appear. Overall, an extra envy may only come from vertex j to its neighbors (e.g., j envies k), who were envied before, and j becomes unenvied. Hence, the lemma follows. □

4.4.2 Step 2 - Satisfying Property (4)

This step starts with the initial allocation constructed in Step 1 satisfying properties (1),(2), and one of (3.1),(3.2), and (3.3). In Step 2, the initial allocation changes according to Algorithm 11 in order to additionally satisfy Property (4). We remark that all our results run the same algorithm in Step 2, no matter which of the three cases is considered.

The necessity of the first part of Property (4) is so that for each *envied* vertex i we can allocate its adjacent unallocated edges $U_i(\mathbf{X})$. We cannot allocate any of those edges to i as long as it is envied. So, the high level idea for Step 2 is that either $U_i(\mathbf{X})$, or more accurately $UNP_i(\mathbf{X})$ (see Definition 4.3), is valued for i more than X_i, in which case i receives this bundle and releases X_i (which in turn is given to $p_i(\mathbf{X})$), or i doesn't value $UNP_i(\mathbf{X})$ that much, and we can find ways to allocate it to other non-envied vertices (in Step 3) without causing envy to i. The idea is similar to Algorithm 2 of [20] but adjusted in the multigraph setting where there are parallel edges.

The necessity of the second part of Property (4) is so that for each *non-envied* vertex i we can allocate its adjacent unallocated edges $U_i(\mathbf{X})$. Vertex i "chooses" its most valued available set of non-parallel edges, so that the remaining unallocated set may be assigned to other non-envied vertices (in Step 3) without causing envy to i.

Algorithm 11 Reducing Envy Algorithm

Input: An allocation $\mathbf{X}$ satisfying properties (1) and (2).
Output: An allocation satisfying Properties (1), (2) and (4). The allocation also preserves any of the Properties (3.1),(3.2),(3.3) if they were initially satisfied.

1: **while** $\exists$ non-envied vertex k s.t. $v_k(UNP_k(\mathbf{X})) > v_k(X_k)$ or
2: $(v_k(UNP_k(\mathbf{X})) = v_k(X_k)$ and $|UNP_k(\mathbf{X})| > |X_k|)$ **do**
3: $X_k \leftarrow UNP_k(\mathbf{X})$
4: **end while**
5: **while** $\exists$ envied vertex i s.t. $v_i(UNP_i(\mathbf{X})) > v_i(X_i)$ **do**
6: **if** $e^2_{ip_i(\mathbf{X})} \in UNP_i(\mathbf{X})$ **then**
7: $X_{p_i(\mathbf{X})} \leftarrow X_{p_i(\mathbf{X})} \setminus \{e^2_{ip_i(\mathbf{X})}\}$
8: **end if**
9: $X_i \leftarrow UNP_i(\mathbf{X})$
10: **while** $\exists$ vertex j and $e \in U(\mathbf{X})$ s.t. $v_j(e) > v_j(X_j)$ **do**
11: $X_j \leftarrow \{e\}$
12: **end while**
13: **while** $\exists$ non-envied vertex k s.t. $v_k(UNP_k(\mathbf{X})) > v_k(X_k)$ or
14: $(v_k(UNP_k(\mathbf{X})) = v_k(X_k)$ and $|UNP_k(\mathbf{X})| > |X_k|)$ **do**
15: $X_k \leftarrow UNP_k(\mathbf{X})$
16: **end while**
17: **end while**

Lemma 4.12. Algorithm 11 terminates and outputs an allocation $\mathbf{X}$ that satisfies Properties (1), (2) and (4). It further preserves any of the properties (3.1),(3.2) and (3.3), if they were satisfied before applying Algorithm 11.

Proof. We first argue that properties (1) and (2) are satisfied after any outer "while", i.e., the ones at lines 1-4 and 5-17, if they were satisfied before. Meanwhile, we show that any non-envied vertex cannot be envied after Algorithm 11.

For the first "while" (lines 1-4), a non-envied vertex k gets a new bundle that it values at least as much as its previous bundle, which is composed of non-parallel edges among the unallocated edges and his own bundle (see Definition 4.4). For each vertex ℓ adjacent to k, the new bundle of k contains a single edge relevant to ℓ, that ℓ doesn't value more than its own bundle, due to Property (2) and to the fact that k was not envied. So, k remains non-envied. Moreover, by the definition of the $UNP_k(\mathbf{X})$ set, Property (1) is satisfied, and Property (2) is satisfied for vertex k. Property (2) is also satisfied for the rest of the vertices since it was satisfied before and due to the fact that k was not envied (so, any edges released by k are not more valued to what the other vertices get). The same arguments hold for the "while" of lines 13-16, which is identical to the one in lines 1-4, if properties (1) and (2) are satisfied before it is executed, as we will show next.

Regarding the "while" in lines 5-17, we will show that if properties (1) and (2) were satisfied before each round of that "while", they are satisfied when the inner while at lines 10-12 terminates. In lines 5-9, an envied vertex i receives a set that may be only envied by $p_i(\mathbf{X})$, due to Property (2) and the definition of the set $UNP_i(\mathbf{X})$ (see Definition 4.3). Moreover, Property (2) is satisfied for any other vertex apart from $p_i(\mathbf{X})$, since the only edge that was released was an edge between i and $p_i(\mathbf{X})$ (see Observation 4.1). Therefore, the inner "while" in lines 10-12 will first run for $j = p_i(\mathbf{X})$, where $p_i(\mathbf{X})$ will receive the edge that i released and caused the envy towards i

in the first place. Overall, in lines 5-12 the number of envied vertices reduces by at least 1 (namely vertex i becomes non-envied), and moreover in the "while" in lines 10-12 only an envied vertex may become non-envied and not vice versa, since a vertex may only change its assignment, after its neighbor releases an edge while receiving a better one. Additionally, after the "while" in lines 10-12, Property (1) is satisfied since only relevant edges are given to the vertices, their values may only increase and no more envy is introduced, and Property (2) is satisfied by the condition of the inner "while" (line 10).

Hence, we have showed that properties (1) and (2) are satisfied at the end of Algorithm 11, and any initially non-envied vertex remains non-envied after Algorithm 11. The latter automatically means that if any of the properties (3.1),(3.2) and (3.3) were satisfied before Algorithm 11, they are preserved after its termination.

Regarding Property (4), consider first an envied vertex i after the termination of Algorithm 11. It is trivial to see that Property (4) is satisfied for i, because i does not satisfy the condition of line 5, otherwise Algorithm 11 would not have terminate. We turn our attention to some non-envied vertex k after the termination of Algorithm 11. Obviously k does not satisfy the condition of line 13 (or the same condition of line 1, if the "while" in lines 5-17 has not been executed). This would mean that either i) $v_k(\mathit{UNP}_k(\mathbf{X})) < v_k(X_k)$, or ii) $v_k(\mathit{UNP}_k(\mathbf{X})) = v_k(X_k)$ and $|\mathit{UNP}_k(\mathbf{X})| \leq |X_k|$. The former case is not possible due to the definition of $\mathit{UNP}_k(\mathbf{X})$ that considers X_k as a possible bundle. So, focusing on the latter case, due to the maximality of $\mathit{UNP}_k(\mathbf{X})$, the set allocated to k should include at least one edge related to each neighbor (since the allocation is an orientation and it is not possible to have allocated both edges to other vertices); we add this as an observation next to be used later. Therefore, $U_k(\mathbf{X})$ is a bundle with no parallel edges, and so $v_k(X_k) \geq v_k(U_k(\mathbf{X})) = v_k(U(\mathbf{X}))$.

Observation 4.13. Let $\mathbf{X}$ be the allocation after Algorithm 11. For any non-envied vertex k, X_k contains one edge that is relevant to each of its neighbors. This in turn means that there is no unallocated edge in $\mathbf{X}$ where both endpoints are non-envied.

Finally, we argue that Algorithm 11 terminates. Note that the "while" at lines 1-4 and 13-16 either strictly increases the social welfare (i.e., the aggregate value of all vertices) or the cardinality of some allocated bundle strictly increases, so they terminate (in pseudopolynomial time). The "while" in lines 10-12 strictly increases the social welfare, so for the same reason it terminates. The "while" in lines 5-17, runs at most n times since at each round at least one vertex becomes non-envied and no non-envied vertex becomes envied. □

4.4.3 Step 3 - Allocating the rest of the edges.

At this final step, we start by an allocation satisfying Properties (1)-(4) (derived by Step 2) where Property (3) corresponds to one of the properties (3.1),(3.2),(3.3) related to one of the three different cases, and we properly allocate the unallocated edges, such that the complete allocation is EFX. We remark that the edges allocated at Step 2, remain as they are, and we only allocate once and for all the unallocated edges, to vertices other than the endpoints (so we only extend the bundles allocated in Step 2, in order to derive a full EFX allocation).

Algorithm 12 does only two things: allocates any edge between an envied vertex i and a non-envied vertex j to a different non-envied vertex k, and allocates any edge between envied agents to a non adjacent and non-envied endpoint.

The difficultly of this final allocation, is to allocate edges with envied endpoints, since we cannot allocate to envied vertices any additional edge, as this would break EFX. This is why we posed the graphical restrictions in Theorems 2 and 3, so that there is always a way to allocate those edges without breaking EFX. Therefore, the remaining unallocated edges that are adjacent to some non-envied vertex i are assigned to vertices that are not i's neighbors and therefore i has no value for their allocated bundle and any unallocated edges are not envied (Property (4)). At the same time, the unallocated edges that are adjacent to an envied vertex i are assigned to vertices that i doesn't envy, even if they receive all the unallocated edges adjacent to i additionally to their bundle, under always the restriction of not containing parallel edges (Property (4)).

Algorithm 12 Final Allocation

Input: An allocation $\mathbf{X}$ satisfying Properties (1)-(4).
Output: A complete EFX allocation.

1: **while** $\exists\, e = (i, j) \in U(\mathbf{X})$ **do**
2: Let k be a non-envied vertex such that $X_k \cup \{e\}$ contains no parallel edges, and
for any $\ell \in \{i, j\}$, $v_\ell(X_\ell) \geq v_\ell\left(X_k \cup \{e\}\right)$
3: $X_k \leftarrow X_k \cup \{e\}$
4: **end while**

In the following lemma we show that Algorithm 12 provides a complete EFX allocation for bipartite multigraphs and for multigraphs under the restrictions in the statements of Theorems 2 and 3, which completes the proofs of the theorems for the case of at most two parallel edges per pair of vertices.

Lemma 4.14. For bipartite multigraphs, or for multigraphs where either there are at most $\lfloor \frac{n}{4} \rfloor$ neighbors per vertex, or the shortest cycle with non-parallel edges has length at least 6, Algorithm 12 returns a complete EFX allocation.

Proof. Let $\mathbf{X}$ be the allocation from Step 2. First note that by Observation 4.13 all unallocated edges have at least one envied endpoint. Moreover, by the same observation, for each adjacent vertices where exactly one of them is envied, there may be at most one unallocated edge with those vertices as endpoints. We next show in the following claim that there are always the vertex k of line 2, and moreover, that there are two distinct such vertices for each pair i, j of both envied vertices, if there are two unallocated edges with those envied endpoints. Then, we argue that all edges will be allocated after running Algorithm 12 by preserving the EFX condition.

Claim 4.15. For bipartite multigraphs, or for multigraphs where either the shortest cycle with non-parallel edges has length at least 6, or there are at most $\lfloor \frac{n}{4} \rfloor$ neighbors per vertex, there always exists the vertex k of line 2 in Algorithm 12. Moreover, if there are two unallocated edges with the same envied endpoints, then there are two *distinct* such vertices of line 2.

Proof. We first give the following observation to be heavily used in the proof.

Observation 4.16. If i is an envied vertex and k is not i's neighbor, then by Property (4), $v_i(X_i) \geq v_i(S) = v_i\left(S \cup X_k\right)$, for all $S \subseteq U(\mathbf{X})$ s.t. $S \cup X_k$ contains no parallel edges.

Bipartite multigraphs. We start with the case of bipartite graphs. Note that in this case the only unallocated edges are between an envied and a non-envied vertex (by using also Observation 4.13). Consider any edge $e = (i, j) \in U(\mathbf{X})$, such that i is envied and j is not envied. We will argue that $p_i(\mathbf{X})$ is the required k vertex: By Property (3.1) it holds that $p_i(\mathbf{X})$ is non-envied, and by Property (4), no envy can be created by giving any unallocated edges to $p_i(\mathbf{X})$. Regarding vertex j, as an observation note that $j \neq p_i(\mathbf{X})$, because one edge between i and $p_i(\mathbf{X})$ has been allocated to i and causing the envy of $p_i(\mathbf{X})$, and the other has been allocated to $p_i(\mathbf{X})$ due to Observation 4.13. Since j is adjacent to i and the graph is bipartite, j is not adjacent to $p_i(\mathbf{X})$, and by Observation 4.16, no envy can be created by giving any unallocated edges to $p_i(\mathbf{X})$.

At most $\lfloor \frac{n}{4} \rfloor$ neighbors. We proceed with the case that each vertex has at most $\lfloor \frac{n}{4} \rfloor$ neighbors. We argue that for any pair of adjacent vertices i, j, there are at least two (or one) non-envied vertices that are not adjacent to neither i nor j, when i, j are both envied (or exactly one of them is envied). Or in short, if q expresses the number of envied vertices in $\{i, j\}$, we will show that there are at least q non-envied vertices that are not adjacent to either i or j.

The total number of i's and j's neighbors (including i and j) is at most $2\lfloor \frac{n}{4} \rfloor \leq \lfloor \frac{n}{2} \rfloor$, so the number of vertices that are not adjacent to either i or j is at least $n - \lfloor \frac{n}{2} \rfloor \geq \lceil \frac{n}{2} \rceil$. Among them there are at most $\lfloor \frac{n}{2} \rfloor - q$ envied vertices, so the number of non-envied vertices that are not adjacent to either i or j is at least: $\lceil \frac{n}{2} \rceil - \lfloor \frac{n}{2} \rfloor + q \geq q$.

Length of shortest cycle with non-parallel edges at least 6. We now turn our attention to the case that the shortest cycle with non-parallel edges has length at least 6. Consider any edge $e = (i, j) \in U(\mathbf{X})$, such that i is envied.

If $j = p_i(\mathbf{X})$ then j must be envied: the reason is that i is envied by j, so i has received an edge relevant to i, j, and by Observation 4.13, j has received the other bundle relevant to i, j, contradicting the fact that $e \in U(\mathbf{X})$. Therefore, $j = p_i(\mathbf{X})$ is envied, and there exists at most one unallocated edge between i and j; the other has been allocated to i. In that case, the role of k in line 2 is given to either $p_j(\mathbf{X})$ if it is non-envied, or to the neighbor of $p_j(\mathbf{X})$ that is non-envied, which is guaranteed to exist by property (3.3). In both cases, k is not i's neighbor (since the shortest cycle with non-parallel edges has length at least 6). The same holds for j, when $k \neq p_j(\mathbf{X})$. Then, by Observation 4.37, and by also using property (4) if $k = p_j(\mathbf{X})$, k satisfies the conditions of line 2.

If $j \neq p_i(\mathbf{X})$, by Property (3.3), either $p_i(\mathbf{X})$ is non-envied or a neighbor of $p_i(\mathbf{X})$ is non-envied. In the latter case, that neighbor is defined as k which is not a neighbor of either i or j, otherwise a cycle of length less than 6 would exist, which is a contradiction. By Observation 4.16, k is a vertex satisfying the conditions of line 2. In the former case, we define $p_i(\mathbf{X})$ as k, which is not a neighbor of j for the same reason as above, and it holds $v_i(\mathit{UNP}_i(\mathbf{X}) \cup X_k) = v_i(\mathit{UNP}_i(\mathbf{X})) \geq v_i(S \cup X_k)$, for all $S \in U(\mathbf{X})$ s.t. $S \cup X_k$ contains no parallel edges. By Property (4) and Observation 4.16, k is a vertex satisfying the conditions of line 2, if no other edge parallel to e has been given to k before; if j is non-envied, then the statement of the claim holds. We proceed the analysis with the case that j is also envied and there may be two unallocated edges between i and j. Let k_i be the k defined above, i.e., the non-envied vertex guaranteed by Property (3.3), by considering the envied vertex i. Also let k_j be similarly the non-envied vertex corresponding to vertex j; for the same reason as above, k_j satisfies the conditions of line 2. Then, k_i and k_j, that have distance at most 2 from i and j, respectively, should be different, otherwise there would be a cycle of length at most 5. Then, one edge between i and j may be allocated to k_i and the other one to k_j, so the claim follows for that case, as well.

□

In Claim 4.15 we showed that for any unallocated edge of **X** (the allocation of Step 2) there is always a non-envied vertex satisfying the conditions of line 2, and so Algorithm 12 terminates in a complete allocation. The condition in line 2 guarantees that whenever an unallocated edge is assigned to a vertex k, no envy is caused towards k, and since k is non-envied, EFX is preserved. □

4.5 Many parallel edges

We provide a proof roadmap that describes how we tweak our techniques to work on bundles.

Definition 4.17. (Bundles) Let i, j be two adjacent vertices and E_{ij} be the set of their common edges. Based on the cut-and-choose protocol [35], we define:

- B^i_{ij} is the bundle that i gets when i cuts E_{ij},
- $B^j_{ij} = E_{ij} \setminus B^i_{ij}$, i.e., this is what j gets when i EFX-cuts E_{ij} and j chooses.
- B^i_{ji} and B^j_{ji} are defined accordingly by swapping i, j.

We present the properties we satisfy in the general case, where the sets B_i and $UB_i(\mathbf{X})$ will be defined in the related paragraph, and $UNPB_i(\mathbf{X})$ is in the same spirit with $UNP_i(\mathbf{X})$ of Definition 4.3:

Properties of the allocation **X** after the 1st step.

(1) A partial EFX orientation, , where for any vertex i, X_i is a union of bundles from B_i.

(2) For any vertex i and $B \in UB_i(\mathbf{X})$, $v_i(X_i) \geq v_i(B)$.

(3.1) No two envied vertices are adjacent. (For Theorem 1)

(3.2) The number of the envied vertices is at most $\lfloor \frac{n}{2} \rfloor$. (For Theorem 2)

(3.3) For any envied vertex i, $p_i(\mathbf{X})$ or at least one neighbor of $p_i(\mathbf{X})$ (Definition 4.2) is non-envied. (For Theorem 3)

Additional Property of the allocation **X** after the 2nd step.

(4) For any vertex i, $v_i(X_i) \geq v_i\,(UNPB_i(\mathbf{X}))$.

Step 1 for Bipartite multigraphs

The initial allocation is very similar with the case of at most 2 parallel edges cases. However we allocate to each vertex in side A their most valued bundle from the partitions formed by the EFX-cuts of the vertices in side B.

Step 1 for bounded number of neighbors The initial allocation now allocates bundles to vertices and not just edges. The construction of the bipartite graph $H(G)$ is different regarding the B vertices, which correspond now to bundles. There are two important cases while considering the bundles of side B, and those are whether two

vertices i and j in G have a common EFX-cut for their common relevant edges or not. For any pair i, j that share a common EFX-cut, we add on side B of $H(G)$ those two bundles of this cut, otherwise we make a three partition of their common edges (and we add those bundles on side B of $H(G)$) such that i and j have a distinct most valued bundle among them (in Lemma 4.22 we show that such a partition always exists). In similar fashion as in Section 4.4, we connect each vertex in side A with their most and second most valued bundle of edges with weights 1 and 0. The same arguments as in Section 4.4 stand in order to show that the allocation corresponding to a maximum A-perfect matching satisfies Properties (1),(2),(3.2).

We remark that in the case of at most two parallel edges each pair of neighbors had a common EFX-cut. In this more general case of many parallel edges, we also require the partition of some parallel edges into three bundles instead of two. This distinction is responsible for requiring $\lceil \frac{n}{4} \rceil - 1$ neighbors (instead of $\lfloor \frac{n}{4} \rfloor$ as in the case of two parallel edges) because we now need more "safe" vertices to "park" those extra bundles.

Step 1 for multigraphs where the shortest cycle using non-parallel edges is of length at least 6

The initial allocation now allocates bundles to vertices and not just edges. We utilize a greedy approach close to Section 4.4 unified with a similar algorithm to Algorithm 10. The problem of neighbors not having the same EFX-cut appears here as well, but we deal with it in a different way (as we cannot guarantee the existence of that many "safe" vertices): we stick to only partition the parallel edges with an EFX-cut, but we may consider both EFX-cuts of both neighbors. Properties (1) and (2) are guaranteed by offering all those possible bundles by prioritizing the endpoints. In order to guarantee property (3.3) we follow the idea of Algorithm 10 and for every new non-envied vertex we create we "lock" their relevant bundles to the ones they EFX-cut, preserving that way that they will remain non-envied.

Remark 3. *We remark that using the approach of case 2, i.e. the three partitions between some pair of vertices, in case 3 would mean that we may require three non-envied vertices to "park" those bundles in case they remain unallocated in the initial allocation. However, property (3.3) and the restriction of this case (i.e., all cycles of non-parallel edges have length at least 6), guarantee the existence of only two such vertices, and therefore we have to restrict ourselves to two partitions only.*

On the other hand, using the approach of case 3 in case 2 may be also problematic. Suppose that in the bipartite graph $H(G)$ we use all bundles formed by EFX-cuts (i.e., for vertices i, j we consider $B^i_{ij}, B^j_{ij}, B^i_{ji}, B^j_{ji}$ on the B side). The question is if we would considering assigning e.g., B^i_{ji} to j in the maximum matching (i.e., if edges (j, B^i_{ji}) could be added in $H(G)$. If we allow such an edge, j could be assigned B^i_{ji} and i could possibly not be EFX-satisfied against j. If we do not allow such an edge, it may be that i is assigned B^i_{ji}, and j would be EFX-satisfied (as required for property (1)). Nevertheless, j may envy i, which is an envy not expressed in $H(G)$, and therefore, the maximum matching in $H(G)$ cannot guarantee limited envy (as in the case of at most two parallel edges).

So, overall, it seems necessary to use different approaches for the initial allocations for those two cases.

Step 2 - Satisfying Property (4). This step is essentially the same with Algorithm 11. The algorithm works in the exact same way, but now it considers non-parallel bundles instead of single edges. There is only a little caution with respect to the priority of agents while offering them bundles, similarly to Step 1 regarding Theorem 3.

Step 3 - Final Allocation. Algorithm 12 now considers bundles instead of single edges. The arguments in Claim 4.15 are exactly the same with the general case. In the case where the shortest cycle with non-parallel edges has length at least 6 or in bipartite graphs, the final allocation proceeds by just considering at most two bundles between two vertices, instead of at most two edges, but the arguments remain the same.

In the case with bounded number of neighbors, in the general case there may be one more unallocated bundle between two vertices compared to the case of two parallel edges, i.e., three bundles when both endpoints are envied, two bundles when only one endpoint is envied, and one bundle when both endpoints are non-envied. This is dealt by restricting the number of possible neighbors by one more, or more specifically, in the case of two parallel edges, the requirement was to have at most $\lfloor \frac{n}{4} \rfloor$ neighbors, whereas, in the general case we assume at most $\lceil \frac{n}{4} \rceil - 1$ neighbors.

4.5.1 Step 1 - Initial Allocation

In this step we construct the initial allocation with specific properties that are crucial towards the final EFX construction. We construct an initial allocation that satisfies properties (1),(2) and respectively one of the properties (3.1),(3.2) and (3.3) for each of our main theorems.

Properties of the allocation $\mathbf{X}$ after the 1st step.

(1) A partial EFX orientation, where for any vertex i, X_i is a union of bundles from B_i.

(2) For any vertex i and $B \in UB_i(\mathbf{X})$, $v_i(X_i) \geq v_i(B)$.

(3.1) No two envied vertices are adjacent. (For Theorem 1)

(3.2) The number of the envied vertices is at most $\lfloor \frac{n}{2} \rfloor$. (For Theorem 2)

(3.3) For any envied vertex i, $p_i(\mathbf{X})$ or at least one neighbor of $p_i(\mathbf{X})$ is non-envied. (For Theorem 3)

The sets B_i and $UB_i(\mathbf{X})$. In each of the three cases we define for each agent i a set of bundles B_i that are adjacent to i, and could be allocated as a whole to them. We require that

1. the set $\cup_{S \in B_i} S$ is the set of all edges relevant to i, and
2. for any adjacent vertices i, j, we specify a single partition of their common relevant edges, and only the bundles of this partition belong to B_i and B_j w.r.t. this pair of vertices.

Then, $UB_i(\mathbf{X})$ is the subset of unallocated bundles of B_i under some allocation $\mathbf{X}$.

Initial allocation for Bipartite Multigraphs

We first define the sets B_i for the case of bipartite multigraphs.

Definition 4.18. (B_i set for Theorem 1) For any vertex i, we define B_i as follows:

- If $i \in A$, then $B_i = \{B^i_{ji}, \forall j$ adjacent to i$\}$
- If $i \in B$, then $B_i = \{B^i_{ij}, B^j_{ij}, \forall j$ adjacent to i$\}$

Note that all edges between two endpoints are partitioned in a single way, i.e., the vertex in side B EFX-cuts.

Then, the initial allocation for the case of bipartite multigraphs is constructed as follows: Starting from the empty allocation, each vertex $i \in A$ is allocated their most valued bundle from B_i, i.e., their most valued bundle when their neighbors EFX-cut their common edges. Then, each vertex in side B is allocated their most valued bundle from the remaining unallocated bundles. This procedure is formally described in Algorithm 13.

Algorithm 13 Initial partial allocation for Bipartite

Input: A bipartite multigraph $G = (V = A \cup B, E)$.
Output: **X** satisfying Properties (1),(2) and (3.1).

for every vertex i **do**
    $X_i \leftarrow \emptyset$
end for
for every $i \in A$ **do**
    $X_i \leftarrow S_i \in \arg\max_{S \in UB_i(\mathbf{X})}\{v_i(S)\}$
end for
for every $j \in B$ **do**
    $X_j \leftarrow S_j \in \arg\max_{S \in UB_j(\mathbf{X})}\{v_j(S)\}$
end for

In the next lemma we show that the allocation derived by Algorithm 13 indeed satisfies properties (1),(2) and (3.1).

Lemma 4.19. Algorithm 13 outputs an allocation **X** that satisfies properties (1), (2) and (3.1).

Proof. Property (1) is satisfied for all vertices in side A, since they get their most valued adjacent bundle. Vertices in side B may envy some vertices from side A, however, the allocation is EFX since they partition the edges with all their neighbors according to the cut-and-choose protocol and they receive their most valued available bundle. Property (2) is also trivially satisfied since when Algorithm 13 updates the bundle allocated to any vertex i, they receive their most valued bundle from $UB_i(\mathbf{X})$, so they prefer X_i to any unallocated bundle.

For Property (3.1) note that all vertices in side A are assigned their most valued bundle before assigning any bundle to side B. Since the graph is bipartite, only vertices from side B may envy vertices from side A. Therefore, all envied vertices are on side A, and in turn, no two envied vertices are adjacent. □

Observation 4.20. After Algorithm 13, between any two adjacent vertices there are at most two unallocated bundles.

Initial allocation for multigraphs with vertices with at most $\lceil n/4 \rceil$ - 1 neighbors

One useful Property in the case of multigraphs with *at most two* parallel edges is that every pair of vertices, in some way "agree" on how to EFX-cut their common edges

(separate the two parallel edges is a EFX-cut for both vertices). However, when there are three or more parallel edges between two vertices creates, it may be the case that no EFX-cut for one vertex is an EFX-cut for the other. This would cause an issue to our matching allocation if we follow the approach of Section 4.4; Specifically, in the general case for adjacent vertices i, j, the allocated bundle when j EFX-cuts and i chooses is the most valued one for i. That bundle is not offered to j and based on the matching allocation, they can be allocated their most valued bundle (from specific bundles constructed from the cut-and-choose protocol) and envy vertex i, therefore not guaranteeing that vertices that were allocated their most valued bundle do not envy others vertices, which is how we guarantee Property (3.2). To accommodate the different EFX-cuts between vertices we prove a 3-partition of the edges between vertices without common EFX-cuts.

Definition 4.21. (PVSP, $B^1_{i(j)}, B^2_{i(j)}$) Let i, j be two adjacent vertices and E_{ij} be the set of their common edges. If there exists a partition (P_1, P_2) of E_{ij} that either corresponds to an envy-free allocation to i and j or i and j are EFX-satisfied against each other no matter how the sets are allocated to them, (i.e, (P_1, P_2) is an EFX-cut for both i and j), then we denote by $B^1_{i(j)}, B^2_{i(j)}$, i's most and second most valued bundles, respectively, between P_1 and P_2. Similarly, we define $B^1_{j(i)}, B^2_{j(i)}$, to be j's most and second most valued bundles, respectively, between P_1 and P_2. We further define PVSP to be the set of Pair of Vertices whose common edges admit such Single Partition as described above.

Next we handle the rest of the edges, and we show that for each pair of vertices $(i, j) \notin$ PVSP there exists a partition (P_1, P_2, P_3) of their common edges E_{ij} into three bundles, such that i, j have different top preference among them.

Lemma 4.22. For any pair of vertices $(i, j) \notin$ PVSP there exists a partition $(P^1_{ij}, P^2_{ij}, P^3_{ij})$ of their common edges E_{ij} into three bundles, such that i, j have different top preference among them.

Proof. Since $(i, j) \notin$ PVSP, for every partition of E_{ij} into two bundles, i, j have the same top preference. Suppose that (P_1, P_2) is an EFX-cut for i, and let P_2 be the top preference between P_1 and P_2 for both i and j. Moreover, i is EFX-satisfied with P_1 as well, whereas j is not (because $(i, j) \notin$ PVSP). This means that there exists a good $g \in P_2$, such that $v_j(P_2 \setminus \{g\}) > v_j(P_1)$, and $v_i(P_2 \setminus \{g\}) \leq v_i(P_1)$. Additionally, $v_i(\{g\}) \leq v_i(P_1)$, because if P_2 contains more than one goods, i should prefer P_1 to P_2 after the removal of any other good but g from P_2. It cannot be that $|P_2| = 1$, because then j would be EFX-satisfied with P_1, as well. So, in the partition $(P_1, P_2 \setminus \{g\}, \{g\})$, i's top preference is P_1 and j's top preference is not P_1. Therefore, for $(P^1_{ij}, P^2_{ij}, P^3_{ij}) = (P_1, P_2 \setminus \{g\}, \{g\})$, the lemma follows. □

The above partitions of the edges are next used in order to define $B_i(\mathbf{X})$, and therefore $UB_i(\mathbf{X})$, for any vertex i, that is needed for Property (2).

Definition 4.23. We define B_i to be the set of bundles adjacent to i such that

- for each pair $(i, j) \in$ PVSP, $B^1_{i(j)}, B^2_{i(j)} \in B_i$ (see Definition 4.21),
- for each pair $(i, j) \notin$ PVSP, $(P^1_{ij}, P^2_{ij}, P^3_{ij}) \in B_i$ (see Lemma 4.22).

We show that a similar matching from Section 4.4 gives us the three properties to proceed to Step 2.

Satisfying Properties (1), (2) and (3.2). The initial allocation is constructed similarly to the one in Section 4.4; we construct a similar bipartite graph, however the main difference is that on the B-side we consider bundles instead of single edges. The tricky part is to recognize the right bundles to consider. This the reason why we distinguish between pairs of neighbors with common EFX-cut (the PVSP) and the rest that do not have any common EFX-cut. In the former case we consider the two bundles of the endpoints' common partition (see Definition 4.21), and in the latter case we consider the 3-partition as defined in Lemma 4.22. Next we formally define the bipartite graph.

Definition 4.24. (Bipartite graph $H(G)$). Given a multigraph G, we define a simple weighted bipartite graph $H(G) = (A \cup B, E_H)$ as follows. The set A represents the vertices of G, so for each vertex of G there exists exactly one vertex in A. We create the set B by adding for each pair of agents $(i, j) \in$ PVSP two vertices to represent the two sets of their common partition, and for each pair of agents $(i, j) \notin$ PVSP we add three vertices to represent the three sets of the partition as defined in Lemma 4.22. We create the edge set E_H as follows: we connect each vertex i of A (i.e., each vertex of G) with the two vertices of B that are associated with i's most valued and second most valued bundle from the B-side. We set the weights of those two edges we add in $H(G)$ to 1 and 0, respectively.

We prove in Lemma 4.25 that the allocation, let it be $\mathbf{X}_M$, denoted by the maximum weighted matching in $H(G)$, satisfies properties (1), (2), and (3.2).

Lemma 4.25. $\mathbf{X}_M$ satisfies properties (1), (2) and (3.2).

Proof. Regarding Property (1), suppose that i envies j in $\mathbf{X}_M$, then $(i, j) \in$ PVSP and j must receive $B^1_{i(j)}$. So, it holds that i receives a bundle in $\mathbf{X}_M$ at least as valued as $B^2_{i(j)}$. Since i would be EFX-satisfied against j if it received $B^2_{i(j)}$, i is EFX-satisfied against j in $\mathbf{X}_M$.

Regarding Property (2), note that for every vertex i, all bundles of B_i are included on the B side of $H(G)$ (i.e., $B_i \subseteq B$). Then, i receives its most or second most valued bundle from B_i. If i receives its most valued bundle, then property (2) is trivially satisfied. It i receives its second most valued bundle, this means that its most valued bundle is allocated to another vertex due to the maximality of $\mathbf{X}_M$. Therefore, property (2) follows.

Regarding Property (3.2), with similar arguments as in the proof of Lemma 4.8, each agent gets their most or second most valued bundle from side B. Again each vertex representing an agent has degree exactly 2, thus they participate in exactly one alternating cycle or path. Since the matching is a maximum weighted matching, at least half of the vertices are matched with their top preference from side B. So, at most half of them envy another vertex. □

Observation 4.26. In $\mathbf{X}_M$, between any two adjacent vertices there are at most three unallocated bundles.

Initial allocation for multigraphs where the shortest cycle using non-parallel edges is of length at least 6

In contrast to the previous case, we need to make sure that between any two envied vertices there are at most two unallocated bundles, and therefore, we cannot use the three partition of Lemma 4.22. Instead we stick to the EFX-cuts so that only two bundles are formed. However, it may be the case that vertices do not share a common EFX-cut,

and it is quite tricky to decide which to use without violating properties (1) and (2). Therefore, at the beginning we cannot define the set B_i for every vertex i, as opposed to the previous cases. Instead, for any pair of vertices, we initially consider both partitions where each of the endpoints EFX-cuts their relevant edges. We finalize the sets B_i along with finalizing the initial allocation for this case. The final B_i are going to be a set of partitions either cut by i or a neighbor of i, the final B_i will not contain bundles that share edges.

Hence, we define some auxiliary B_i's that will be finalized later.

Definition 4.27. For multigraphs where the shortest cycle using non-parallel edges is of length at least 6, for a vertex i, we define:

$$B_i^{\text{aux}} = \{B_{ij}^i, B_{ji}^i, B_{ij}^j, B_{ji}^j, \forall j \text{ adjacent to } i\}.$$

For simplicity, until we finalize the sets B_i, we use $UB_i(\mathbf{X})$ as the set of the bundles of B_i^{aux} that are unallocated (as a whole) in $\mathbf{X}$.

Satisfying properties (1) and (2) and (3.3).

Similarly to the case with only 2 parallel edges, we first consider some initial allocation that satisfies properties (1) and (2), by ignoring how many envied vertices there may be. Then, we transform it by creating non-envied vertices to satisfy property (3.3). In order to preserve that those vertices remain non-envied, we should consider only the bundles formed when they EFX-cut the relevant edges with each of their neighbors. Therefore, the sets B_i are finalized at the end of the algorithm that satisfies all required properties.

Algorithm 14 starts by constructing an EFX orientation that satisfies properties (1) and (2), where for property (2) we consider the sets B_i^{aux} that are changing during the algorithm. Note that if for some pair of vertices i, j, vertex j is free to get B_{ji}^i at any point, i may not be EFX-satisfied against j and property (1) would be violated. On the other hand, if j did not have the chance to get B_{ji}^i, property (2) may be violated for j. To address this issue, we just "offer" B_{ji}^i first to i and then to j, and this prioritization of the endpoints for offering them specific bundles is sufficient in order to satisfy both properties (1) and (2). We follow the same tactic any time we convert some envied vertex into non-envied to satisfy Property (3.3). As we mentioned above we make sure that these vertices will remain non-envied throughout the execution of the algorithm by changing (or more accurately shrinking) the sets B_i^{aux}. At the end of the algorithm those sets become the finalized sets B^i.

Next we show that Algorithm 14 terminates in an allocation satisfying properties (1), (2) and (3,3).

Lemma 4.28. The allocation returned by Algorithm 14 satisfies properties (1), (2) and (3,3).

Proof. We first give a useful observation:

Observation 4.29. For each pair of vertices i, j, if j receives B_{ji}^i, then j is not envied.

Proof. The reason is that j may receive B_{ji}^i only in lines 6 and 18, in which cases $v_i(X_i) \geq v_i(B_{ji}^i)$. Further note that the only case that a value of an agent may decrease is in line 11. We argue that i cannot be the vertex b (line 10), if $X_j = B_{ji}^i$. The reason is that i may be vertex b, only when all of its neighbors are envied. However, i has a neighbor, namely j, that is not envied. □

Algorithm 14 Initial Greedy allocation

Input: A multigraph $G = (V, E)$ and B_i^{aux} for each $i \in V$.
Output: $\mathbf{X}$ satisfying Properties (1), (2) and (3.3) and B_i for each $i \in V$.

for every vertex i **do**
  $X_i \leftarrow \emptyset$
end for
while $\exists i, j$ and $S \in \{B_{ji}^i, B_{ji}^j\}$ where $S \in UB_i(\mathbf{X})$, such that $v_i(S) > v_i(X_i)$ or $v_j(S) > v_j(X_j)$ **do**
  if $B_{ji}^i \in UB_i(\mathbf{X})$ and $v_i(B_{ji}^i) > v_i(X_i)$ **then** $X_i \leftarrow B_{ji}^i$
  else $X_j \leftarrow S$
  end if
end while
while $\exists$ envied vertex a for which Prop. (3.3) is not satisfied **do**
  Let $b = p_a(\mathbf{X})$ and $c = p_b(\mathbf{X})$
  $X_b \leftarrow \emptyset$
  $B_b^{\text{aux}} \leftarrow \{B_{br}^b, B_{br}^r \mid \forall r \text{ adjacent to } b\}$
  for Every vertex r adjacent to b **do**
    $B_r^{\text{aux}} \leftarrow B_r^{\text{aux}} \setminus \{B_{rb}^b, B_{rb}^r\}$
  end for
  while $\exists i, j$ and $S \in \{B_{ji}^i, B_{ji}^j\}$ where $S \in UB_i(\mathbf{X})$, such that $v_i(S) > v_i(X_i)$ or $v_j(S) > v_j(X_j)$ **do**
    if $B_{ji}^i \in UB_i(\mathbf{X})$ and $v_i(B_{ji}^i) > v_i(X_i)$ **then** $X_i \leftarrow B_{ji}^i$
    else $X_j \leftarrow S$
    end if
  end while
end while
for every vertex i **do**
  for every vertex j adjacent to i **do**
    if $B_{ij}^r \in X_i \cup X_j$, for some $r \in \{i, j\}$, or it holds that $\{B_{ij}^i, B_{ij}^j, B_{ji}^i, B_{ji}^j\} \subseteq UB_i(\mathbf{X})$ **then**
      $B_i^{\text{aux}} \leftarrow B_i^{\text{aux}} \setminus \{B_{ji}^i, B_{ji}^j\}$
      $B_j^{\text{aux}} \leftarrow B_j^{\text{aux}} \setminus \{B_{ji}^i, B_{ji}^j\}$
    end if
  end for
end for
for every vertex i **do**
  $B_i \leftarrow B_i^{\text{aux}}$
end for

We start by showing that after the termination of the first while (lines 4-8) properties (1) and (2) are satisfied. Consider any vertex j envied by vertex i, after the termination of the first while. By Observation 4.29, vertex j is not assigned B^i_{ji}, therefore it is assigned B^j_{ij}. Then i gets a bundle of value at least of B^i_{ij}: either B^i_{ij} is assigned to i, or B^i_{ij} is available and since the while condition (line 4) is not satisfied, i receives a bundle at least as good. So, i is EFX-satisfied against j. Property (2) is trivially satisfied at the end of the first while since the condition of line 4 is not true. Note that this while will terminate since the social welfare strictly increases at each iteration.

Similarly, properties (1) and (2) hold at the end of the second while (lines 9-21), since the first while run as its last subroutine; only for property (1) we need to further justify that $X_i \in B^{\text{aux}}_i$, since B^{aux}_i may change inside this while, which we will show at the end. Regarding property (3.3), this is trivially satisfied at the end of the second while since since the condition in line 9 is not satisfied. However, we need to show that the second while will terminate, and for each vertex i, we always have $X_i \in B^{\text{aux}}_i$.

Regarding the termination of the algorithm, it is sufficient to show that whenever a vertex b becomes non-envied, it remains that way. This would in turn mean that no vertex a can satisfy the while condition (line 9) more than once. When a vertex b is considered, we shrink the set B^{aux}_b to contain only the two bundles that b EFX-cuts, and we keep the same bundles in the B^{aux}_r set of each neighbor r of b. By Observation 4.29, b cannot be envied as long as B^{aux}_b does not change any further. This cannot happen, because the B^{aux}_i sets may only alter with respect to bundles where both endpoints were envied before the while (this was the case with b and all its neighbors r), and clearly b is not envied anymore. Therefore, since vertex b will remain non-envied until the end of the while, vertex a cannot be considered again.

Finally, we need to show that for any neighbor r of b, X_r cannot be B^b_{rb} or B^r_{rb}, so X_r belongs to the updated B^{aux}_r. If $X_r = B^b_{rb}$, by Observation 4.29, r was non-envied before the while, and the same holds if $X_r = B^r_{rb}$, because b values X_b at least as much as B^b_{rb}, which in turn values at least as much as B^r_{rb}. In any case r cannot be non-envied, because then a would not satisfy the while condition (line 9).

The last for-loops (lines 22-29) guarantee the properties for the B_i sets. Overall, the algorithm terminates by satisfying properties (1), (2) and (3.3). □

Observation 4.30. After Algorithm 14, between any two adjacent vertices there are at most two unallocated bundles.

The rest of the steps are the same for all cases. The initial allocations between cases are different.

4.5.2 Step 2 - Satisfying Property (4).

This step starts with the initial allocation constructed in Step 1. In Step 2, the initial allocation changes according to Algorithm 15 in order to additionally satisfy Property (4), which needs the following definition:

Definition 4.31. For any envied vertex i, let $T = \{X_{p_i(\mathbf{X})}\} \cap B_i$ be the edges that $p_i(\mathbf{X})$ receives in $\mathbf{X}$ that are adjacent to i, if any (if $\mathbf{X}$ is an initial allocation from Step 1, then T is either a bundle from B_i that $p_i(\mathbf{X})$ gets, or the empty set, if $p_i(\mathbf{X})$ gets edges irrelevant to i). Let $\hat{U}B_i(\mathbf{X}) = UB_i(\mathbf{X}) \cup \{T\}$. We define the most valued set of potentially unallocated non-parallel bundles as

$$UNPB_i(\mathbf{X}) \in \underset{S \subseteq \hat{U}B_i(\mathbf{X})}{\text{argmax}} \{v_i(S) | \text{ any } B_1, B_2 \in S \text{ are not parallel}\}^3.$$

where $UNPB_i(\mathbf{X})$ is chosen to have the *maximum possible cardinality* with respect to the number of bundles from $\hat{U}B_i(\mathbf{X})$. This is the best set of non-parallel bundles that i could get if X_i would be given to $p_i(\mathbf{X})$ (who envies i), and $X_{p_i(\mathbf{X})}$ becomes available.

Definition 4.32. For any non-envied vertex i, let $\hat{U}B_i(\mathbf{X}) = UB_i(\mathbf{X}) \cup \{X_i\}$. We define the most valued set of potentially unallocated non-parallel bundles as

$$UNPB_i(\mathbf{X}) \in \underset{S \subseteq \hat{U}B_i(\mathbf{X})}{\text{argmax}} \{v_i(S) | \text{ any } B_1, B_2 \in S \text{ are not parallel}\}.$$

where $UNPB_i(\mathbf{X})$ is chosen to have the *maximum possible cardinality* (considering the number of bundles from $\hat{U}B_i(\mathbf{X})$). This is the best set of non-parallel bundles that i could get without changing the allocation of the other vertices.

Additional Property of the allocation **X** after the 2nd step.

(4) For any vertex i, $v_i(X_i) \geq v_i(UNPB_i(\mathbf{X}))$.

The necessity of Property (4) is so that we can allocate the remaining unallocated bundles at Step 3 in non-adjacent vertices. Note that for any envied vertex i we cannot allocate any of its adjacent unallocated bundles to i additionally to what it has. So, the high level idea of Step 2 is that either $UB_i(\mathbf{X})$, or more accurately $UNPB_i(\mathbf{X})$, is valued for i, so we allocate it to i, and i releases the bundle he had which caused the envy, or i doesn't value it that much so we can give $UNPB_i(\mathbf{X})$, or subsets of it, to other agents. In the same spirit, the non-envied vertices receive the best set of their adjacent available non-parallel bundles, so that allocating the rest adjacent bundles to other vertices would preserve EFX. The idea is similar to Algorithm 2 of [20], but adjusted in the multigraph setting.

Lemma 4.33. Algorithm 15 terminates and outputs an allocation **X** that satisfies Properties (1), (2) and (4). It further preserves any of the Properties (3.1),(3.2) and (3.3), if they were satisfied before applying Algorithm 15.

Proof. We first argue that properties (1) and (2) are satisfied after any outer "while", i.e., the ones at lines 1-4 and 5-17, if they were satisfied before. Meanwhile, we show that any non-envied vertex cannot be envied after Algorithm 15.

For the first "while" (lines 1-4), a non-envied vertex k gets a new bundle that it values at least as much as its previous bundle, which is composed of non-parallel bundles among the unallocated bundles and his own bundle (see Definition 4.32). For each vertex ℓ adjacent to k, the new bundle of k contains a single bundle relevant to ℓ, that ℓ doesn't value more than its own bundle, due to Property (2) and to the fact that k was not envied. So, k remains non-envied. Moreover, by the definition of the $UNPB_k(\mathbf{X})$ set, Property (1) is satisfied, and Property (2) is satisfied for vertex k. Property (2) is also satisfied for the rest of the vertices since it was satisfied before entering the while loop and due to the fact that k was not envied (so, any bundles released by k are not more valued to what the other vertices get). The same arguments hold for the "while" of lines 13-16, which is identical to the one in lines 1-4, if Properties (1) and (2) are satisfied before it is executed, as we will show next.

Regarding the "while" in lines 5-17, we will show that if properties (1) and (2) were satisfied before each round of that "while", they are satisfied when the inner while

[3] Recall that $v_i(S) = v_i(\cup_{B \in S} B)$

Algorithm 15 Reducing Envy Algorithm

Input: An allocation $\mathbf{X}$ satisfying properties (1) and (2).
Output: An allocation satisfying Properties (1), (2) and (4). The allocation also preserves any of the Properties (3.1),(3.2),(3.3) if they were initially satisfied.

1: **while** $\exists$ non-envied vertex k s.t. $v_k(\mathit{UNPB}_k(\mathbf{X})) > v_k(X_k)$ or
2: $(v_k(\mathit{UNPB}_k(\mathbf{X})) = v_k(X_k)$ and $|\mathit{UNPB}_k(\mathbf{X})| > |X_k|)$ **do**
3: $X_k \leftarrow \mathit{UNPB}_k(\mathbf{X})$
4: **end while**
5: **while** $\exists$ envied vertex i s.t. $v_i(\mathit{UNPB}_i(\mathbf{X})) > v_i(X_i)$ **do**
6: **if** $B^2_{ip_i(\mathbf{X})} \in \mathit{UNPB}_i(\mathbf{X})$ **then**
7: $X_{p_i(\mathbf{X})} \leftarrow X_{p_i(\mathbf{X})} \setminus \{B^2_{ip_i(\mathbf{X})}\}$
8: **end if**
9: $X_i \leftarrow \mathit{UNPB}_i(\mathbf{X})$
10: **while** $\exists$ envied vertex j and $S \in UB_i(\mathbf{X})$ s.t. $v_j(S) > v_j(X_j)$ **do**
11: $X_j \leftarrow S$
12: **end while**
13: **while** $\exists$ non-envied vertex k s.t. $v_k(\mathit{UNPB}_k(\mathbf{X})) > v_k(X_k)$ or
14: $(v_k(\mathit{UNPB}_k(\mathbf{X})) = v_k(X_k)$ and $|\mathit{UNPB}_k(\mathbf{X})| > |X_k|)$ **do**
15: $X_k \leftarrow \mathit{UNPB}_k(\mathbf{X})$
16: **end while**
17: **end while**

at lines 10-12 terminates. In lines 5-9, an envied vertex i receives a bundle that may be only envied by $p_i(\mathbf{X})$, due to Property (2) and the definition of the set $\mathit{UNPB}_i(\mathbf{X})$ (see Definition 4.31). Moreover, Property (2) is satisfied for any other vertex apart from $p_i(\mathbf{X})$, since the only bundle that was released was a bundle between i and $p_i(\mathbf{X})$ (see Observation 4.1). Therefore, the inner "while" in lines 10-12 will first run for $j = p_i(\mathbf{X})$, where $p_i(\mathbf{X})$ will receive the bundle that i released and caused the envy towards i in the first place. Overall, in lines 5-12 the number of envied vertices reduces by at least 1 (namely vertex i becomes non-envied), and moreover in the "while" in lines 10-12 only an envied vertex may become non-envied and not vice versa, since a vertex may only change its assignment, after its neighbor releases a bundle while receiving a better one. Additionally, after the "while" in lines 10-12, Property (1) is satisfied since only relevant bundles are given to the vertices, their values may only increase and no more envy is introduced, and Property (2) is satisfied by the condition of the inner "while" (line 10).

Hence, we have showed that properties (1) and (2) are satisfied at the end of Algorithm 15, and any initially non-envied vertex remains non-envied after Algorithm 15. The latter automatically means that if any of the properties (3.1),(3.2) and (3.3) were satisfied before Algorithm 15, they are preserved after its termination.

Regarding Property (4), consider first an envied vertex i after the termination of Algorithm 15. It is trivial to see that Property (4) is satisfied for i, because i does not satisfy the condition of line 5, otherwise Algorithm 15 would not have terminated. We turn our attention to some non-envied vertex k after the termination of Algorithm 15. Obviously k does not satisfy the condition of line 13 (or the same condition of line 1, if the "while" in lines 5-17 has not been executed). This would mean that either i) $v_k(\mathit{UNPB}_k(\mathbf{X})) < v_k(X_k)$, or ii) $v_k(\mathit{UNPB}_k(\mathbf{X})) = v_k(X_k)$ and $|\mathit{UNPB}_k(\mathbf{X})| \leq |X_k|$. The former case is not possible due to the definition of $\mathit{UNPB}_k(\mathbf{X})$ that considers X_k as

a possible bundle. So, focusing on the latter case, due to the maximality of $UNPB_k(\mathbf{X})$, the set allocated to k should include at least one bundle related to each neighbor (since the allocation is an orientation and it is not possible to have allocated both bundles to other vertices); we add this as an observation next to be used later.

Observation 4.34. Let $\mathbf{X}$ be the allocation after Algorithm 15. For any non-envied vertex k, X_k contains one bundle that is relevant to each of its neighbors.

Finally, we argue that Algorithm 15 terminates. Note that the "while" at lines 1-4 and 13-16 either strictly increases the social welfare (i.e., the aggregate value of all vertices) or the cardinality of some agents bundle, so they terminate (in pseudopolynomial time). The "while" in lines 10-12 strictly increases the social welfare, so for the same reason it terminates. The "while" in lines 5-17, runs at most n times since at each round at least one vertex becomes non-envied and no non-envied vertex becomes envied. □

4.5.3 Step 3 - Final Allocation

At this final step, we start by an allocation (derived by Step 2) satisfying Properties (1),(2),(4) and one of the properties (3.1),(3.2) or (3.3) and we properly allocate the unallocated bundles, such that the complete allocation is EFX. We define the set of unallocated bundles as $UB(\mathbf{X}) = \bigcup_{i \in N} UB_i(\mathbf{X})$, given an allocation $\mathbf{X}$.

At this final step, we start by an allocation satisfying Properties (1)-(4) (derived by Step 2) where Property (3) corresponds to one of the properties (3.1),(3.2),(3.3) related to one of the three different cases, and we properly allocate the unallocated bundles, such that the complete allocation is EFX. We remark that the bundles allocated at Step 2, remain as they are, and we only allocate once and for all the unallocated bundles, to vertices other than the endpoints (so we only extend the bundles allocated in Step 2, in order to derive a full EFX allocation). The reason of allocating those remaining bundles to vertices that are not the endpoints is because if we would allocate them to one of the endpoints, this could possibly violate the EFX condition: e.g., if an envied vertex is allocated more edges, the envious vertex would not be EFX-satisfied anymore, and if a non-envied vertex receives more edges, those would be parallel to a bundle that he already has and may cause envy to one of his neighbors who even may not be EFX-satisfied anymore.

Algorithm 16 does a simple thing: it allocates any unallocated bundle to a non-envied vertex that remains non-envied after this allocation. The existence of such non-envied vertices that are used for "parking" the unallocated bundles is guaranteed for each case separately by using the restriction of that case and properties (3) and (4). This is why we posed the graphical restrictions in the three cases, so that there is always a way to allocate those bundles without breaking EFX. Therefore, the unallocated bundles that are adjacent to an envied vertex i are assigned to vertices that i doesn't envy, even if they receive all the unallocated bundles adjacent to i additionally to their bundle, under always the restriction of not containing parallel bundles (Property (4)). Similarly, the remaining unallocated bundles that are adjacent to some non-envied vertex i are assigned to vertices that are not i's neighbors and therefore i has no value for their allocated bundle and any unallocated edges (Property (4)).

In the following lemma we show that Algorithm 16 provides a complete EFX allocation for bipartite multigraphs and for multigraphs under the restrictions in the statements of Theorems 1, 2 and 3.

Algorithm 16 Final Allocation

Input: An allocation **X** satisfying Properties (1)-(4).
Output: A complete EFX allocation.

1: **for** $\exists\ S \in UB(\mathbf{X})$ with endpoints i, j **do**
2: Let k be a non-envied vertex such that $X_k \cup S$ contains no parallel bundles, and
for any $\ell \in \{i, j\}$, $v_\ell(X_\ell) \geq v_\ell(X_k \cup S)$
3: $X_k \leftarrow X_k \cup S$
4: **end for**

Lemma 4.35. For bipartite multigraphs, or for multigraphs where either there are at most $\lceil \frac{n}{4} \rceil - 1$ neighbors per vertex, or the shortest cycle with non-parallel edges has length at least 6, Algorithm 16 returns a complete EFX allocation.

Proof. Let **X** be the allocation from Step 2. This algorithm allocates each bundle to a vertex that is not an endpoint of the bundle (vertex k in line 2). We show in the following claim that there are always the vertex k of line 2, therefore Algorithm 16 will terminate after allocating all unallocated bundles of **X**. Then, we argue the allocation returned by Algorithm 16preserves the EFX condition.

Claim 4.36. For bipartite multigraphs, or for multigraphs where either the shortest cycle with non-parallel edges has length at least 6, or there are at most $\lceil \frac{n}{4} \rceil - 1$ neighbors per vertex, there always exists the vertex k of line 2 in Algorithm 16.

Proof. Note that if r is the maximum number of bundles of any partition of parallel edges considered in the sets B_i, then the maximum number of unallocated bundles between two adjacent vertices i, j is at most

$$r + q - 2, \text{ where } r = 2 \text{ for Theorems 1 and 3, and } r = 3 \text{ for Theorem 2}, \qquad (4.1)$$

where q expresses the number of envied vertices in $\{i, j\}$. This can be seen by Observation 4.34. We give the following observation to be heavily used in the proof, and then we consider each of the three cases separately.

Observation 4.37. For any vertex i, if vertex k is not i's neighbor, then by property (4), $v_i(X_i) \geq v_i(S) = v_i(S \cup X_k)$, for all $S \subseteq UB(\mathbf{X})$ s.t. $S \cup X_k$ contains no parallel bundles.

Bipartite multigraphs. Note that in this case by property (3.1) there is no unallocated bundle between envied vertices and by (4.1) there is no unallocated bundle between non-envied vertices. So, the only unallocated bundles are between an envied and a non-envied vertex. Consider any bundle $S \in U(\mathbf{X})$ with endpoints i, j, such that i is envied and j is not envied. We will argue that $p_i(\mathbf{X})$ is the required k vertex: By property (3.1) it holds that $p_i(\mathbf{X})$ is non-envied, and by property (4), no envy can be created to i by giving any unallocated bundles to $p_i(\mathbf{X})$. Regarding vertex j, as an observation note that $j \neq p_i(\mathbf{X})$, because one bundle between i and $p_i(\mathbf{X})$ has been allocated to i and causing the envy of $p_i(\mathbf{X})$, and the other has been allocated to $p_i(\mathbf{X})$ due to Observation 4.34. Since j is adjacent to i and the graph is bipartite, j is not adjacent to $p_i(\mathbf{X})$, and by Observation 4.37, no envy can be created to £j by giving any unallocated edges to $p_i(\mathbf{X})$.

At most $\lceil \frac{n}{4} \rceil - 1$ neighbors.

We argue that for any pair of adjacent vertices i, j, there are $q + 1$ (as defined in (4.1)) non-envied vertices that are not adjacent to either i or j. The total number of i's and j's neighbors (including i and j) is at most $2(\lceil \frac{n}{4} \rceil - 1)$, so the number of vertices that are not adjacent to either i or j is at least $n - (2(\lceil \frac{n}{4} \rceil - 1) \geq 2\lfloor \frac{n}{4} \rfloor + 2$. Among them there are at most $\lfloor \frac{n}{2} \rfloor - q$ envied vertices, so the number of non-envied vertices that are not adjacent to either i or j is at least: $2\lfloor \frac{n}{4} \rfloor + 2 - (\lfloor \frac{n}{2} \rfloor - q) \geq q + 1$.

Length of shortest cycle with non-parallel edges at least 6. This case is proven similarly to the setting with at most two parallel edges, with the only difference that instead of single edges we consider bundles. For the sake of completion we give the full proof here. Consider any bundle $S \in UB(\mathbf{X})$ with endpoints i, j. By (4.1), at least one of the endpoints should be envied; so, w.l.o.g., let i be envied.

If $j = p_i(\mathbf{X})$ then j must be envied: the reason is that i is envied by j, so i has received a bundle relevant to both i, j, and by Observation 4.34, j has received the other bundle relevant to i, j, contradicting the fact that $S \in UB(\mathbf{X})$. Therefore, $j = p_i(\mathbf{X})$ is envied, and there exists at most one unallocated bundle between i and j; the other has been allocated to i. In that case, the role of k in line 2 is given to either $p_j(\mathbf{X})$ if it is non-envied, or to the neighbor of $p_j(\mathbf{X})$ that is non-envied, which is guaranteed to exist by property (3.3). In both cases, k is not i's neighbor (since the shortest cycle with non-parallel edges has length at least 6). The same holds for j, when $k \neq p_j(\mathbf{X})$. Then, by Observation 4.37, and by also using property (4) if $k = p_j(\mathbf{X})$, k satisfies the conditions of line 2.

If $j \neq p_i(\mathbf{X})$, by Property (3.3), either $p_i(\mathbf{X})$ is non-envied or a neighbor of $p_i(\mathbf{X})$ is non-envied. In the latter case, that neighbor is defined as k which is not a neighbor of either i or j, otherwise a cycle of length less than 6 would exist, which is a contradiction. By Observation 4.37, k is a vertex satisfying the conditions of line 2. In the former case, we define $p_i(\mathbf{X})$ as k, which is not a neighbor of j for the same reason as above, and it holds $v_i(\mathit{UNPB}_i(\mathbf{X}) \cup X_k) = v_i(\mathit{UNPB}_i(\mathbf{X})) \geq v_i(S \cup X_k)$, for all $S \in UB(\mathbf{X})$ s.t. $S \cup X_k$ contains no parallel bundles. By Property (4) and Observation 4.37, k is a vertex satisfying the conditions of line 2 if no other bundle parallel to S has been given to k before; if j is non-envied, there is only one unallocated bundle between i, j (due to (4.1)), so this is true, and the statement of the claim holds. We proceed the analysis with the case that j is also envied and there may be two unallocated bundles between i and j. Let k_i be the k defined above, i.e., the non-envied vertex guaranteed by Property (3.3), by considering the envied vertex i. Let also k_j be similarly the non-envied vertex corresponding to vertex j; for the same reason as above, k_j satisfies the conditions of line 2. Then, k_i and k_j, that have distance at most 2 from i and j, respectively, should be different, otherwise there would be a cycle of length at most 5. Then, since there are at most two unallocated bundles between i, j (due to (4.1)), one may be allocated to k_i and the other to k_j, so the claim follows for that case, as well. □

In Claim 4.36 we showed that for any unallocated bundle of $\mathbf{X}$ (the allocation of Step 2) there is always a non-envied vertex satisfying the conditions of line 2, and so Algorithm 16 terminates in a full allocation. The condition in line 2 guarantees that whenever an unallocated bundle is assigned to a vertex k, no envy is caused towards k, and since k is non-envied, EFX is preserved. □

CHAPTER 5

CONCLUSION

We presented important results in the Fair Division literature, and basic algorithms. In this thesis, we pushed the state-of-the-art regarding one of the most important and academically interesting problems in Fair Division: the existence of EFX allocations. Following the work of Christodoulou et al. [20], we consider a graph structure that poses a restriction to valuation functions. Based on the notation of [20], we consider the $(2, \infty)$-bounded setting, which represents the case where each good is important for at most two agents. This can be seen as a multigraph setting where the vertices correspond to agents and the edges to goods, and each agent is interested only in goods/edges that are adjacent to him. Even if Christodoulou et al. [20] showed the existence of EFX allocations in simple graphs, i.e., the $(2, 1)$-bounded setting, even for general monotone valuation functions, the generalization to multigraphs poses extra challenges: It is well known [20, 40] that allocating edges to vertices other than their endpoints is inevitable, unless there are high restrictions on the graph structure. However, how to allocate those edges without violating EFX was the most challenging aspect in simple graphs settings [20], and becomes much harder in the setting of multigraphs.

As a result, we pose some restrictions on the structure of the multigraphs in order to address the above challenges. We handle each restriction differently only with respect to some initial partial orientation where each agent receives a bundle that is relevant to only one of its neighbors. This initial allocation is carefully constructed so that it provides crucial properties that are essential in order to manage to assign edges to vertices that are different from their endpoints by preserving the EFX property, and as a result derive a complete EFX allocation. We remark that after the construction of the initial allocation, we follow a unified approach, which is a generalization of the approach in [20] to multigraphs. This further indicates that hopefully our approach and ideas will be useful to push the state-of-the-art even further.

One important future direction is to drop the structure restrictions on the multigraphs and show the existence of EFX allocations in multigraphs with general valuations. Moreover, extending the existence of EFX allocations to hypergraphs, or other more general or incomparable (p, q)-bounded settings, could be a stepping stone towards the ultimate goal of showing EFX existence without any restriction on the setting; note that the setting of hyper-multigraphs corresponds to the unrestricted setting.

At last, we remark that our approach heavily relies on the cut-and-choose protocol

for general monotone valuation functions. It is known that finding such an allocation is hard (see [35, 25]), so our algorithms are not efficient. Therefore, another aspect to consider is the complexity of finding such allocations, and it may be reasonable to consider approximate EFX allocations that can be computed efficiently.

BIBLIOGRAPHY

[1] Mahyar Afshinmehr et al. *EFX Allocations and Orientations on Bipartite Multigraphs: A Complete Picture*. 2024. arXiv: 2410.17002 [cs.GT]. URL: https://arxiv.org/abs/2410.17002.

[2] Hannaneh Akrami et al. ``EFX: A Simpler Approach and an (Almost) Optimal Guarantee via Rainbow Cycle Number''. In: *Proceedings of the 24th ACM Conference on Economics and Computation, EC 2023, London, United Kingdom, July 9-12, 2023*. Ed. by Kevin Leyton-Brown, Jason D. Hartline, and Larry Samuelson. ACM, 2023, p. 61. DOI: 10.1145/3580507.3597799. URL: https://doi.org/10.1145/3580507.3597799.

[3] Georgios Amanatidis, Aris Filos-Ratsikas, and Alkmini Sgouritsa. ``Pushing the Frontier on Approximate EFX Allocations''. In: *Proceedings of the 25th ACM Conference on Economics and Computation, EC 2024, New Haven, United States of America, July 8 - 11, 2024*. 2024.

[4] Georgios Amanatidis, Evangelos Markakis, and Apostolos Ntokos. ``Multiple Birds with One Stone: Beating 1/2 for EFX and GMMS via Envy Cycle Elimination''. In: *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*. AAAI Press, 2020, pp. 1790–1797. DOI: 10.1609/AAAI.V34I02.5545. URL: https://doi.org/10.1609/aaai.v34i02.5545.

[5] Georgios Amanatidis, Evangelos Markakis, and Apostolos Ntokos. ``Multiple birds with one stone: Beating 1/2 for EFX and GMMS via envy cycle elimination''. In: *Theor. Comput. Sci.* 841 (2020), pp. 94–109. DOI: 10.1016/J.TCS.2020.07.006. URL: https://doi.org/10.1016/j.tcs.2020.07.006.

[6] Georgios Amanatidis et al. ``Maximum Nash welfare and other stories about EFX''. In: *Theor. Comput. Sci.* 863 (2021), pp. 69–85. DOI: 10.1016/J.TCS.2021.02.020. URL: https://doi.org/10.1016/j.tcs.2021.02.020.

[7] Haris Aziz and Simon Mackenzie. ``A discrete and bounded envy-free cake cutting protocol for four agents''. In: *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*. Ed. by Daniel Wichs and Yishay Mansour. ACM, 2016. DOI: 10.

1145/2897518.2897522. URL: https://doi.org/10.1145/2897518.2897522.

[8] Moshe Babaioff, Tomer Ezra, and Uriel Feige. ``Fair and Truthful Mechanisms for Dichotomous Valuations''. In: *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*. AAAI Press, 2021, pp. 5119–5126. DOI: 10.1609/AAAI.V35I6.16647. URL: https://doi.org/10.1609/aaai.v35i6.16647.

[9] Benjamin Aram Berendsohn, Simona Boyadzhiyska, and László Kozma. ``Fixed-Point Cycles and Approximate EFX Allocations''. In: *47th International Symposium on Mathematical Foundations of Computer Science, MFCS 2022, August 22-26, 2022, Vienna, Austria*. Ed. by Stefan Szeider, Robert Ganian, and Alexandra Silva. Vol. 241. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, 17:1–17:13. DOI: 10.4230/LIPICS.MFCS.2022.17. URL: https://doi.org/10.4230/LIPIcs.MFCS.2022.17.

[10] Ben Berger et al. ``Almost Full EFX Exists for Four Agents''. In: *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*. AAAI Press, 2022, pp. 4826–4833. DOI: 10.1609/AAAI.V36I5.20410. URL: https://doi.org/10.1609/aaai.v36i5.20410.

[11] Umang Bhaskar and Yeshwant Pandit. *EFX Allocations on Some Multi-graph Classes*. 2024. arXiv: 2412.06513 [cs.GT]. URL: https://arxiv.org/abs/2412.06513.

[12] Eric Budish. ``The combinatorial assignment problem: approximate competitive equilibrium from equal incomes''. In: *Proceedings of the Behavioral and Quantitative Game Theory - Conference on Future Directions, BQGT '10, Newport Beach, California, USA, May 14-16, 2010*. Ed. by Moshe Dror and Greys Sosic. ACM, 2010, 74:1. DOI: 10.1145/1807406.1807480. URL: https://doi.org/10.1145/1807406.1807480.

[13] Eric Budish and Estelle Cantillon. ``The Multi-Unit Assignment Problem: Theory and Evidence from Course Allocation at Harvard''. In: *American Economic Review* 102 (Jan. 2010). DOI: 10.1257/aer.102.5.2237.

[14] Ioannis Caragiannis, Nick Gravin, and Xin Huang. ``Envy-Freeness Up to Any Item with High Nash Welfare: The Virtue of Donating Items''. In: *Proceedings of the 2019 ACM Conference on Economics and Computation, EC 2019, Phoenix, AZ, USA, June 24-28, 2019*. Ed. by Anna R. Karlin, Nicole Immorlica, and Ramesh Johari. ACM, 2019, pp. 527–545. DOI: 10.1145/3328526.3329574. URL: https://doi.org/10.1145/3328526.3329574.

[15] Ioannis Caragiannis et al. ``The Unreasonable Fairness of Maximum Nash Welfare''. In: *ACM Trans. Economics and Comput.* 7.3 (2019), 12:1–12:32. DOI: 10.1145/3355902. URL: https://doi.org/10.1145/3355902.

[16] Hau Chan et al. ``Maximin-Aware Allocations of Indivisible Goods''. In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*. Ed. by Sarit Kraus. ijcai.org, 2019, pp. 137–143. DOI: 10.24963/IJCAI.2019/20. URL: https://doi.org/10.24963/ijcai.2019/20.

[17] Bhaskar Ray Chaudhury, Jugal Garg, and Kurt Mehlhorn. ``EFX Exists for Three Agents''. In: *J. ACM* 71.1 (2024), 4:1–4:27. DOI: 10.1145/3616009. URL: https://doi.org/10.1145/3616009.

[18] Bhaskar Ray Chaudhury et al. ``A Little Charity Guarantees Almost Envy-Freeness''. In: *SIAM J. Comput.* 50.4 (2021), pp. 1336–1358. DOI: 10.1137/20M1359134. URL: https://doi.org/10.1137/20M1359134.

[19] Bhaskar Ray Chaudhury et al. ``Improving EFX Guarantees through Rainbow Cycle Number''. In: *EC '21: The 22nd ACM Conference on Economics and Computation, Budapest, Hungary, July 18-23, 2021*. Ed. by Péter Biró, Shuchi Chawla, and Federico Echenique. ACM, 2021, pp. 310–311. DOI: 10.1145/3465456.3467605. URL: https://doi.org/10.1145/3465456.3467605.

[20] George Christodoulou et al. ``Fair allocation in graphs''. In: *Proceedings of the 24th ACM Conference on Economics and Computation, EC 2023, London, United Kingdom, July 9-12, 2023*. Ed. by Kevin Leyton-Brown, Jason D. Hartline, and Larry Samuelson. ACM, 2023, pp. 473–488. DOI: 10.1145/3580507.3597764. URL: https://doi.org/10.1145/3580507.3597764.

[21] Vasilis Christoforidis and Christodoulos Santorinaios. ``On the Pursuit of EFX for Chores: Non-existence and Approximations''. In: *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*. Ed. by Kate Larson. Main Track. International Joint Conferences on Artificial Intelligence Organization, Aug. 2024, pp. 2713–2721. DOI: 10.24963/ijcai.2024/300. URL: https://doi.org/10.24963/ijcai.2024/300.

[22] Argyrios Deligkas et al. *EF1 and EFX Orientations*. 2024. arXiv: 2409.13616 [cs.GT]. URL: https://arxiv.org/abs/2409.13616.

[23] Duncan Karl Foley. *Resource allocation and the public sector*. Yale University, 1966.

[24] G. Gamow and M. Stern. *Puzzle-math*. Viking Press, 1958. ISBN: 9780670583355. URL: https://books.google.gr/books?id=_vdytgAACAAJ.

[25] Paul W. Goldberg, Kasper Høgh, and Alexandros Hollender. ``The Frontier of Intractability for EFX with Two Agents''. In: *Algorithmic Game Theory - 16th International Symposium, SAGT 2023, Egham, UK, September 4-7, 2023, Proceedings*. Ed. by Argyrios Deligkas and Aris Filos-Ratsikas. Vol. 14238. Lecture Notes in Computer Science. Springer, 2023, pp. 290–307. DOI: 10.1007/978-3-031-43254-5_17. URL: https://doi.org/10.1007/978-3-031-43254-5%5C_17.

[26] P. Hall. ``On Representatives of Subsets''. In: *Journal of the London Mathematical Society* s1-10.1 (1935), pp. 26–30. DOI: https://doi.org/10.1112/jlms/s1-10.37.26. eprint: https://londmathsoc.onlinelibrary.wiley.com/doi/pdf/10.1112/jlms/s1-10.37.26. URL: https://londmathsoc.onlinelibrary.wiley.com/doi/abs/10.1112/jlms/s1-10.37.26.

[27] Hadi Hosseini et al. ``Fair and Efficient Allocations under Lexicographic Preferences". In: *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*. AAAI Press, 2021, pp. 5472–5480. DOI: 10.1609/AAAI.V35I6.16689. URL: https://doi.org/10.1609/aaai.v35i6.16689.

[28] Kevin Hsu. *EFX Orientations of Multigraphs*. 2024. arXiv: 2410.12039 [cs.GT]. URL: https://arxiv.org/abs/2410.12039.

[29] Shayan Chashm Jahan et al. ``Rainbow Cycle Number and EFX Allocations: (Almost) Closing the Gap". In: *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China*. ijcai.org, 2023, pp. 2572–2580. DOI: 10.24963/IJCAI.2023/286. URL: https://doi.org/10.24963/ijcai.2023/286.

[30] Alireza Kaviani, Masoud Seddighin, and AmirMohammad Shahrezaei. ``Almost Envy-free Allocation of Indivisible Goods: A Tale of Two Valuations". In: *Web and Internet Economics - 20th International Conference, WINE 2024, Edinburgh, United Kingdom, December 2-5*.

[31] Richard J. Lipton et al. ``On approximately fair allocations of indivisible goods". In: *Proceedings 5th ACM Conference on Electronic Commerce (EC-2004), New York, NY, USA, May 17-20, 2004*. Ed. by Jack S. Breese, Joan Feigenbaum, and Margo I. Seltzer. ACM, 2004, pp. 125–131. DOI: 10.1145/988772.988792. URL: https://doi.org/10.1145/988772.988792.

[32] Ryoga Mahara. ``Extension of Additive Valuations to General Valuations on the Existence of EFX". In: *Math. Oper. Res.* 49.2 (2024), pp. 1263–1277. DOI: 10.1287/MOOR.2022.0044. URL: https://doi.org/10.1287/moor.2022.0044.

[33] Ryoga Mahara. ``Extension of Additive Valuations to General Valuations on the Existence of EFX". In: *Math. Oper. Res.* 49.2 (2024), pp. 1263–1277. DOI: 10.1287/MOOR.2022.0044. URL: https://doi.org/10.1287/moor.2022.0044.

[34] Evangelos Markakis and Christodoulos Santorinaios. ``Improved EFX Approximation Guarantees under Ordinal-based Assumptions". In: *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2023, London, United Kingdom, 29 May 2023 - 2 June 2023*. Ed. by Noa Agmon et al. ACM, 2023, pp. 591–599. DOI: 10.5555/3545946.3598689. URL: https://dl.acm.org/doi/10.5555/3545946.3598689.

[35] Benjamin Plaut and Tim Roughgarden. ``Almost Envy-Freeness with General Valuations". In: *SIAM J. Discret. Math.* 34.2 (2020), pp. 1039–1068. DOI: 10.1137/19M124397X. URL: https://doi.org/10.1137/19M124397X.

[36] Ariel D. Procaccia. ``An answer to fair division's most enigmatic question: technical perspective". In: *Commun. ACM* 63.4 (2020), p. 118. DOI: 10.1145/3382131. URL: https://doi.org/10.1145/3382131.

[37] H. Steihaus. ``The problem of fair division". In: *Econometrica* 16 (1948), pp. 101–104.

[38] Vishwa Prakash H. V. et al. *EFX Exists for Three Types of Agents*. 2024. arXiv: 2410.13580 [cs.GT]. URL: https://arxiv.org/abs/2410.13580.

[39] Hal R Varian. ``Equity, envy, and efficiency". In: *Journal of Economic Theory* 9.1 (1974), pp. 63–91. ISSN: 0022-0531. DOI: https://doi.org/10.1016/0022-0531(74)90075-1. URL: https://www.sciencedirect.com/science/article/pii/0022053174900751.

[40] Jinghan A Zeng and Ruta Mehta. ``On the structure of envy-free orientations on graphs". In: *CoRR* abs/2404.13527 (2024). DOI: 10.48550/ARXIV.2404.13527. arXiv: 2404.13527. URL: https://doi.org/10.48550/arXiv.2404.13527.

[41] Yu Zhou et al. ``A Complete Landscape of EFX Allocations on Graphs: Goods, Chores and Mixed Manna". In: *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*. Ed. by Kate Larson. Main Track. Jeju, Korea: International Joint Conferences on Artificial Intelligence Organization, Aug. 2024, pp. 3049–3056. DOI: 10.24963/ijcai.2024/338. URL: https://doi.org/10.24963/ijcai.2024/338.