Harmonious Balanced Partitioning of a Network of Agents

Pulkit Agarwal[1] , **Harshvardhan Agarwal**[1] , **Vaibhav Raj**[1] and **Swaprava Nath**[1]
[1]Indian Institute of Technology Bombay, Mumbai, India,
{200050113,200050050,200050148,swaprava}@iitb.ac.in

Abstract

We consider the problem of balanced partitioning, i.e., dividing n agents into k groups of almost equal size ($\lfloor n/k \rfloor$ or $\lceil n/k \rceil$), where the agents form a friendship network, ensuring various fairness and efficiency criteria. The utility of an agent is the count of its friends in the same group as itself. When partitions into two groups are considered, we show that approximate envy-freeness related to the maximum degree of the graph can be obtained via a linear-time algorithm for arbitrary graphs. We also show that envy-freeness and core properties can be extended along with Pareto optimality in arbitrary graphs for such partitions. We then concentrate on the case of grid graphs having nodes on the 2D integer lattice, and demonstrate the impossibility of perfect envy-freeness. However, weaker guarantees like envy-freeness up to two friends are achievable for balanced $k-$partitions in a computationally efficient manner. We show that certain such balanced partitions belong to an exact and an approximate core when considering balanced $2-$partitions.

1 Introduction

Consider a school conducting a large sports event where multiple teams formed out of its students contest for an overall prize. In such competitions, the players are usually divided into (almost) equal-sized teams at random. However, a crucial component of team sports is coordination between teammates which demands a feeling of friendship. This constraint adds a layer of complexity to the problem.

This paper works with a model that represents situations like the above. Suppose we have symmetric friendship relations among n agents, depicted by a network. The problem looks to divide these n agents into k groups of nearly equal size, with the utility of an agent defined as the number of friends of that agent in its own group. We call such a division a balanced $k-$partition (BP$-k$). In the above example, we see that the higher utility of an agent in a balanced partition is informally equivalent to the agent being able to coordinate better with their teammates.

This problem of partitioning agents in an undirected friendship network into groups of almost equal size was introduced by Li *et al.* (2023). They posed it as an extension of the stable roommates problem (Irving, 1985), which requires an even number of agents and limits group size to exactly *two* agents. The problem usually considers preference profiles for each agent over all other agents. However, under our formulation, we only look at binary preferences (friend or not), which are useful in practice where it may not be possible to obtain an entire preference profile list. Numerous works have also studied the three-dimensional generalization (Huang, 2007; Arkin *et al.*, 2009; McKay and Manlove, 2021; Cseh *et al.*, 2022; McKay, 2022), where groups of *three* have to be formed.

Envy-freeness (EF) and efficiency guarantees for balanced $k-$partitions of graphs are natural questions that can arise in this context. In this paper, we first look at general graphs and provide EF guarantees w.r.t. the maximum degree of the graph. Li *et al.* (2023) also studied trees extensively, and suggested looking at planar graphs and graphs with bounded degrees. However, structures like trees are much less common for friendship graphs. We study grid graphs instead and prove stronger guarantees. Such graphs are often studied in graph bisection problems due to their importance in real-world scenarios (Feldmann, 2012). For instance, a potential use-case arises when we represent a city as a grid. Considering houses as agents that occupy the grid coordinates, an obvious (and simplified) *friendship relation* is that of *neighborhood*. In such cases, agents (households) would prefer to be put alongside their neighbors when dividing households into groups for societal activities within the locality (e.g., for cleanliness drives or the conduction of ceremonies).

1.1 Our Contributions

Our work builds upon Li *et al.* (2023). We refine the concept of envy, and investigate balanced partitions in graphs that adhere to specific fairness criteria, exploring Pareto optimality and approximations of envy-freeness and core.

For any graph $G = (V, E)$ with $|V| = n$, we introduce a linear-time algorithm for constructing a BP-2 that satisfies EF$-(\max(\Delta(G) - 2, 2))$ for any arbitrary graph. This solves the open problem raised by Li *et al.* (2023) regarding the existence of an EF-2 BP-2 for all graphs with maximum degree at most 4. For graphs with $\Delta(G) \ll \sqrt{n}$, this is also a considerable improvement over the best-known result on the existence of an EF$-(\mathcal{O}(\sqrt{n}))$ BP-2 (Li *et al.*, 2023). In particular, the two results together guarantee an

	Results	Guarantee
General Graphs	$\forall G = (V, E)$, $\exists$ an EF$-(\max(\Delta(G) - 2, 2))$ BP-2 (Theorem 1)	$\mathcal{O}(\lvert V\rvert + \lvert E\rvert)$ algorithm (Algorithm 3)
	$\forall G = (V, E)$ and $\forall\, r \geqslant 0, \alpha \geqslant 1, \beta \geqslant 0$, 1. $\exists$ EF$-r$ BP-2 $\implies$ $\exists$ (EF$-r$ + PO) BP-2 2. $\exists$ $(\alpha, \beta)-$ core BP-2 $\implies$ $\exists$ $((\alpha, \beta)-$ core + PO) BP-2 3. $\exists$ $((\alpha, \beta)-$ core + EF$-r$) BP-2 $\implies$ $\exists$ $((\alpha, \beta)-$ core + EF$-r$ + PO) BP-2 (Theorems 2 and 3)	Existence
Grid Graphs	$\forall G = (V, E) \in$ GrG, $\exists$ an EF-2 BP$-k$, $\forall k \geqslant 2$ (Theorem 4)	$\mathcal{O}(\lvert V\rvert \log(\lvert V\rvert))$ algorithm (Algorithm 4)
	$\forall G = (V, E) \in$ GrG, $\exists$ an EF-2 BP-2 in $(1, 1) -$ core (Theorem 5)	$\mathcal{O}(\lvert V\rvert \log(\lvert V\rvert))$ algorithm (Algorithm 4)
	$\forall G = (V, E) \in$ GrG, $\exists$ a BP-2 in $(1, 0) -$ core (Theorem 6)	Existence (Algorithm 5)

Table 1: Summary of the results.

EF$-(\mathcal{O}(\min(\Delta(G), \sqrt{n})))$ BP-2. Additionally, we demonstrate the existence of a Pareto optimal (PO) BP-2 without compromising its fairness (EF) and stability (core) criteria.

For the special class of grid graphs, we show that an EF-2 BP$-k$ can always be found via an efficient algorithm. For $k = 2$, the same algorithm returns a BP-2 that satisfies the additional guarantee of being in the $(1, 1) -$ core. Furthermore, we prove that for grid graphs, a BP-2 in $(1, 0) -$ core always exists. Table 1 highlights our major contributions.

1.2 Related Works

In the literature, most works have focused on finding minimum-cut balanced and unbalanced partitions (Kernighan and Lin, 1970; Garey and Johnson, 1990; Andreev and Räcke, 2004; Sotirov, 2018). Li *et al.* (2023) used min-cut partitions for a number of their results on the existence of balanced partitions in the core. Recent studies on balanced $2-$partitioning of graphs (or *graph bisections*) have also looked at computational complexity and approximations of partitions in which every node has at least H neighbors in its own group (Bazgan *et al.*, 2010; Behrens *et al.*, 2022; Minzer *et al.*, 2023). If H is not constant, and taken as half of the degree of each node, this reduces to Nash stability. Unlike these works, we demonstrate the connection of this notion with envy-freeness and come up with an efficient algorithm for an approximation of the $H = 1$ case, which solves the open problem of the existence of an EF-2 BP-2 for all graphs with $\Delta(G) \leqslant 4$.

A different dimension along which existing literature has varied is finding partitions with bounded group sizes. Levinger *et al.* (2023) demonstrated the NP-completeness of the utility maximization problem in this setup, proposed a poly-time approximation algorithm for the task, and further explored stability through the notion of the core. Boehmer and Elkind (2020) formulated the group-partitioning task from the viewpoint of diversity preferences, where agents belong to exactly one of two groups and each agent has a weak preference order on the composition of their group. Our work differs from these tasks both due to the complexity of forming a fixed-size coalition, and the presence of a direct binary preference order over all the participants.

The most under-researched concept in balanced partitioning is that of Pareto optimality. Aziz *et al.* (2013) introduced Pareto optimality in coalition formation games for the first time, showing various results on their intractability in different game classes, while Li *et al.* (2023) only showed that some of their BP$-k$ algorithms are not PO. To the best of our knowledge, this paper is the first to look at the existence of PO bisections along with other fairness guarantees.

2 Preliminaries

Define $[\ell] = \{1, 2, \ldots, \ell\}$. Consider a group of agents (denoted by the set V with $\lvert V\rvert = n$) connected over an undirected graph $G = (V, E)$, where each edge $(i, j) \in E$ represents that i and j are neighbors in G. The degree of node i in G is denoted by $d_G(i)$, and $\Delta(G) = \max_{i \in V} d_G(i)$ denotes the maximum degree of the graph G. The set of neighbors of agent i in G is denoted by $N_G(i)$. The subgraph of G induced by a set of vertices $S \subseteq V$ is denoted by $G[S]$.

Definition 1 ($k-$partition)**.** *A $k-$partition of a graph $G = (V, E)$ is a set of k mutually exclusive and exhaustive subsets of V. Formally, it is the collection $Z = \{Z_1, Z_2, \ldots, Z_k\}$ s.t.*

$$Z_\ell \subseteq V, \forall \ell \in [k], Z_{\ell_1} \cap Z_{\ell_2} = \emptyset, \forall \ell_1 \neq \ell_2, \text{ and } \bigcup_{\ell \in [k]} Z_\ell = V$$

Definition 2 (Balanced $k-$partition)**.** *A* balanced $k-$partition $Z = \{Z_1, Z_2, \ldots, Z_k\}$ *of a graph $G = (V, E)$ is a $k-$partition with $\lvert Z_\ell\rvert \in \left\{\left\lfloor \frac{n}{k} \right\rfloor, \left\lceil \frac{n}{k} \right\rceil\right\}, \forall \ell \in [k]$.*

We will use the shorthand BP$-k$ to mention a balanced $k-$partition. The neighbors of agent i in a $k-$partition Z is given by $N_G^Z(i) = \{j \in N_G(i) : i, j \in Z_\ell, \text{ for some } \ell \in [k]\}$. The *utility* of agent i in a $k-$partition Z is defined as $u_i^G(Z) = \lvert N_G^Z(i)\rvert$, i.e., the number of neighbors of i in G that belong to the same subset as i in the partition Z. Wherever it is clear from context, we will drop G from every notation that uses it. The subset in which agent i be-

longs in a balanced $k-$partition Z is denoted by $Z^{-1}(i)$, i.e., $Z^{-1}(i) = \ell,$ s.t. $i \in Z_\ell$.

Define a *swapping partition* of a given $k-$partition Z between two agents i and j as the new $k-$partition Z' such that

$$\begin{aligned} Z'_{Z^{-1}(i)} &= Z_{Z^{-1}(i)} \setminus \{i\} \cup \{j\} \\ Z'_{Z^{-1}(j)} &= Z_{Z^{-1}(j)} \setminus \{j\} \cup \{i\} \\ Z'_{Z^{-1}(p)} &= Z_{Z^{-1}(p)}\ ,\ \forall p \in V \setminus \{i,j\}. \end{aligned} \tag{1}$$

In other words, the swapping partition Z' swaps the subsets in which agents i and j belonged in Z. We will denote the swapping partition Z' with the notation $\textbf{swap}(Z,i,j)$. It is easy to see that, $\textbf{swap}(Z,i,j)$ is non-trivial only when $Z^{-1}(i) \neq Z^{-1}(j)$, i.e., i and j are in different subsets of Z.

Define a *transfer partition* of a given $k-$partition Z for agent i to another subset $Z_\ell, \ell \neq Z^{-1}(i), |Z_\ell| < |Z_{Z^{-1}(i)}|$ as the new $k-$partition Z' such that

$$\begin{aligned} Z'_{Z^{-1}(i)} &= Z_{Z^{-1}(i)} \setminus \{i\} \\ Z'_\ell &= Z_\ell \cup \{i\} \\ Z'_p &= Z_p\ ,\ \forall p \neq Z^{-1}(i), \ell. \end{aligned} \tag{2}$$

In simpler terms, the transfer moves an agent i from its original larger subset to a smaller subset. We will denote such a transferred partition Z' with the notation $\textbf{tran}(Z,i,\ell)$.

We start off with an observation, used in defining envy.

Observation 1. *Given a pair of agents (i,j) and a* BP$-k$ *Z,* $\textbf{swap}(Z,i,j)$ *is also a* BP$-k$*. Also, if $|Z_{Z^{-1}(j)}| < |Z_{Z^{-1}(i)}|$, then* $\textbf{tran}(Z,i,Z^{-1}(j))$ *is also a* BP$-k$*, and*

$$u_i\left(\textbf{tran}(Z,i,Z^{-1}(j))\right) \geqslant u_i\left(\textbf{swap}(Z,i,j)\right) \tag{3}$$

Proof. Note that both **swap** and **tran** are $k-$partitions. Here, **swap** is a BP$-k$ since the sizes of all subsets in the new partition remain the same. And **tran** is a BP$-k$ as we transfer node i from a larger subset of size $\lceil n/k \rceil$ to a smaller subset of size $\lfloor n/k \rfloor$, which simply exchanges the sizes of these subsets. Finally, taking $Z^{-1}(j) = \ell$,

$$u_i\left(\textbf{tran}(Z,i,\ell)\right) = \begin{cases} u_i\left(\textbf{swap}(Z,i,j)\right) + 1 & ;\ (i,j) \in E \\ u_i\left(\textbf{swap}(Z,i,j)\right) & ;\ (i,j) \notin E \end{cases}$$

since j is in i's subset after **tran**, but not after **swap**. □

We can now define envy and envy-freeness as follows.

Definition 3 (Envy)**.** *In a* BP$-k$ *X, an agent i has an envy $r \geqslant 0$ towards another agent j if*

1. *case $|X_{\ell_j}| < |X_{\ell_i}|$: $u_i(\textbf{tran}(X,i,\ell_j)) - u_i(X) = r$.*
2. *case $|X_{\ell_j}| \geqslant |X_{\ell_i}|$: $u_i(\textbf{swap}(X,i,j)) - u_i(X) = r$.*

where $\ell_i = X^{-1}(i)$ and $\ell_j = X^{-1}(j)$.

The definition says that if agent i thinks that a swap or transfer with another agent outside its own subset in a partition can increase its utility, then it is envious of that agent. Here, if **tran** is possible, then it is preferred over **swap**, as that yields a weakly better utility for the agent (Observation 1).

Definition 4 (EF$-r$)**.** *For $r \geqslant 0$, a* BP$-k$ *X is said to be* envy-free up to r *(*EF$-r$*), if for every pair of agents $i,j \in V$, i envies j by at most r.*

We also make a useful observation about envy below.

Observation 2. *If agent i has an envy $r > 0$ towards some agent j in* BP$-k$ *X, then $X^{-1}(j) \neq X^{-1}(i)$ and i has at least r neighbors in $X_{X^{-1}(j)}$.*

Proof. Since agent i has an envy $r > 0$ towards agent j in BP$-k$ X, utility of agent i increases from its current utility $u_i(X)$ by r. This increase can either occur by a **tran** or a **swap** operation by i from $X_{X^{-1}(i)}$ to $X_{X^{-1}(j)}$ (and these two subsets must be different). In both cases, it is necessary that there exist at least r neighbors of i in $X_{X^{-1}(j)}$. □

Definition 5 (Core)**.** *For $\alpha \geqslant 1, \beta \geqslant 0$, a* BP$-k$ *X is said to be in $(\alpha,\beta)-$* core *if there does not exist another* BP$-k$ *X' and an index $\ell \in [k]$ s.t. $\forall i \in X'_\ell\ ,\ u_i(X') > \alpha \cdot u_i(X) + \beta$.*

This definition only requires the utility of agents in X'_ℓ to increase. If for some BP$-k$ X such a coalition X'_ℓ exists, we call X'_ℓ an $(\alpha,\beta)-$*blocking coalition* of BP$-k$ X. Wherever clear from context, we simply call X'_ℓ a blocking coalition.

A BP$-k$ is said to be in the *core* for graph G if it satisfies $(1,0)-$core property as defined above. For $k=2$, the existence of core means that there is no other BP-2 $(S, V \setminus S)$, such that the utility of all nodes in S strictly increases in this new partition. Also note that $(1,0)-$core is stronger than $(\alpha,\beta)-$core for any other choice of $\alpha \geqslant 1, \beta \geqslant 0$.

Definition 6 (Pareto Optimal)**.** *A* BP$-k$ *X is called* Pareto optimal *(*PO*), if there does not exist another* BP$-k$ *X' such that for all nodes $i \in V$, we have $u_i(X') \geqslant u_i(X)$, and the strict inequality holds for at least one node $j \in V$.*

If X is not Pareto optimal, then the BP$-k$ X' is called a *Pareto improvement* to partition X.

2.1 Graphs of interest

Since this paper considers balanced partitions of agents on a graph, in this section, we provide a brief review of certain types of graphs and their properties that will be used later in the paper. We use the standard graph-theoretic terminology. Let $\overline{G}$ be the complement of graph G. We denote by K_n, C_n, and P_n, the complete undirected graph on n vertices, a simple cycle with n vertices, and a path graph with n vertices respectively. Also, we define the *comb graph*[1] on $2n$ vertices as $P_n \bigodot K_1$, i.e., the graph constructed by connecting n vertices in a path, each of which is connected to a pendant edge. This graph consists of $2n-1$ edges, and is denoted by $\textbf{comb}_n$.

Biconnected graphs. A *biconnected graph* $G = (V,E)$ with $|V| > 1$ is a connected graph for which the subgraph $G[V \setminus \{v\}]$ remains connected for all $v \in V$.

Definition 7 ($st-$Numbering)**.** *An $st-$*numbering *of $G = (V,E)$ with $|V| = n > 1$ and two vertices $s,t \in V$ is a mapping from V to $N = [n]$, such that the source s is labeled 1, the sink t is labeled n, and every vertex $i = 2,\ldots,n-1$ is adjacent both to some vertex $h < i$ and to some vertex $j > i$.*

It is well known that for any biconnected graph with any arbitrarily chosen source s and sink t, an $st-$numbering exists and can be found in linear time (Ebert, 1983). We treat

[1] The name of this graph comes from the fact that it can be drawn in the form of a comb.

$S = \textbf{st-numbering}(G,\ \text{source},\ \text{sink})$ as a procedure that returns the above mapping, i.e. $S(v)$ denotes the label of some $v \in V$ in this st-numbering. When the procedure is called without a sink argument, it is understood that any arbitrary vertex (different from the source) in G can be used as the sink. The utility of st-numbering comes from the following observation.

Observation 3. *Consider a biconnected graph $G = (V, E)$ with an arbitrary node $v \in V$ and $S = \textbf{st-numbering}(G, source = v)$. For any positive integer k with $V_k = \{v \in V \mid S(v) \leqslant k\}$, subgraphs $G[V_k]$ and $G[V \setminus V_k]$ are connected graphs.*

Proof. We show that $G[V_k]$ is connected for all $k \geqslant 1$, and the result for the other subgraph follows simply by reversing the source and the sink in the st-numbering of G. By Definition 7, any node in V_k labeled $i > 1$ also has a neighbor in V_k with a smaller label $h < i$. This ensures that every node labeled $i > 1$ has a path joining it to the node labeled 1. Thus, $G[V_k]$ forms a connected graph. □

A connected graph that is not a biconnected graph will have *cut vertices*, i.e. a vertex whose removal causes the graph to become disconnected. A *biconnected component* is a maximal biconnected subgraph in a graph.

Definition 8 (Block-Cut Tree)**.** *Any connected graph decomposes into a tree with cut vertices and biconnected components as vertices, called the* block-cut tree *of the graph, where the blocks are attached to shared cut vertices.*

Hopcroft and Tarjan (1973) gave a linear-time algorithm to find the block-cut tree for any connected graph.

Definition 9 (Leafy-Cut Vertex)**.** *A* leafy-cut vertex *in a rooted block-cut tree is a cut vertex that does not have any other cut vertex as its descendant.*

A leafy-cut vertex z has a non-zero number of descendants which can be partitioned into sets of vertices $b_1, \ldots, b_k$ ($k \geqslant 1$) such that $\forall i \in [k],\ |b_i| > 0$ and $G[b_i \cup \{z\}]$ is a biconnected graph. We refer to each b_i as a *biconnected block*.

Grid graphs. A grid graph is a collection of nodes on the integer 2D coordinate lattice with possible edges between them only if the nodes are adjacently placed on the lattice.

Definition 10 (Grid Graph)**.** *$G = (V, E)$ is a grid graph if we can create mappings $X : V \to \mathbb{Z}$ and $Y : V \to \mathbb{Z}$ such that for each pair of nodes $i, j \in V$, where $(X(i), Y(i)) \neq (X(j), Y(j))$,*

$$[(i, j) \in E] \implies [|X(i) - X(j)| + |Y(i) - Y(j)| = 1]\,,$$

where X and Y are the mappings to the x and y-coordinates of the nodes on the lattice.

We will denote the set of grid graphs with `GrG`. We are now ready to present the main results of this paper.

3 Main Results: General Graphs

We first look at envy-freeness and Pareto optimality properties for arbitrary graphs. For both these properties, we will need a lemma that formally shows the necessary and sufficient conditions for the existence of an EF$-r$ BP-2.

Lemma 1. *A* BP-2 *X in $G = (V, E)$ is* EF$-r$

1. *if*
$$|N^X(i)| \geqslant \frac{d(i) - r}{2},\ \forall i \in V,\ \textit{and,} \tag{4}$$
2. *only if*
$$|N^X(i)| \geqslant \frac{d(i) - r - 1}{2},\ \forall i \in V. \tag{5}$$

We state the conditions 1 and 2 separately since both of them will be used for proving the upcoming results.

Proof. (Condition 1: sufficiency): Let X be a BP-2. Suppose X is not EF$-r$. Hence, $\exists i, j \in V$ such that i has an envy $> r$ towards j. Case 1: $|X_{X^{-1}(i)}| > |X_{X^{-1}(j)}|$: here the violation of EF$-r$ implies $u_i(\textbf{tran}(X, i, X^{-1}(j))) - u_i(X) > r$. This implies, $(d(i) - |N^X(i)|) - |N^X(i)| > r$. This violates the condition in Equation (4). Case 2: $|X_{X^{-1}(i)}| \leqslant |X_{X^{-1}(j)}|$: here the violation of EF$-r$ implies $u_i(\textbf{swap}(X, i, j)) - u_i(X) > r$. Here, two subcases can occur, (a) i and j are neighbors in G: then $(d(i) - |N^X(i)| - 1) - |N^X(i)| > r$ (the additional -1 appears since j also changes its subset in a **swap**), and (b) i and j are not neighbors in G: then $(d(i) - |N^X(i)|) - |N^X(i)| > r$. Both these subcases violate the condition in Equation (4).

(Condition 2: necessity): Suppose X is an EF$-r$ BP-2. This implies that $\forall i, j \in V$, i has an envy $\leqslant r$ towards j. Now consider the cases 1 and 2 above. Carrying out similar calculations we get that $(d(i) - |N^X(i)|) - |N^X(i)| \leqslant r$ for case 1 and the second subcase of case 2. But for the first subcase of case 2 (where $(i, j) \in E$), we get $(d(i) - |N^X(i)| - 1) - |N^X(i)| \leqslant r$. Hence the condition that satisfies *all* these conditions (and hence necessary) is given by Equation (5). □

We are now ready to present the results.

3.1 Envy-Freeness

Theorem 1. *For every graph G, there exists a* BP-2 *that is* EF$-(\max(\Delta(G)-2, 2))$*, which can be found in linear time.*[2]

We first derive a corollary of Lemma 1 to get a sufficient condition for Theorem 1.

Lemma 2. *A* BP-2 *X of $G = (V, E)$ is* EF$-(\max(\Delta - 2, 2))$ *if for all nodes $i \in V$, at least one of the following two conditions holds: (1) $d(i) \leqslant 2$, or (2) $|N^X(i)| \geqslant 1$.*

Proof. For $d(i) \leqslant 2$, we know that $\frac{d(i)-2}{2} \leqslant 0 \leqslant |N^X(i)|$. Otherwise $|N^X(i)| \geqslant 1 \geqslant \frac{d(i)-(\Delta-2)}{2}$. Thus, $\forall i \in V, |N^X(i)| \geqslant \frac{d(i)-\max(\Delta-2,2)}{2}$. Following condition 1 of Lemma 1, we conclude that X is EF$-(\max(\Delta - 2, 2))$. □

By Lemma 2, to obtain an EF$-(\max(\Delta - 2, 2))$ BP-2, it suffices to ensure that every vertex with degree > 2 has at least 1 neighbor in the same group. For simplicity, we refer to the two subsets of the partition by the colors red and blue. We construct the BP-2 sequentially, first for biconnected graphs, then for connected graphs, and finally for arbitrary graphs.

[2] From now on, we will drop G from our notations.

Algorithm 1 BICONNPART: Biconnected Partition

Input: Biconnected graph $G = (V, E)$, integers r, b
Require: $r, b > 1;\ r + b = |V|$
Output: $X = \{\mathcal{R}, \mathcal{B}\}, |\mathcal{R}| = r$
1: $y,\ z \leftarrow$ arbitrary nodes in $V,\ y \neq z$;
2: $S \leftarrow$ **st-numbering**$(G,\ \text{source} = y,\ \text{sink} = z)$
3: $\mathcal{R} \leftarrow \{v \in V \mid S(v) \leqslant r\},\ \mathcal{B} \leftarrow V \setminus \mathcal{R}$
4: **return** $X = \{\mathcal{R},\ \mathcal{B}\}$

Lemma 3. *For any biconnected graph $G = (V, E)$ and $r, b > 1$ with $r + b = |V|$, Algorithm 1 returns a $2-$partition $X = (\mathcal{R}, \mathcal{B})$ with $|\mathcal{R}| = r, |\mathcal{B}| = b$ s.t. $|N^X(i)| \geqslant 1,\ \forall i \in V$.*

Proof. Algorithm 1 adds the first r vertices in an st-numbering of G to set $\mathcal{R}$. Using Observation 3, $G[\mathcal{R}]$ and $G[V \setminus \mathcal{R}]$ are connected graphs, which ensures that every vertex in V has at least 1 neighbor in the same group (since $|\mathcal{R}|, |V \setminus \mathcal{R}| > 1$). □

For connected graphs $G = (V, E)$, we use their rooted block-cut tree $\mathcal{T}$ to color the graph appropriately in Algorithm 3. We first color all vertices in V blue, and then change the color of $r > 1$ vertices to red to get a desired $2-$partition X. Here, in each iteration (starting at Line 6 of Algorithm 3), we color all the biconnected children blocks of some leafy-cut vertex red (Line 12), till r nodes have been colored red. Lemma 4 deals with the case when the remaining nodes, which are to be colored red, fit within the subtree of a leafy-cut vertex z (Line 17). In this case, we call the SELECTPAIRSFIRST procedure given in Algorithm 2, which colors some of the nodes in this subtree of z red, while satisfying Equation (6) below for all nodes. The coloring is done by using an $st-$numbering of the biconnected blocks with the source as node z.

Lemma 4. *For a connected graph $G = (V, E)$ and its rooted block-cut tree $\mathcal{T}$, let z be a leafy-cut vertex in $\mathcal{T}$ with biconnected blocks $\{b_i\}_{i=1}^k$ as its children. Then for any $1 < r \leqslant \Sigma_{i=1}^k |b_i|,\ b > 1,\ r + b = |V|,\ \exists\ 2-$partition $X = (\mathcal{R}, \mathcal{B})$ with $|\mathcal{R}| = r, |\mathcal{B}| = b$ such that $\forall\ i \in V$,*

$$\text{either } d(i) = 1, \text{ or } |N^X(i)| \geqslant 1 \tag{6}$$

Also, for any $j \in [k]$,

$$|b_j| = 1 \text{ and } b_j \subseteq \mathcal{B} \implies z \in \mathcal{B} \tag{7}$$

This $2-$partition can be found in $\mathcal{O}(|V| + |E|)$ time using the SELECTPAIRSFIRST $\left(G, \mathcal{T}, z, \{b_i\}_{i=1}^k, r, b\right)$ *procedure, given in Algorithm 2.*

Proof. We use Algorithm 2 to show the existence of the desired $2-$partition. In particular, we show that if $\mathcal{R}$ is the output of Algorithm 2, and $\mathcal{B} = V \setminus \mathcal{R},\ X = (\mathcal{R}, \mathcal{B})$, then Equation (6) holds for all $i \in V$, and Equation (7) holds for all $j \in [k]$. We first color the complete graph blue, and then change the color of r vertices to red.

Consider an arbitrary $j \in [k]$. Let S_j denote the $st-$numbering of $G[b_j \cup \{z\}]$ with z as the source vertex. Also, let LAST_BLUE$[j]$ denote the last blue node in the coloring of block b_j in order of its st-numbering S_j. In other words, at any instant, all nodes in $\{v \in b_j \mid S_j(v) \leqslant$ LAST_BLUE$[j]\}$ are colored blue. Thus, decreasing the value of LAST_BLUE$[j]$ is equivalent to coloring some nodes in b_j red. Let $C = \cup_{i=1}^k b_i$. We first show that every $v \in C$ satisfies Equation (6). Consider the following two cases for r.

1. r is even: We start with Phase 1 (Line 13). In each iteration, we color an even number of nodes (denoted by `reqd` in Line 18) from b_j red for some $j \in [k]$, i.e. the last `reqd` nodes in S_j are colored red. Let $b_R = \{v \in b_j \mid S_j(v) >$ LAST_BLUE$[j]\}$, which is the set of red-colored nodes in b_j. By Observation 3, $G[b_R]$ and $G[(b_j \cup \{z\}) \setminus b_R]$ are connected graphs. Note that

Algorithm 2 SELECTPAIRSFIRST

Input:
Connected graph $G = (V, E)$
Block-cut tree $\mathcal{T}$, leafy-cut vertex z of $\mathcal{T}$
List of children biconnected blocks $\{b_i\}_{i=1}^k$, integers r, b
Require: $1 < r \leqslant \Sigma_{i=1}^k |b_i|,\ b > 1,\ r + b = |V|$
Output: $\mathcal{R} \subseteq V$ such that $|\mathcal{R}| = r$
1: $S_j \leftarrow$ **st-numbering**$(G[b_j \cup \{z\}],\ \text{source} = z),\ \forall j \in [k]$
2: LAST_BLUE$[j] \leftarrow |b_j|,\ \forall j \in [k]$
3: // Make r even by allotting an odd number of nodes to $\mathcal{R}$
4: **if** r is odd **then**
5: **if** $\exists\ j \in [k]$ s.t. $|b_j| = 1$ **then**
6: LAST_BLUE$[j] \leftarrow 0$; $r \leftarrow r - 1$
7: **else if** $\exists\ j \in [k]$ s.t. $|b_j| \geqslant 3$ **then**
8: LAST_BLUE$[j] \leftarrow |b_j| - 3$; $r \leftarrow r - 3$
9: **else if** $|b_j| = 2$ for all $j \in [k]$ **then**
10: $\mathcal{R} \leftarrow \bigcup_{i=1}^{\lfloor r/2 \rfloor} b_i \cup \{z\}$; **return** $\mathcal{R}$
11: **end if**
12: **end if**
13: **PHASE I: Color Pairs**
14: idx $\leftarrow$ 1 // First loop to color pairs red
15: **while** $r > 0$ & idx $\leqslant k$ **do**
16: remaining $\leftarrow$ LAST_BLUE[idx]
17: // Note that 'reqd' is even, hence pairs
18: reqd $\leftarrow \min\left(r,\ 2\left\lfloor \text{remaining}/2 \right\rfloor\right)$
19: LAST_BLUE[idx] $\leftarrow$ remaining $-$ reqd
20: $r \leftarrow r -$ reqd; idx $\leftarrow$ idx $+ 1$
21: **end while**
22: **PHASE II: Color Remainders**
23: idx $\leftarrow$ 1 // Second loop to color remaining odd blocks
24: **while** $r > 0$ & idx $\leqslant k$ **do**
25: remaining $\leftarrow$ LAST_BLUE[idx]
26: // Note that 'reqd' will be either 0 or 1
27: reqd $\leftarrow \min(r, \text{remaining})$
28: LAST_BLUE[idx] $\leftarrow$ remaining $-$ reqd
29: $r \leftarrow r -$ reqd; idx $\leftarrow$ idx $+ 1$
30: **end while**
31: $\mathcal{R} \leftarrow \emptyset$; idx $\leftarrow$ 1 // Adding assigned red nodes to $\mathcal{R}$
32: **while** idx $\leqslant k$ **do**
33: $\mathcal{R} \leftarrow \mathcal{R} \cup \{v \in b_{\text{idx}} \mid S_{\text{idx}}(v) >$ LAST_BLUE[idx]$\}$
34: **end while**
35: **return** $\mathcal{R}$

$|b_R|$ is even (i.e. $|b_R| \neq 1$), so any node in b_R has a same-colored neighbor. And any node in $b_j \setminus b_R$ has either z as its neighbor or another node from $b_j \setminus b_R$ as neighbor. Since z is blue-colored, every node in $b_j \setminus b_R$ also has a same-colored neighbor. Thus, after Phase 1 ends, every descendant of z has a same-colored neighbor. Also, we color at most r nodes red by the end of Phase 1 (ensured by Line 18).

If r nodes were not colored red by the end of Phase 1, then we move to Phase 2 (Line 22). Note that if more nodes need to be colored red after Phase 1, then every $b_i,\ i \in [k]$ has at most 1 blue node. In each iteration, we choose some $j \in [k]$ with $|b_j|$ odd, and color the remaining blue node v in b_j red (the number of nodes to be colored red is denoted by `reqd` in Line 27, such that `reqd` $\leqslant 1$). This is done till r nodes get colored red. If $|b_j| = 1$, then the node $v \in b_j$ satisfies Equation (6) (as $d(v) = 1$). Else the whole set b_j is colored red, and since $G[b_j]$ forms a connected graph (Observation 3), so all nodes in b_j will have a same-colored neighbor. Finally, note that the two phases will end after coloring exactly r nodes red, since $r \leqslant |C|$.

2. r is odd: We start at Line 4. If $\exists j \in [k]$ with $|b_j| = 1$ (Line 5), then we color the only vertex $v \in b_j$ red. Since $d(v) = 1$, v satisfies Equation (6). And we can color the remaining graph according to r even case. Otherwise, if $\exists j \in [k]$ with $|b_j| \geqslant 3$ (Line 7), then we color the last 3 nodes in the order of S_j red, i.e. $b_R = \{v \in b_j \mid S_j(v) > |b_j| - 3\}$ is colored red (we can do so since $r \neq 1$). By Observation 3, b_R forms a connected graph, so that every node in b_R has a same-colored neighbor. The remaining graph can again be colored using the r even method. Finally, if the above two conditions do not hold, then we must have $|b_i| = 2,\ \forall i \in [k]$ (Line 9). In this case, we color the first $^{(r-1)}/_2$ biconnected blocks red, along with node z. Then every $b_i,\ \forall i \in [k]$ has 2 neighboring nodes of the same color, ensuring that both of these nodes satisfy Equation (6).

Thus, the output of Algorithm 2 returns a $2-$partition $X = (\mathcal{R}, \mathcal{B})$ such that $|\mathcal{R}| = r$ and every descendant of node z satisfies Equation (6). We now show that Equation (6) holds for all nodes in $V \setminus C$ as well, and Equation (7) holds for all $j \in [k]$. Consider the color of node z in partition X.

1. $z \in \mathcal{B}$: Note that $G[V \setminus C]$ is also a connected graph (since z is a cut vertex). And every node $v \in V \setminus C$ with $v \neq z$ is also colored blue. If $|V \setminus C| > 1$, then every node in $V \setminus C$ has a same-colored blue neighbor. Otherwise, we must have $V = C \cup \{z\}$. Since $b > 1$, there must be at least 1 blue node in C. As we color nodes red from the end in the st-numbering of the biconnected blocks in C, some blue node $v \in C$ must be a neighbor of z, as desired.

2. $z \in \mathcal{R}$: This can only happen when $|b_i| = 2,\ \forall i \in [k]$ (Line 9). Thus, if $\exists j \in [k]$ with $|b_j| = 1$, then we must have $z \in \mathcal{B}$, proving Equation (7). Since $r > 1$, z will have a same-colored red neighbor in b_1. Let $V_B = V \setminus (C \cup \{z\})$. Then the graph $G[V_B]$ is a blue-colored connected graph. If $|V_B| > 1$, then each node in V_B will also have a same-colored neighbor in V_B. Otherwise, if $|V_B| = 1$, then this node $v \in V_B$ satisfies $d(v) = 1$.

Thus, the output of Algorithm 2 satisfies all the conditions given in this Lemma.

(Complexity): To see why Algorithm 2 finishes in $\mathcal{O}(|V| + |E|)$ time, note that it requires finding the st-numbering of each biconnected block (which is a linear-time operation), and simply iterating over the blocks in phases 1 and 2. □

Algorithm 3 CONNPART: Block – Cut Partition

Input: Connected graph $G = (V, E)$, integers r, b
Require: $r, b > 1;\ r + b = |V|$
Output: $X = \{\mathcal{R}, \mathcal{B}\}, |\mathcal{R}| = r, |\mathcal{B}| = b$

1: **if** G is biconnected **then**
2: **return** BICONNPART$(G,\ r,\ b)$
3: **end if**
4: $\mathcal{T} \leftarrow$ **BlockCutTree**(G) rooted at a cut vertex c
5: $\mathcal{R} \leftarrow \emptyset;\ V_{\text{rem}} \leftarrow V$
6: **STARTLOOP:**
7: $z \leftarrow$ Arbitrary leafy-cut vertex of $\mathcal{T}[V_{\text{rem}}]$
8: z has k biconnected blocks $b_1, b_2, \ldots, b_k$ as children
9: $C \leftarrow \bigcup_{i=1}^{k} b_i$ // b_i does not contain $z, \forall i \in [k]$
10: **if** $r > |C| + 1$ **then**
11: // Color all children of z red, and trim the graph
12: $\mathcal{R} \leftarrow \mathcal{R} \cup C;\ r \leftarrow r - |C|;\ V_{\text{rem}} \leftarrow V \setminus \mathcal{R}$
13: **goto** STARTLOOP
14: **else if** $r = |C| + 1$ **then**
15: // Color z and all its children red
16: $\mathcal{R} \leftarrow \mathcal{R} \cup C \cup \{z\}$
17: **else if** $r < |C| + 1$ **then**
18: $G \leftarrow G[V_{\text{rem}}];\ \mathcal{T} \leftarrow \mathcal{T}[\mathcal{V}_{\text{rem}}]$
19: $\mathcal{R} \leftarrow \mathcal{R} \cup$ SELECTPAIRSFIRST$(G, \mathcal{T}, z, \{b_i\}, r, b)$
20: // If r is odd, we first color odd nodes to make r
21: // even. Pairs are then selected from the b_i's.
22: // Finally, an odd element of some b_i's may be
23: // chosen till the requirement of r nodes is met.
24: **end if**
25: **return** $X = (\mathcal{R},\ V \setminus \mathcal{R})$

Lemma 5 provides a complete characterization of the output of Algorithm 3 for any connected graph G.

Lemma 5. *For any connected graph $G = (V, E)$ and $r, b > 1$ with $r + b = |V|$, Algorithm 3 returns a $2-$partition $X = (\mathcal{R}, \mathcal{B})$ s.t. $|\mathcal{R}| = r, |\mathcal{B}| = b$ and $\forall i \in V$, Equation (6) holds.*

Proof. If G is biconnected, then Algorithm 3 returns the output of Algorithm 1 (in Line 2). In this case, we get the desired property of the $2-$partition from Lemma 3. Suppose G is not biconnected. Let $\mathcal{T}$ be the block-cut tree of G rooted at some cut vertex c. Suppose z is the chosen leafy-cut vertex on Line 7, with $k \geqslant 1$ biconnected blocks $\{b_i\}_{i=1}^{k}$ as children. An example of this setup is shown in Figure 1. Also let $C = \cup_{i=1}^{k} b_i$, and note that $|C| \geqslant 1$ (since every biconnected block has at least 1 vertex). Here, the graph being partitioned

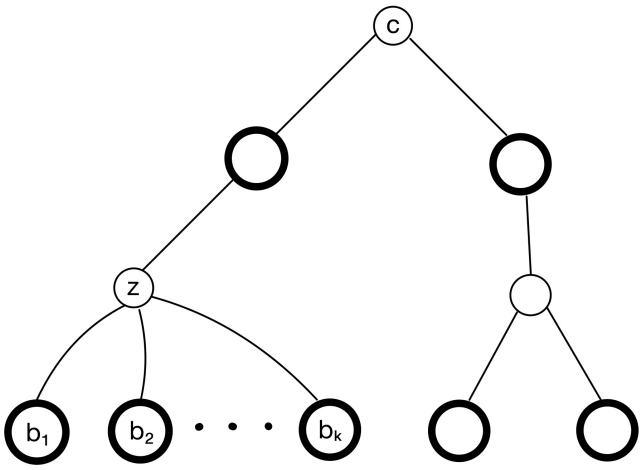

Figure 1: Block-cut tree $\mathcal{T}$ rooted at a cut vertex c; z is a leafy-cut vertex in $\mathcal{T}$, and has biconnected blocks $\{b_i\}_{i=1}^{k}$ as children

is $G[V_{\text{rem}}]$ (initially, $V_{\text{rem}} = V$). In each iteration (Line 6), there are 3 possible situations:

Case 1: $r \leqslant |C|$: We apply the SELECTPAIRSFIRST procedure (Line 19). By Lemma 4, every $i \in V_{\text{rem}}$ must have either $d_{G[V_{\text{rem}}]}(i) = 1$ or $|N^X(i)| \geqslant 1$. If Equation (6) is not satisfied by i in the final partition X of G, then the second condition cannot hold, giving $d_{G[V_{\text{rem}}]}(i) = 1$ (i.e. $|b_j| = 1$) and $d_G(i) > 1$. Thus, i is a cut vertex in G, and so it will have a pre-colored red child (from some previous iteration). If i is red, then this child is a same-colored neighbor of i. Else if i is blue, then Lemma 4 gives that its parent z must also be blue. Hence, we reach a contradiction in both situations, and every node $i \in V_{\text{rem}}$ satisfies Equation (6).

Case 2: $r = 1 + |C|$: z and all its children are colored red (Line 16). Note that $G[C \cup \{z\}]$ and $G[V_{\text{rem}} \setminus (C \cup \{z\})]$ are red and blue colored connected graphs respectively, each having size > 1. Thus, every node in V_{rem} has at least 1 neighbor in the same group, satisfying Equation (6).

Case 3: $r > 1 + |C|$: In this case, we color all nodes in C red (Line 12), and reduce r to $r' = r - |C|$. For any $j \in [k]$, there are 2 cases:

1. $|b_j| > 1$, so that every node $i \in b_j$ has a same-colored red neighbor in b_j (since $G[b_j]$ is connected).
2. $|b_j| = 1$, so that there is only one node $v \in b_j$. If $d_G(v) = 1$ (i.e. v is only joined to z in G), then v already satisfies Equation (6). Otherwise, if v is a cut vertex in G, then v (which is colored red) must have a pre-colored red child (from some previous iteration), giving $|N^X(v)| \geqslant 1$.

Thus, Equation (6) holds for all $i \in C$. The graph is now trimmed to $G' = G[V_{\text{rem}} \setminus C]$, which is still a connected graph. And the problem reduces to finding a $2-$partition $X' = (\mathcal{R}', \mathcal{B}')$ in this connected graph G' with $|\mathcal{R}'| = r', |\mathcal{B}'| = b$ (where r', b are still > 1 and $r' < r$). We now repeat the same procedure for this subgraph. □

Lemmas 2 and 5 together give the desired BP-2 for connected graphs, as given in Lemma 6 below.

Lemma 6. *For any connected graph $G = (V, E)$, there exists an* EF$-(\max(\Delta - 2, 2))$ BP-2.

Proof. By Lemmas 2 and 5, for any connected graph with $|V| \geqslant 4$, taking $r = \lceil |V|/2 \rceil$ in Algorithm 3 returns an EF$-(\max(\Delta - 2, 2))$ BP-2. And, if $|V| \leqslant 3$, then Observation 2 ensures the existence of an EF-2 BP-2 (since any node has $\leqslant 2$ neighbors, so it cannot have envy > 2). □

Proof. (of Theorem 1) Divide an arbitrary graph G into its connected components $c_1, c_2, \ldots, c_m$. For $j \in [m]$, let $G_j = \bigcup_{i=1}^{j} c_i$, with $G_m = G$. We will use induction on $j \in [m]$ to get an EF$-(\max(\Delta - 2, 2))$ BP-2 for all G_j. By Lemma 6, G_1 has such a partition. For the inductive step, suppose the desired BP-2 X exists for G_{j-1} ($j > 1$). If there are more blue nodes in X than red nodes, then color c_j as per Lemma 6, with at least as many red nodes as blue nodes. Else color c_j with the opposite colors. This ensures that the absolute difference between the number of red and blue nodes in this new $2-$partition of G_j is at most 1, giving us a BP-2 of G_j. Also, since the partition in each component is EF$-(\max(\Delta - 2, 2))$ and there are no edges between components, this final BP-2 will also be EF$-(\max(\Delta - 2, 2))$, completing the induction and our proof for Theorem 1.

(Complexity): As per the above proof, to find the final BP-2, we have to run Algorithm 3 for each c_j, $j \in [m]$ once. Algorithm 3 requires finding the block-cut tree of a connected graph and the $st-$numbering within different biconnected components, both of which are linear-time algorithms. Algorithm 2 is also called only once (in Line 19 of Algorithm 3), and it is also linear time according to Lemma 4. This ensures that for each connected component of G, Algorithm 3 runs in linear time, making the overall algorithm finish in $\mathcal{O}(|V| + |E|)$ time. □

3.2 Pareto Optimality

Theorem 2. *For every graph G with a* BP-2 X *that is* EF$-r$ *for some $r \geqslant 0$, there exists a* BP-2 X' *that is* EF$-r$ *and* PO.

Proof. Suppose some graph G has a BP-2 X that is EF$-r$. If X is PO, then we are done. Otherwise, we perform a sequence of Pareto improvements to X till we get a PO BP-2 X'. We claim that X' is EF$-r$. Suppose X' is not EF$-r$. Then by condition 1 of Lemma 1, $\exists i \in V$ such that $d(i) - r > 2|N^{X'}(i)|$. Since all variables in this inequality are natural numbers, this implies that $d(i) - r - 1 \geqslant 2|N^{X'}(i)|$ or $|N^{X'}(i)| \leqslant \frac{d(i)-r-1}{2}$. However, since X is EF$-r$, by condition 2 of Lemma 1, we have $|N^X(i)| \geqslant \frac{d(i)-r-1}{2}$. The partition X' is obtained by sequential Pareto improvements from X, and hence $|N^X(i)| \leqslant |N^{X'}(i)|$. Combining these three inequalities, we get

$$|N^X(i)| = \frac{d(i) - r - 1}{2} = |N^{X'}(i)|. \tag{8}$$

Suppose the BP-2's are given by $X = (\mathcal{R}, \mathcal{B})$ and $X' = (\mathcal{R}', \mathcal{B}')$, and WLOG $i \in \mathcal{R}, \mathcal{R}'$. From Equation (8), we have

$(d(i) - |N^X(i)|) - |N^X(i)| = r+1$, i.e., 'the number of blue ($\mathcal{B}$) neighbors of i' minus 'the number of red ($\mathcal{R}$) neighbors of i' is $r+1$. Consider the following two possible cases.

1. $|\mathcal{B}| = \lfloor n/2 \rfloor < \lceil n/2 \rceil$: According to Definition 3, i can apply a **tran** operation from $\mathcal{R}$ to $\mathcal{B}$ and increase its utility by $r+1$, which contradicts X being EF$-r$.
2. $|\mathcal{B}| = \lceil n/2 \rceil$: Consider an arbitrary node $j \in \mathcal{B}$ and a possible **swap** of i with j. If j is not a neighbor of i, then on swapping groups with j, the utility of i increases by $r+1$, which contradicts X being EF$-r$. So every $j \in \mathcal{B}$ can only be a neighbor of i, or equivalently, $\mathcal{B} = N(i) \setminus N^X(i)$. But, we also have $|N^X(i)| = |N^{X'}(i)|$, which gives $\lceil n/2 \rceil = |B| = d(i) - |N^X(i)| = d(i) - |N^{X'}(i)| \leqslant |\mathcal{B}'|$, where the last inequality comes from the fact that the blue neighbors of i must be at most the whole of $\mathcal{B}'$. Since X' is a BP-2, $|\mathcal{B}'|$ is also bounded from above by $\lceil n/2 \rceil$, and hence $|\mathcal{B}'| = d(i) - |N^{X'}(i)| = \lceil n/2 \rceil \Rightarrow \mathcal{B}' = N(i) \setminus N^{X'}(i)$. Thus, we have $|\mathcal{R}'| = \lfloor n/2 \rfloor$, which means that only the **swap** operation is possible for i in X'. Pick an arbitrary $j' \in \mathcal{B}'$ for i to **swap** with. This **swap** leads to a change in utility of i from $|N^{X'}(i)|$ to $d(i) - |N^{X'}(i)| - 1$, which is $d(i) - 2|N^{X'}(i)| - 1 = r$. This contradicts the fact that i has an envy $> r$ with some $j' \in \mathcal{B}'$.

Since both conclusions above lead to a contradiction, our original assumption on X' not being EF$-r$ must be false, and X' is the desired PO EF$-r$ partition. □

Theorem 3. *Consider an arbitrary graph G that has a* BP-2 *X which belongs to the $(\alpha, \beta)-$core for some $\alpha \geqslant 1, \beta \geqslant 0$.*

1. *Then there exists a* PO BP-2 *X' in the $(\alpha, \beta)-$core.*
2. *If X is also* EF$-r$ *for some $r \geqslant 0$, there exists a* BP-2 *X' in the $(\alpha, \beta)-$core, which is both* EF$-r$ *and* PO.

Proof. Part 1: Assume, for contradiction, that X is not PO. Let X' be PO partition obtained by sequentially improving X (as we did for Theorem 2). Suppose X' is not in $(\alpha, \beta)-$core. Then there exists a blocking coalition S of size either $\lfloor n/k \rfloor$ or $\lceil n/k \rceil$ such that for the BP-2 $Y = (S, V \setminus S)$, $u_i(Y) > \alpha \cdot u_i(X') + \beta$ for all $i \in S$. But, by construction, $u_i(X') \geqslant u_i(X)$ since X' was obtained by sequentially improving X. Hence, S is also a blocking coalition for BP-2 X, which contradicts the fact that X is in $(\alpha, \beta)-$core.

Part 2: Now, suppose X is also EF$-r$. Construct X' in a similar way as part 1. Part 1 shows that X' will be in the $(\alpha, \beta)-$core. By Theorem 2, X' will also be EF$-r$. □

4 Main Results: Grid Graphs

In Section 3, we saw the fairness and efficiency guarantees for general graphs. These results have their limitations owing to the arbitrary structure of the graphs. We focus on grid graphs in this section to investigate whether a relatively simpler graph structure can yield stronger fairness and stability properties. In the following two subsections, we will consider envy-freeness and core properties, respectively, for GrGs.

Algorithm 4 L2R: Left-to-Right

Input: $G = (V, E) \in$ GrG, partition size $k \geqslant 2$
Output: BP$-k$ X of G

```
Suppose |V| = n such that n = q · k + r, 0 ⩽ r < k
// Arrange nodes from left-top to right-bottom
Sort nodes in V using (x, −y) as key
// First r groups have ⌈n/k⌉ nodes
// Next (k − r) groups have ⌊n/k⌋ nodes
for cnt = 1 to k do
    X_cnt ← ∅; num ← 1
    size ← ⌈n/k⌉ if cnt ⩽ r else ⌊n/k⌋
    while num ⩽ size do
        v ← left uppermost node in V ; V ← V \ {v}
        X_cnt ← X_cnt ∪ {v}; num ← num + 1
    end while
end for
```

4.1 Envy-Freeness

We start this section by showing the incompatibility of BP$-k$ and EF-0 for all $k \geqslant 2$ even in GrGs.

Example 1. *For any $k \geqslant 2$, the graph*

$$G_k = \textbf{comb}_{k+1} \cup \overline{K_{k-2}}$$

consisting of a comb graph with $2(k+1)$ vertices and $k-2$ isolated vertices, has no EF-0 BP$-k$.

To see why, note that each subset in a BP$-k$ *X of G_k should have 3 nodes. Let S_1 denote the set of nodes with degree 1. Since $|S_1| = k+1$ and there are only k groups, $\exists\, i, j \in S_1$ which are in the same group X_ℓ (say). But if X is* EF-0*, then every node in S_1 must have their only neighbor in the same subset. This would imply that $|X_\ell| \geqslant 4$ (i, j and their 1 neighbor each), which is a contradiction.* □

However, if the EF guarantee is relaxed to EF-2, we show that there is an efficient algorithm (Algorithm 4) to find a BP$-k$ in GrGs, for any $k \geqslant 2$. The algorithm traverses the grid graph column-wise along the integer lattice $\mathbb{Z}^2$ from the left-top corner to the right-bottom corner, placing a contiguous sequence of traversed vertices in one subset of the partition.[3] The traversal is continued until there is no vertex left. This method first creates $n \pmod k$ such subsets of size $\lceil |V|/k \rceil$, and then remaining subsets of size $\lfloor |V|/k \rfloor$. An example of a BP-2 created by Algorithm 4 is shown in Figure 2.

Theorem 4. *For all $k \geqslant 2$ and $G = (V, E) \in$ GrG, Algorithm 4 returns an* EF-2 BP$-k$ *X in $\mathcal{O}(|V| \log(|V|))$ time.*

Proof. Assume, for contradiction, that BP$-k$ X (returned by Algorithm 4) is not EF-2. Then $\exists i, j \in V$ such that i envies j by at least 3. By Observation 2, i has at least 3 neighbors in $X_{X^{-1}(j)}$. Define $N_{TL}(i)$ as the set of neighbors (can be empty) to the top and left of i in the integer lattice $\mathbb{Z}^2$. Formally, if $i = (x, y)$ on the lattice, then $N_{TL}(i) = \{(x, y+1), (x-1, y)\} \cap N(i)$. Similarly define $N_{BR}(i) = \{(x, y-1), (x+1, y)\} \cap N(i)$ as the neighbors of i that lie to the bottom and right of i. Note that

[3] Here traversal is done over all vertices that exist in the graph on the lattice irrespective of whether they are joined by an edge or not.

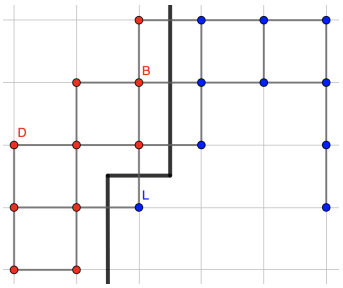

Figure 2: Partitioning done by Algorithm 4 (L2R) for $k = 2$

$|N_{BR}(i)|, |N_{TL}(i)| \leqslant 2$. Since at least 3 neighbors of i are in the same subset, by the pigeonhole principle, $\exists\ u \in N_{TL}(i),\ v \in N_{BR}(i)$ such that both nodes u and v belong to the same subset $X_{X^{-1}(j)}$ of the partition returned by Algorithm 4. But then all nodes visited between u and v in Algorithm 4 should also be in $X_{X^{-1}(j)}$, which includes i. This leads to a contradiction.

(Complexity): For the time complexity, note that Algorithm 4 sorts V in $\mathcal{O}(|V|\log(|V|))$ time, and then iterates through all the vertices in this sorted order, making the whole algorithm complete in $\mathcal{O}(|V|\log(|V|))$ time. □

4.2 Core

Theorem 5. *For every $G = (V, E) \in$ GrG, there is a polynomially computable* EF-2 BP-2 *that lies in the* $(1, 1) -$ *core.*

Consider the partition X churned out by Algorithm 4 (L2R) on a GrG $G = (V, E)$ for $k = 2$. From Theorem 4, no node has more than 2 neighbors in the other subset in X, i.e., for all $i \in V$, $u_i(X) \geqslant d(i) - 2$. We classify the nodes into three categories, based on their neighbors in the other group.

Definition 11. *Let* $X =$ L2R $(G = (V, E), k = 2)$. *Then we call a node* $i \in V$, (1) *an* internal *node if* $u_i(X) = d(i)$, (2) *an* edge *node if* $u_i(X) = d(i) - 1$, *and* (3) *a* corner *node if* $u_i(X) = d(i) - 2$.

For instance in Figure 2, D is an internal node, B is an edge node and L is a corner node.

Observation 4. *There can be at most* 2 *corner nodes. If both exist, there must be an edge between them, and they cannot share a common neighbor.*

Proof. Let $X = (\mathcal{R}, \mathcal{B})$. Let $c_R \in \mathcal{R}$ be the bottom-most node in the rightmost column of $\mathcal{R}$, and $c_B \in \mathcal{B}$ be the uppermost node in the leftmost column of $\mathcal{B}$. Then c_B will be visited immediately after c_R in the traversal done in Algorithm 4. For any $i \in \mathcal{R},\ i \neq c_R$, i will have at most 1 neighbor in $\mathcal{B}$ (a neighbor in $\mathcal{B}$ can only lie to its right). So the only possible corner node in $\mathcal{R}$ is c_R. Similarly, for any $j \in \mathcal{B},\ j \neq c_B$, j will have at most 1 neighbor in $\mathcal{R}$ (that can lie to its left), making c_B the only possible corner node in $\mathcal{B}$. Thus, there are at most 2 corner nodes. Finally, if both c_R and c_B are corner nodes, then they must be joined to each other to ensure that each of them has 2 neighbors in the other subset. And, since there can be no C_3 in a grid graph, the 2 adjacent corner nodes cannot share a common neighbor. □

Algorithm 5 CoreL2R: Core Left-to-Right

Input: $G = (V, E) \in$ GrG
Output: BP-2 X of G

```
X ← L2R(G, k = 2); // Algorithm 4
if noBlocking(X) then return X
else
    S = blockingSet(X) ; X' = (S, V \ S)
    if noBlocking(X') then return X'
    else
        S' = blockingSet(X'); return (S', V \ S')
    end if
end if
```

We first prove a lemma that will aid us in proving the $(1, 1) -$ core property.

Lemma 7. *No internal or edge node can be a part of a* $(1, 1)-$*blocking coalition in* BP-2 X.

Proof. For a node i to be a part of a $(1, 1)-$blocking coalition S (Definition 5), we must have $u_i((S, V \setminus S)) > u_i(X) + 1$, which gives $u_i(X) \leqslant d(i) - 2$ (since the utility of i is bounded above by the degree). From Definition 11, this is only possible for corner nodes, as desired. □

Proof. (of Theorem 5) According to Theorem 4, BP-2 X returned by Algorithm 4 is already EF-2. For the sake of contradiction, suppose a $(1, 1)-$blocking coalition S exists in partition X. Using Lemma 7, we see that only the corner nodes can be a part of S, which gives $|S| \leqslant 2$ (using Observation 4). So any vertex in S can have at most 1 neighbor in S. Thus, for any $i \in S$, we have $u_i((S, V \setminus S)) \leqslant 1 \leqslant u_i(X)+1$, which is a contradiction. This proves that such a blocking coalition S cannot exist, and so the BP-2 returned by Algorithm 4 is both EF-2 and in the $(1, 1) -$ core. □

Discussion: For arbitrary graphs, Li *et al.* (2023) showed that when $k \geqslant 3$, a BP$-k$ in the $(1, 0) -$ core may not exist, with the graph C_{k+1} as the desired counter-example. Since odd cycles can not exist in GrGs, the same example does not work when k is even. So for even k, we can instead consider P_{k+1} as an instance of a GrG that has no BP$-k$ in $(1, 0) -$core (the same proof works). Thus, for $k \geqslant 3$, there may be no BP$-k$ in $(1, 0) -$ core even for GrGs. Li *et al.* (2023) also raised an open problem about the existence of a BP-2 belonging to $(1, 0) -$ core in arbitrary graphs. We show that for GrGs, we can always find such a partition.

Theorem 6. *For every* $G \in$ GrG, *Algorithm 5 returns a* BP-2 *that belongs to* $(1, 0) -$ *core.*

Consider Algorithm 4 (L2R) for $k = 2$, which returns a BP-2 X. Suppose it is not in the core (otherwise we have the required partition). Then there exists a blocking coalition S in our partition X. Let $X' = (S, V \setminus S)$ be another BP-2. As per Definition 5, for a node i to be a part of S, we must have $u_i(X') > u_i(X)$. Thus, its utility in X' must strictly increase over that in X. We have the following lemma.

Lemma 8. *No internal node can lie in the blocking coalition* S *as defined above. Also, if an edge node* $j \in S$, *then all of its neighbors are also in* S *(i.e.,* $N(j) \subseteq S$*).*

Proof. An internal node has all neighbors of the same color as itself, so its utility cannot increase any further. Thus, it cannot be part of S. Suppose an edge node $j \in S$. Since $u_j(X') \geqslant u_j(X) + 1 = d(j)$, and the utility of j is also bounded above by $d(j)$, we get equality here, i.e. $u_j(X') = d(j)$ (all neighbors of j are in S). □

Proof. (of Theorem 6) If X' is in core then we have the desired partition. Otherwise, there is some blocking coalition S' in this new partition X' as well. Consider any $i \in S'$. There are 2 possible scenarios:

Case 1: $i \in S' \cap S$*:* then $u_i((S', V \setminus S')) > u_i(X') > u_i(X) \implies u_i((S', V \setminus S')) \geqslant u_i(X) + 2$. This is only possible for corner nodes, as only they can have 2 neighbors from the other group in X. And, if a corner node i is in S', then all its neighbors must also be in S' to ensure the above condition (since $u_i(X) = d(i) - 2$).

Case 2: $i \in S' \cap T$*:* then $u_i((S', V \setminus S')) \geqslant u_i(X') + 1$, which means that i must have at least 1 neighbor $\in S$ so that its utility can increase from that in X'. Call any such neighbor j. As per Lemma 8, j cannot be an internal node. If it is an edge node, all its neighbors will also be in S (Lemma 8), which would imply that it cannot have a neighbor $i \in T$. So the only option left is j being a corner node, i.e. i's neighbors in S can only be corner nodes. Since a node cannot be a common neighbor of two corner nodes (Observation 4), i will have exactly one neighbor in S. Thus, to increase the utility of i in $(S', V \setminus S')$, all neighbors of i must also be in S'.

In both the cases above, we see that any $i \in S'$ must have all its neighbors also in S', implying that there is no edge between S' and $V \setminus S'$. Then the BP-2 $(S', V \setminus S')$ is in the core, as desired. □

5 Conclusions and Future Work

In this paper, we have designed various efficient algorithms to find approximately envy-free balanced partitions, along with existential results on stable and Pareto optimal partitions. Our work raises several important questions:

- Is it possible to show the existence of an EF-2 BP-2 in all biconnected graphs, and then to show it for all connected graphs by making use of the block-cut tree?
- For an arbitrary graph, can $(1, 0)-$blocking coalitions be taken consecutively (similar to Algorithm 5) a total of $\mathcal{O}(\Delta(G))$ times to reach a BP-2 in $(1, 0)-$core?
- Does every `GrG` admit an EF-1 BP$-k$? We provide partial results on this in Appendix A.
- Given a BP-2 from Theorems 2 and 3, can the PO BP-2 be found via an efficient algorithm? Also, can this result be extended to any BP$-k$ ($k \geqslant 3$)?

References

Konstantin Andreev and Harald Räcke. Balanced graph partitioning. In *Proceedings of the Sixteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '04, page 120–124, New York, NY, USA, 2004. Association for Computing Machinery.

Esther M. Arkin, Sang Won Bae, Alon Efrat, Kazuya Okamoto, Joseph S.B. Mitchell, and Valentin Polishchuk. Geometric stable roommates. *Information Processing Letters*, 109(4):219–224, 2009.

Haris Aziz, Felix Brandt, and Paul Harrenstein. Pareto optimality in coalition formation. *Games and Economic Behavior*, 82:562–581, 2013.

Cristina Bazgan, Zsolt Tuza, and Daniel Vanderpooten. Satisfactory graph partition, variants, and generalizations. *European Journal of Operational Research*, 206(2):271–280, 2010.

Freya Behrens, Gabriel Arpino, Yaroslav Kivva, and Lenka Zdeborová. (dis) assortative partitions on random regular graphs. *Journal of Physics A: Mathematical and Theoretical*, 55(39):395004, 2022.

Niclas Boehmer and Edith Elkind. Stable roommate problem with diversity preferences. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 96–102. International Joint Conferences on Artificial Intelligence Organization, 7 2020. Main track.

Ágnes Cseh, Michael McKay, and David Manlove. Envy-freeness in 3d hedonic games. *arXiv preprint arXiv:2209.07440*, 2022.

J. Ebert. st-ordering the vertices of biconnected graphs. *Computing*, 30:19–33, 1983.

Andreas Feldmann. *Balanced Partitions of Grids and Related Graphs.* PhD thesis, 04 2012.

Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., USA, 1990.

John Hopcroft and Robert Tarjan. Algorithm 447: Efficient algorithms for graph manipulation. *Commun. ACM*, 16(6):372–378, jun 1973.

Chien-Chung Huang. Two's company, three's a crowd: Stable family and threesome roommates problems. In Lars Arge, Michael Hoffmann, and Emo Welzl, editors, *Algorithms – ESA 2007*, pages 558–569, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

Robert W Irving. An efficient algorithm for the "stable roommates" problem. *Journal of Algorithms*, 6(4):577–595, 1985.

B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 49(2):291–307, 1970.

Chaya Levinger, Amos Azaria, and Noam Hazon. Socially aware coalition formation with bounded coalition size. *arXiv preprint arXiv:2008.06179*, 2023.

Lily Li, Evi Micha, Aleksandar Nikolov, and Nisarg Shah. Partitioning friends fairly. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'23/IAAI'23/EAAI'23. AAAI Press, 2023.

Michael McKay and David Manlove. The three-dimensional stable roommates problem with additively separable preferences. In Ioannis Caragiannis and Kristoffer Arnsfelt Hansen, editors, *Algorithmic Game Theory*, pages 266–280, Cham, 2021. Springer International Publishing.

Michael McKay. *Algorithmic aspects of fixed-size coalition formation*. Phd thesis, University of Glasgow, 2022.

Dor Minzer, Ashwin Sah, and Mehtaab Sawhney. On perfectly friendly bisections of random graphs. *arXiv preprint arXiv:2305.03543*, 2023.

Renata Sotirov. Graph bisection revisited. *Annals of Operations Research*, 265:143–154, 2018.

Appendix

A EF-1 BP-2 in Grid Graphs

In this section, we show that an EF-1 BP-2 exists for GrGs with an odd number of nodes. For odd graphs, the two groups to be formed by the partition will have different numbers of nodes $\left(\frac{|V|-1}{2} \text{ and } \frac{|V|+1}{2}\right)$. This relaxes the partitioning problem by introducing the freedom of switching the group of a single node belonging to the larger group.

Theorem 7. *For every $G = (V, E) \in$ GrG with $|V| = n$ and $n \mod 2 = 1$, there exists an* EF-1 BP-2 *that can be found in $\mathcal{O}(n \log(n))$ time complexity.*

Proof. We start with Algorithm 4 ($L2R$) and make some modifications based on different cases that arise. According to Observation 2, the only nodes that can violate the EF-1 property are the corner nodes, since the remaining nodes can have at most 1 neighbor in the other subset (Definition 11). Before proceeding to the intricate casework, it is important to note that Algorithm 4 will provide a BP-2 X in which the group on the left side (say colored red) will have more nodes than the other group (say colored blue). Thus, we have the option to switch the color of a single red node in X to blue.

We call a node *conflicting* if it does not satisfy EF-1 property (i.e. its envy towards some other node is greater than 1). A corner node i is *conflicting* only when it has exactly two neighbors and both belong to the other group. Since if i has a neighbor in the same subset (i.e. $u_i(X) \geqslant 1$), then its utility on joining the other subset will have to change to at least 3, which is not possible.

- **Case 1 :** Red corner node A is conflicting.
  Simply switching the color of this conflicting node resolves the issue, as shown in Figure 3. In the resulting partition, A is no longer conflicting, and the other possible corner node (i.e. E) can also not be conflicting.

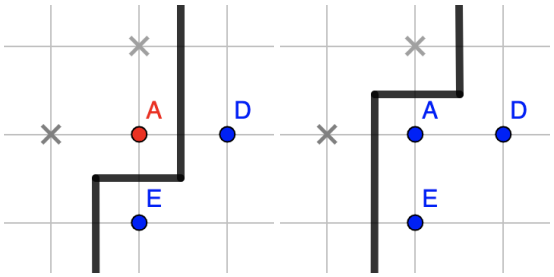

Figure 3: Case 1

- **Case 2 :** Only the blue corner node E is *conflicting*, and switching the color of the red node directly above (i.e. A) resolves the issue.
  As shown in Figure 4, after switching the color of node A, the new possible corner nodes so formed will be A and the node (say B) above it. A will have at least 1 neighbor in the same subset (i.e. E). The case where node B becomes *conflicting* is handled in Case 3 below.

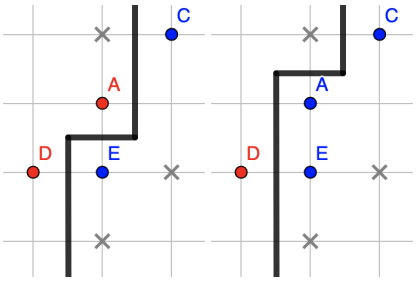

Figure 4: Case 2

- **Case 3 :** Only the blue corner node E is *conflicting*, and switching the color of the red node directly above (i.e. A) does not resolve the issue.
  Figure 5 shows the scenario in which simply switching the color of vertex A can result in the node above it becoming *conflicting*. Here, node B will have two neighbors A and C belonging to the other group (after switching the color of A), and none in its own group. For this case, constructing the boundary as shown in the last diagram of Figure 5 (i.e. switching the groups of E and B from the original BP-2 X) resolves the issue since no node is now *conflicting*.

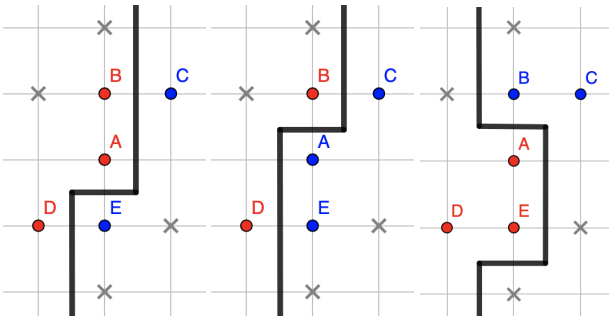

Figure 5: Case 3

Thus, in all scenarios, we can find a BP-2 in which no node violates the EF-1 condition, as desired. □