Budget-feasible Egalitarian Allocation of Conflicting Jobs

AAAI Track

Sushmita Gupta
The Institute of Mathematical Sciences
Chennai, India
sushmitagupta@imsc.res.in

Pallavi Jain
Indian Institute of Technology Jodhpur
Jodhpur, India
pallavi@iitj.ac.in

A. Mohanapriya
The Institute of Mathematical Sciences
Chennai, India
mohana@imsc.res.in

Vikash Tripathi
The Institute of Mathematical Sciences
Chennai, India
vikasht@imsc.res.in

ABSTRACT

Allocating conflicting jobs among individuals while respecting a budget constraint for each individual is an optimization problem that arises in various real-world scenarios. In this paper, we consider the situation where each individual derives some satisfaction from each job. We focus on finding a feasible allocation of conflicting jobs that maximize egalitarian cost, i.e. the satisfaction of the individual who is worst-off. To the best of our knowledge, this is the first paper to combine egalitarianism, budget-feasibility, and conflict-freeness in allocations. We provide a systematic study of the computational complexity of finding budget-feasible conflict-free egalitarian allocation and show that our problem generalizes a large number of classical optimization problems. Therefore, unsurprisingly, our problem is NP-hard even for two individuals and when there is no conflict between any jobs. We show that the problem admits algorithms when studied in the realm of approximation algorithms and parameterized algorithms with a host of natural parameters that match and in some cases improve upon the running time of known algorithms.

KEYWORDS

Fair Allocation; Maximize Egalitarian Cost; Parameterized Algorithms; Approximation Algorithms

ACM Reference Format:
Sushmita Gupta, Pallavi Jain, A. Mohanapriya, and Vikash Tripathi. 2024. Budget-feasible Egalitarian Allocation of Conflicting Jobs: AAAI Track. In *Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), Auckland, New Zealand, May 6 – 10, 2024*, IFAAMAS, 9 pages.

1 INTRODUCTION

The division of resources among several interested parties that is satisfactory to all is a central question in game theory and is studied under the name of *fair division*. It is among the most commonly encountered challenges in life and industry, such as splitting an inheritance, sharing rent, a partnership dissolution, sharing food, assigning jobs to people or machines, and on and on. Unsurprisingly, research on this forms the basis of a large body of research that spans mathematics, economics, computer science, operations research, and social sciences. The applied nature of the problem notwithstanding, research on fair division can be dated back to mathematics, in the 1948 work of [35].

In recent years, the theoretical AI community has delved into this question from multiple fronts, combining realistic constraints such as allocating conflicting resources that respect budget constraints. Chiarelli et al. [12], Biswas et al. [7], and Hummel et al. [26] have studied *conflict-free allocation*, one where the bundle of items assigned to any agent can contain at most one of the conflicting items. Such constraints arise quite naturally in scheduling problems, be it for jobs to machines, room assignments to sessions in a conference, or panel selections while avoiding conflicts of interest.

Each of the aforementioned papers study *conflict-free allocations* with different objective functions such as partial egalitarian (maximin) allocations [12], i.e allocations that maximize the utility of the worst-off agent (also called the Santa Claus guarantee), but where some items may remain unallocated; whereas [26] studies complete allocations with fairness criterion such as envy-freeness, maximin share guarantees, and Nash welfare; and [7] compliments the work of Chiarelli by studying uniform and binary valuations for the setting of course allocation. The other direction in which fair division has been explored is with *budget* constraint. Clearly, in most realistic scenarios, a notion of “budget” comes into play quite naturally, it can represent money available to purchase items, time available to complete tasks, and so on. Indeed, in recent years papers on the budgeted version of fair division have appeared in top algorithmic research conference venues, both in theory, Garg et al. [25], and in AI, Barman et al. [4], where the objectives are Nash social welfare and envy-freeness. These papers look at the classical computational complexity divide and devise algorithms for restricted settings of utility function families (such as [7, 26]) and for special graph classes ([12]) that capture the conflict among items. The precise definitions of the problem may vary in that [7] studies the course allocation problem, which is a many-to-many allocation scenario. In some cases, they give approximation algorithms that work in polynomial time.

In this paper, we extend this line of research by combining these two perspectives and study conflict-free allocation that respects

Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), N. Alechina, V. Dignum, M. Dastani, J.S. Sichman (eds.), May 6 – 10, 2024, Auckland, New Zealand.

budget constraints such that the worst-off agent attains a given level of utility/satisfaction, formally defined below. For example, consider the challenge facing a CEO of a startup who has a small number of employees, say k, and a large number of tasks, say n, that she needs to assign to her employees in a manner that respects logistical constraints so that no one is assigned more than one task in any given time, and also give each employee a level of satisfaction while ensuring that nobody's workload is excessive. This is a scenario in which our model, formally defined below, can be employed.

> Budgeted Conflict Free Egalitarian Allocation (BCFEA)
> **Input:** A set of agents $[k] = \{1, \ldots, k\}$, a set of n items V, each agent's utility function $\{p_i : 2^V \rightarrow \mathbb{Z}_{\geq 0}\}_{i \in [k]}$, and cost function $\{c_i : 2^V \rightarrow \mathbb{Z}_{\geq 0}\}_{i \in [k]}$; a graph on the item set $G = (V, E)$, called the *conflict graph*; and two positive integers P and B.
> **Question:** Does there exists a partition $\mathcal{S} = \{S_1, \ldots S_k\}$ of the item set I, called *bundles*, such that for each $i \in [k]$, S_i is an independent set in G, bundle's cost $c_i(S_i) \leq B$, and utility $p_i(S_i) \geq P$?

Moreover, in this paper we look at the fair allocation problem from the perspective of parameterized complexity that incorporates an upper bound on budget(or size) (such as [7]) on (conflict) graph classes that generalize well-known tractable cases (such as [12]) that go beyond maximum degree of the conflict graph (such as studied by [26]). Our problem generalizes several classical optimization problems, such as Partition, k-Coloring, 3-Parition, Santa Claus, Bin Packing, and Knapsack. Hence, unsurprisingly, our problem is NP-hard even under strict restrictions and thereby sets us well to explore a host of input and output parameters individually and in combination.

Specifically, we point to Chiarelli et al. [12], whose work is closely related to ours, as considering various graph classes and exploring the classical complexity boundary between strong NP-hardness and pseudo-polynomial tractability for a constant number of agents. Our analysis probes beyond the NP-hardness of BCFEA and explores this world from the lens of parameterized complexity and approximation algorithm, thereby drawing out the suitability of natural parameters–such as the number of agents k, the number of items n, the maximum size of each allocated set s, the number of distinct profit and cost values, λ. In addition to this, we explore the effectiveness of structural parameters of the underlying graph (such as the number of (connected) components, the treewidth, the pathwidth, and chordality)–towards yielding polynomial time algorithms when the parameters take on constant values. This is aligned with the emerging area of research in the theoretical AI community that has studied treewidth as a parameter along with various objectives: connected fair division, Deligkas et al. [16]; compact fair division, Madathil [30]; gerrymandering on planar graphs, Dippel et al. [19].

BCFEA is quite recognizable in its restricted avatar as several well-known problems such as k-Coloring, Partition, Knapsack, Bin packing, Job Scheduling [24], and Santa Claus [3]. Their connection to BCFEA can be easily established if for each agent $i \in [k]$, we take the utility and cost functions to be additive over each item. That is, we have functions $p_i, c_i : V \rightarrow \mathbb{Z}_{\geq 0}$ such that on any subset S of items we define $f_i(S) = \sum_{v \in S} f_i(v)$ for both $f_i \in \{p_i, c_i\}$. We briefly discuss the connection of these problems with BCFEA. The notation $[k]$ is used to denote the set $\{1, \ldots, k\}$.

k-Coloring: Let (H, k) be an instance of k-Coloring. The goal is to decide if there exists a proper coloring of H using at most k colors. In the reduced instance of BCFEA, $G = H$; $p_i(v) = 1$ and $c_i(v) = 0$, for each $i \in [k]$ and $v \in V(G)$; $P = 0$; and $B = 0$.

Partition: Let $S = \{s_1, \ldots, s_n\} \subseteq \mathbb{Z}$ define an instance of Partition. The goal is to find a subset $X \subseteq S$ such that $\sum_{s_i \in X} s_i = \sum_{s_i \in S \setminus X} s_i$. In the reduced instance of BCFEA, G is edgeless; $k = 2$; $p_i(v) = s_i$ and $c_i(v) = 0$ for each $i \in [k]$ and $v \in V$; $P = \sum_{v \in V} p_i(v)/2$ and $B = 0$.

3-Partition: The input consists of set S of $3m$ elements, a bound $X \in \mathbb{Z}_+$, and a size $s(x) \in \mathbb{Z}_+$ for each $x \in X$ such that $X/4 < s(x) < X/2$ and $\sum_{x \in S} s(x) = mB$. The goal is to decide whether there exists a partition of S into m disjoint sets $S_1, S_2, \ldots, S_m$ such that for each $1 \leq i \leq m$, $\sum_{x \in S_i} s(x) = B$. In our reduced instance, we set G to be edgeless, $k = m$, $p_i(v) = s_i$, and $c_i(v) = 0$ for each $i \in [k]$ and $v \in V$, $P = X$, and $B = 0$.

Egalitarian Fair Division[1]: Let $\mathcal{I} = (S, k, \{p_i\}_{i \in [k]}, \tau)$ denote an instance of Santa Claus problem, [3]. The goal is to decide if there exists a partition $(S_1, \ldots, S_k)$ of the set S such that $\min_{i \in k}\{\sum_{s \in S_i} s\} \geq \tau$. In the reduced instance, G is edgeless, $V = S$, $c_i(v) = 0$ and $p_i(v) = p_i(v)$ for each $i \in [k]$ and $v \in S$ and $B = 0$;

Bin Packing: Let $\mathcal{I} = (X = [n], W, k)$ be an instance of Bin Packing where each item $i \in X$ has size s_i. The goal is to decide if there exists a partition of X into k parts $(X_1, \ldots, X_k)$ such that for each $j \in [k]$, $\sum_{i \in X_j} s_i \leq B$. In the reduced instance, G is edgeless, and $p_i(v) = 0$, $c_i(v) = s_i$ for each $i \in [k]$ and $v \in V$, $B = W$, and $P = 0$.

Knapsack: Let $\mathcal{I} = (V', \{p, w\}_{v \in V}, P', W)$ be an instance of Knapsack, where p and w are the profit and cost functions, respectively, on the item set V, and $P', W \in \mathbb{N}$. The goal is to find a subset of V such that the sum of the profit values and the cost values of the item is at least P' and at most W, respectively. In the reduced instance of BCFEA, we have $V = V' \cup \{d\}$, G is edgeless, $k = 2$, and $p_1(v) = p_v$, $p_2(v) = P'$, $c_1(v) = w_v$, $c_2(v) = 0$ for each item $v \in V$, and for the "dummy" item $p_1(d) = 0$, $p_2(d) = P'$, $c_1(d) = B + 1$, $c_2(d) = 0$; and $P = P'$ and $B = W$.

Remark 1. *Under* P $\neq$ NP *and* FPT $\neq$ W[1] − hard*, we can infer the following.*

(a) Due to the reduction from Partition*,* BCFEA *is* NP-hard *even for $k = 2$, there are no conflicts, and $B = 0$. Thus, we cannot hope for an* FPT(k + tw + B) *time algorithm.*[2]
(b) Due to the reduction from 3-Coloring*,* BCFEA *is* NP-hard *even for $k = 3, P = 0, B = 0$, unit profit function and zero cost function. Thus, we cannot hope for an* FPT($k + P + B$) *algorithm even when values are encoded in unary;*
(c) Due to reduction from 3-Partition*,* BCFEA *is* NP-hard *even when $s = 3, B = 0$, and graph is edgeless. Thus, we cannot hope for an* FPT($s + B$) *or* FPT(tw + B) *even when values are encoded in unary.*

Here, tw *and s denote the treewidth of the conflict graph and the maximum size of a bundle, respectively.*

[2]For a parameter ζ, FPT(ζ) refers to running time of an algorithm that runs in $f(\zeta) \cdot m^{O(1)}$, for any computable function f.

Thus, the problem Fair k-division of Indivisible Items studied by Chiarelli et al. [12] can be reduced to BCFEA if for every agent $i \in [k]$ and item $v \in V$,

$c_i(v) = 1$ for every agent $i \in [k]$ and item $v \in V$, and the budget $B = n$, while the profit of each item $p_i(v)$

Our Contributions. Due to ease of exposition we will present the proofs for the case of *identical valuation and cost functions*, as defined formally below.

Input: An instance of BCFEA where for each agent $i \in [k]$, we have identical utility and cost functions, $p_i = p$ and $c_i = c$.
Output: A solution for BCFEA.
An instance of BCFEA is denoted by $\mathcal{I} = (G, k, p, c, P, B)$.

However, as we discuss in Section 3.3, each of our algorithms can be easily extended to the general setting of non-identical utility and cost functions. For simplicity of exposition, throughout the paper, we will use c_v and p_v to denote the cost $c(v)$ and the utility $p(v)$, respectively, for $v \in V$.

Our work in this paper is a deep dive into the combinatorial and optimization aspects of BCFEA whereby we show a wide array of algorithmic results that explore the NP-hard problem with respect to various parameters. As explained earlier, the formal definition of the problem can be seen as model that captures various well-known and well-studied combinatorial optimization problems. In this sections, we will discuss the relevance of the results presented in this paper vis-a-vis the known literature about the underlying problems. Here, we will discuss the wider backdrop of this work.

Remark 2. *When there is exactly one agent,* BCFEA *is rather simple to solve as the conflict graph must be edgeless and the sum of the costs and utilities of all the objects must be at most B and at least P, respectively. All these conditions are simple to test in linear time.*

Next, we discuss the results for the other values of k.

Number of agents is two: BCFEA exhibits a dichotomy, whereby if the conflict graph is edgeless, the problem is NP-hard and it is polynomial-time solvable if the graph is connected, Corollary 3. More generally, we show that BCFEA is NP-hard when the graph is disconnected, Theorem 1. However, we supplement this by exhibiting that BCFEA has an FPT algorithm with respect to the number of components, Theorem 2. In contrast to Remark 1, we note that when $k = 2$, BCFEA can be solved in polynomial-time when the values are encoded in unary, Theorem 4.

We note that Chiarelli et al. [12] study a restricted version of BCFEA, where there is no budget constraint, that is, there is no cost associated with any bundle. Our algorithm for the $k = 2$ case, when applied to their setting (Theorem 4) has a better running time $O(nP^2)$, where P is the utility of the worst-off agent, than the one Chiarelli et al. propose in [12] (Lemma 13), which is $O(nQ^4)$, where Q denotes the sum of the profits of all the items (clearly, $Q \geq P$).

Number of agents is arbitrary: We present an exact-exponential algorithm, Theorem 9, with running time $O^\star(2^n)$[3] for BCFEA, where n denotes the number of items. This is asymptotically tight given that under the Exponential Time Hypothesis (ETH) we know that k-Coloring cannot be solved in $2^{o(n)}$ time.

As a consequence of the hardness of BCFEA with respect to k, it is natural for us to probe the parameterized complexity of BCFEA with respect to k with *domain restrictions*, which in the context of our study means that we have restrictions on the graph, the number of pairs of utility and cost values (formalized as the *type*), the range of utility and cost values (or the size of the encoding) and the *bundle size*, defined to be the maximum size of a set assigned to an agent in a solution of BCFEA.

Each of the following results assumes that the values are encoded in unary.

- BCFEA has an FPT(k + tw) algorithm, Theorem 5. As discussed in the technical section, if the conflict graph is a chordal graph, then any yes-instance of BCFEA has treewidth is at most k. Specifically, if the conflict graph is an interval graph, then the pathwidth is at most k. Hence, when k is a constant, BCFEA admits a polynomial-time algorithm when G is an interval graph, Corollary 6. Interval graphs are of practical importance due to their relevance in Job Scheduling.
- **(Type)** BCFEA admits an FPT(k + tw + λ) algorithm, Theorem 7, where λ denotes the type. Consequently, when both k and λ are constants, the problem is polynomial-time solvable, Corollary 8. Due to Remark 1, we cannot expect an FPT(k + tw) algorithm or FPT(k + λ) algorithm.
- **(Bundle size)** Due to Remark 1(b) we cannot expect an FPT(k + P + B) algorithm. Hence, we look at s, the *bundle size*, as a parameter. The bundle size is likely to be small, but again due to Remark 1(c) we cannot expect an FPT(s) algorithm either. But for the case $s = 2$, we have a polynomial-time algorithm, Theorem 11. More generally, we show that we can have an FPT(k + s) algorithm, Theorem 10.
- **(FPT-AS)** BCFEA has an FPT(k + tw) algorithm that outputs a solution in which every agent has profit at least $P/(1 + \epsilon)$ and cost at most $(1 + \omega)B$ (such algorithms are known as FPT-*approximation schemes*), Theorem 12.

Choice of graph classes: As discussed above, BCFEA generalizes k-Coloring, it is the only graph problem among the ones discussed above. Thus, any hope of designing an algorithm for BCFEA rests on the tractability of k-Coloring in the conflict graph. Hence, our search for amenable graph classes is narrowed down to interval graphs, chordal graphs, and graphs with bounded treewidth. In chordal graphs, it is polynomial-time solvable, and in graphs of bounded treewidth it is FPT(tw), [14]. Interval graphs are of special significance to applications related to Job Scheduling. Our work in this article runs the gamut of designing algorithms for appropriate small valued parameters and graph classes where the underlying combinatorial problem is tractable. We bookend our algorithmic results with hardness borrowed from Bin Packing in addition to k-Coloring. Please refer to Table 1 for an overview of the results in this paper.

Related Work. In addition to the works discussed earlier in the Introduction, we will further discuss some more work that are related to our work in this paper. A well-known framework within which fair allocation has long been studied is in the world of Job Scheduling problems on non-identical machines. In this scenario, the machines are acting as agents and the jobs are the tasks such that certain machines are better suited for some jobs than others and this variation is captured by the “satisfaction level" of the

[3] $O^\star(\cdot)$ suppresses polynomial or poly-logarithmic factors.

#Agents	Graph	Complexity	Reference
1	General	$O(n)$	Remark 2
2	Connected	$O(n)$	Corollary 3
	r components, $r > 1$	NP-hard	Theorem 1
	r components	$O(2^r n)$	Theorem 2
	General	$O((BP)^2 n)$	Theorem 4
$O(1)$	Bounded Treewidth	$O(\text{tw}^k(\alpha\gamma)^{2k} n)$	Theorem 5
	Interval Graph	$O(k^k(\alpha\gamma)^k n)$	Corollary 6
	Bounded Treewidth	$O(\text{tw}^k(\log_{1+\epsilon}\alpha \log_{1+\omega}\gamma)^{2k} n^{O(1)})$	Theorem 12
	(Bi-criteria approximation)	$\left(\frac{1}{1+\epsilon}, 1+\omega\right)$-factor approx	
Arbitrary	General	$O^\star(2^n)$	Theorem 9
	General	FPT$(k+s)$	Theorem 10
	Bounded Treewidth	$O(\text{tw}^k(\lambda+1)^{2k} kn)$	Theorem 7

Table 1: Overview of our algorithms. Here, $\gamma = \sum_{v\in V} c(v)$, $\alpha = \sum_{v\in V} p(v)$, $\lambda = |\{(c_v, p_v) \mid \forall v \in V\}|$, tw denote the treewidth of G in an instance $\mathcal{I} = (G, k, p, c, P, B)$ of BCFEA, s denote the size of bundle, and $\epsilon, \omega \in (0, 1]$.

machine towards the assigned jobs. Moreover, the jobs have specific time intervals within which they have to be performed and only one job can be scheduled on a machine at a time. Results on the computational aspect of fair division that incorporates interactions and dependencies between the items are relatively few. This is the backdrop of our work in this article. A rather inexhaustive but representative list of papers that take a combinatorial approach in analysing a fair division problem and are aligned with our work in this paper is [1, 5, 6, 11, 12, 20, 29, 36, 37]. In particular, we can point to the decades old work of Deuermeyer et. al [17] that studied a variant of Job Scheduling in which they goal is to assign a set of independent jobs to identical machines in order to maximize the minimal completion time of the jobs. Their NP-hardness result for two machines (i.e. two agents in our setting) is an early work with a similar flavor. They analyse a well-known heuristic called the LPT-algoirthm to capture best-case performance and show that its worst case performance is 4/3-factor removed from optimum.

Moreover, we note that conflict like constraints in an underlying graph have also been studied in the context of Knapsack [32, 33] and Bin Packing [21] have also been studied. Interestingly, Pferschy and Schaue [32] studies Knapsack with conflict and present pseudo-polynomial algorithms for graphs of bounded treewidth and chordal graphs. From these algorithms, they derive fully polynomial time approximation schemes (FPTAS). More generally, there can be other combinatorial notions of quality that capture scenarios where certain subsets of items, based on what they represent, should be bundled together and thus be assigned to the same agent. Conflict is a special case of this in which we want independence between the items, connectivity is another well-studied notion. Such combinatorial properties are amenable to graph-theoretic modeling and have been studied as well [9, 15, 22, 23]. Suksompong [36] surveys the landscape of fair division with a host of combinatorial constraints such as connectivity, cardinality, matroid, geometric, separation, budget, and conflict.

Other paradigmatic problems that are subsumed by BCFEA is Bin Packing and Knapsack and the literature around them–both offline and online–is so immense that we cannot faithfully survey it here and point the reader to [13] and [28] for a deeper look.

We would like to conclude this discussion by noting that the literature on fair division is *vast* due to the myriad of variations among the nature of goods–divisible or indivisible; the nature of preferences–cardinal or ordinal; the nature of solution–based on envy-freeness or some objective function such as social welfare, Nash welfare, maximin, leximin, and several others that are too numerous to enumerate; and even the algorithmic paradigm–offline or online. Amanatidis et al. [2] have recently authored an extensive survey on fair division that describes the research on the topic from the perspective of theoretical computer science.

2 PRELIMINARIES

Graph Theoretic Notations: For the graph theoretic notations, not defined here, we refer to Diestel's [18]. Let G be a graph with vertex set $V(G)$ and edge set $E(G)$. A subset $S \subseteq V(G)$ is called an *independent set* (resp. *clique*) in G if the subgraph induced by S is edgeless (resp. complete).

Consider sets A, B, and C such that $A \subseteq B$. If $f : B \mapsto C$ is a function then the *restriction* of f to A, denoted as $f|_A$, is a function $f|_A : A \mapsto C$ defined as $f|_A(x) = f(x)$ for all $x \in A$. Let G be a graph. For a subset $S \subseteq V(G)$, k subsets $X_1, X_2, \ldots, X_k$ of S are called *k-independent partition* of $V(G)$ if $\bigcup_{i\in[k]} X_i = S$ and for each $i \in [k]$, X_i is an independent set in $G[S]$. A function $f : V(G) \mapsto [k]$ is called a *k-coloring* of G, if for any edge $uv \in E(G)$, $f(u) \neq f(v)$. Let $A, B \subseteq V(G)$ such that $A \subseteq B$. Let f and g be a k-coloring of $G[A]$ and $G[B]$, respectively. We say that coloring g *agrees* with coloring f on A if $g|_A = f$. More precisely, $g(u) = f(u)$ for all $u \in A$.

For an instance $\mathcal{I} = (G, k, p, c, P, B)$ of BCFEA, we denote the total cost and utility of items to be $\gamma = \sum_{v\in V} c(v)$ of V and $\alpha = \sum_{v\in V} p(v)$, respectively. We say that a partition $(S_1, \ldots, S_k)$ *satisfies the cost and profit constraint* (P, B) if for each $i \in [k]$, $c(S_i) \leq B$ and $p(S_i) \geq P$. Moreover, if each S_i is an independent set in G, then call each S_i a *bundle*. We define the *type* of $\mathcal{I}$ to be $|\{(c_i, p_i) \mid \forall i \in [k]\}|$, denoted by λ.

Observation 1. *In a given instance of* BCFEA *the underlying graph has clique size of at most k.*

3 OUR RESULTS

3.1 When the number of agents is two

In this section, we study BCFEA when $k = 2$. We show that BCFEA exhibits a dichotomy: when the conflict graph is disconnected it is NP-hard and it has a polynomial-time algorithm when it is connected. More interestingly, it is FPT with respect to the number of components. Furthermore, due to Remark 1, the problem is NP-hard for $k = 2$ and large profit values, however, in Theorem 4, we design a polynomial time algorithm when values are in unary.

We begin with the observation that for the case $k = 2$ a solution for BCFEA is a solution for 2-Coloring in the conflict graph. This, yields the following.

Observation 2. *When* $k = 2$*, the conflict graph in a* yes*-instance of* BCFEA *must be bipartite.*

Thus, for the case $k = 2$, the interesting case is when conflict graph is in fact bipartite. In light of Remark 1, we know that BCFEA is already known to be NP-hard when the conflict graph is edgeless. By slightly tweaking the reduction from Partition: by adding dummy items that create a matching with the "real" items, we can show that the hardness carries forward for the case when the conflict graph has edges but is still disconnected.

Theorem 1. *When* $k = 2$*,* BCFEA *is* NP-hard *when the conflict graph is disconnected.*

Proof sketch. Let $\mathcal{I} = (G, 2, p, c, P, B)$ denote an instance of BCFEA. It is easy to see that the certificate of the problem can be verified in polynomial time. To prove the hardness, we give a reduction from Partition to BCFEA. For each element i of partition, we create a K_2, say u_i, v_i with $c_{u_i} = s_i, p_{u_i} = 1$ and $c_{v_i} = p_{v_i} = 0$. The allocation must be done in such a way that exactly one of the vertex in any K_2 must be allocated to one agent and other one must be to another agent. By having the values of $B = \frac{\sum_{i \in [n]} s_i}{2}$ and $P = 1$, yields the correspondence between the solutions of Partition and BCFEA. ◻

Notwithstanding this hardness, we can show that BCFEA does admit a parameterized algorithm with respect to the number of components. This allows us to infer that when conflict graph is connected, it is infact polynomial-time solvable.

Theorem 2. *When* $k = 2$*,* BCFEA *admits an* FPT(r) *algorithm, where* r *denotes the number of components in the conflict graph.*

Proof sketch. Let $\mathcal{I} = (G, 2, p, c, P, B)$ denote an instance of BCFEA. By Observation 2, we know that G must be a bipartite graph. In each connected component C_i, we have bipartition (X_i, Y_i), by the property of BCFEA, we know that X_i must be allocated to one agent and Y_i to the other. For each connected component, we have two choices of allocating it to an agent. Thus we have 2^r possible ways to allocate the items to an agent. ◻

Corollary 3. *When* $k = 2$ *and the conflict graph is connected,* BCFEA *admits a polynomial-time algorithm.*

Hence, we can conclude that BCFEA exhibits a dichotomy with respect to connectivity: it is "hard" when the graph is disconnected and "easy" when it is connected. The next result shows that when the representation is in unary, we can solve BCFEA in polynomial-time. Alternately worded, this means that when $k = 2$, BCFEA is pseudo-polynomial time solvable. The following result is based on a path-style dynamic programming approach on a *layered graph.*

Theorem 4. *When* $k = 2$*,* BCFEA *admits an algorithm with running time* $O(nP^2B^2)$*. That is, it has a polynomial-time algorithm when values are in unary.*

Proof sketch. Let $\mathcal{I} = (G, 2, p, c, P, B)$ denote an instance of BCFEA. Since $k = 2$ we may assume that the conflict graph G is bipartite but not connected. Suppose that it has r components. Then each of the components is in fact a bipartite graph, denoted by $C_i = (X_i, Y_i)$, for each $i \in [r]$. We define a graph H with r layers, with layer i consisting of vertices $\{x_i, y_i\}$ and H containing directed arcs between consecutive layers. Specifically, there are a red and blue arcs from x_i to both x_{i+1} and y_{i+1}; and from y_i to x_{i+1} and y_{i+1}. We create a special vertex s and connect it to the vertices of layer 1. For each layer $i \in [r]$, and $z_i \in \{x, y\}$, each of the arcs to the vertex z_i have utility $\sum_{x \in z_i} u(x)$ and cost $\sum_{x \in z_i} c(x)$.

The objective is to check if there exist r-length vertex disjoint red and blue paths starting from s whose total utility is at least P and cost is at most B. This condition can be tested via a dynamic programming based algorithm which for layers $i \in [r]$, values $\tau_1, \tau_2 \in [\alpha]$, $\kappa_1, \kappa_2 \in [\beta]$ and a flag $f \in \{x, y\}$ computes a Boolean function $M[i, \tau_R, \kappa_R, \tau_B, \kappa_B, f]$ which is true if and only if there exist i-length vertex disjoint red and blue paths starting from s such that the total utility along the red(blue) path is at least $\tau_R(\tau_B)$ and cost is at most κ_R (κ_R) with the red path ending at f_i.

Assuming that this function can be computed, the answer to the instance of BCFEA can be found if $\vee_{f \in \{x,y\}} M[r, P, B, P, B, f]$ is true. ◻

3.2 When the number of agents is arbitrary

As discussed in the Introduction, interval graphs are a natural setting to study Job Scheduling, a problem that is a special case of BCFEA. In this section we present a result, Corollary 6, which implies that when the conflict graph is an interval graph, BCFEA has pseudo-polynomial time algorithm. Specifically, we show that chordal graphs have this property and the result is due to a simple reduction to the problem when conflict graph has bounded treewidth, for which we show that BCFEA admits an FPT algorithm with respect to $k + \text{tw}$ and is pseudo-polynomial on α and γ.

Theorem 5. BCFEA *admits an* FPT$(k + \text{tw})$ *algorithm when the values are encoded in unary.*

Proof. Let $\mathcal{I} = (G, k, p, c, P, B)$ be an instance of BCFEA, where k is constant and let $(T, \{\beta_t\}_{t \in V(T)})$ be a tree decomposition of conflict graph G with treewidth tw. We propose a dynamic programming based algorithm. Towards this end, we define the *PC-value* of a k-partition $(X_1, X_2, \ldots, X_k)$ of a set $S \subseteq V(G)$ to be a vector $((P_1, C_1), \ldots, (P_k, C_k))$ where for each $i \in [k]$, $P_i = \sum_{v \in X_i} p_v$ and $C_i = \sum_{v \in X_i} c_v$. In our algorithm, we compute all possible PC-values that can be obtained from a k-independent partition of G. To design the dynamic programming algorithm, we consider the nice tree decomposition of G. Let $(T, \{\beta_t\}_{t \in V(T)})$ be a nice tree decomposition of G of width tw. Recall that, in nice tree decomposition, each node of the tree T can be identified as a *leaf node, root node, introduce* *node, a forget node, or a join node.* For a node $t \in V(T)$, let T_t denotes the subtree of T rooted at t and $V_t = \bigcup_{t \in V(T_t)} \beta_t$. We traverse tree T bottom-up and use a dynamic programming approach to compute a "partial solution" for every node $t \in V(T)$ and every k-coloring f of $G[\beta_t]$. For a node $t \in V(T)$ and a k-coloring f of $G[\beta_t]$, the idea is to store the set of all PC-values, denoted by $\mathrm{PC}[t, f]$, that can be obtained from a k-coloring of $G[V_t]$ that agrees with f on β_t.

Note that for the root r, bag β_r is empty. Thus, every color class in a k-coloring of β_r is empty, as well. Since $G[V_r] = G$ and any k-coloring of $G[V_r]$ agrees with the coloring f of β_r, we can conclude that $\mathrm{PC}[r, f]$ contains all possible PC-values which can be obtained from a k-coloring of $V(G)$. Now, we compute $\mathrm{PC}[t, f]$ depending on the node type t.

(1) t **is the leaf (first) node:** Since $X_t = \emptyset$, each color class in a k coloring of X_t is an empty set. Hence,
$$\mathrm{PC}[t, f] = \{((0, 0), (0, 0), \dots (0, 0))\}.$$
(2) t **is an introduce node:** By definition, t has exactly one child, say t', such that $\beta_t = \beta_{t'} \cup \{v\}$ for some vertex $v \in V \setminus \beta_{t'}$. We compute $\mathrm{PC}[t, f]$ for a k-coloring f of β_t assuming that we have computed the values of $\mathrm{PC}[t', f']$ for all possible k-coloring f' of $\beta_{t'}$. Let f be a k-coloring of β_t and $f' = f_{|\beta'_t}$. We define,
$$\mathrm{PC}[t, f] = \{\mathrm{b} + \Delta_i(p_v, c_v) \mid \mathrm{b} \in \mathrm{PC}[t', f'], i = f(v)\},$$
where $\Delta_i(p_v, c_v)$ is a k-length vector whose ith entry is (p_v, c_v) and all other entries are $(0, 0)$.

(3) t **is a forget node:** t has exactly one child t', such that $\beta_t = \beta_{t'} \setminus \{v\}$ for some vertex $v \in \beta_{t'}$. We say a coloring f of β_t can be extendable to a coloring f' of $\beta_{t'}$ if $f'(u) = f(u)$ for all $u \in \beta_t$ and $f'(v) = i$ if $f^{-1}(i) \cap N_G[v] = \emptyset$. For a forget node t, we define
$$\mathrm{PC}[t, f] = \bigcup_{f \text{ is extendable to } f'} \mathrm{PC}[t', f'].$$
(4) t **is join node:** By definition of join node, t has exactly two children t_1 and t_2 such that $\beta_t = \beta_{t_1} = \beta_{t_2}$. For a coloring f of β_t, let $(A_1, A_2, \dots, A_k) = (f^{-1}(1), f^{-1}(2), \dots, f^{-1}(k))$ and denote $\tilde{b} = ((p(A_1), c(A_1)), (p(A_2), c(A_2)), \dots, (p(A_k), c(A_k)))$. For a join node t, we define,
$$\mathrm{PC}[t, f] = \{\mathrm{b}_1 + \mathrm{b}_2 - \tilde{b} \mid \mathrm{b}_1 \in \mathrm{PC}[t_1, f], \mathrm{b}_2 \in \mathrm{PC}[t_2, f]\}.$$

Finally, we solve the problem by checking if there is an element $((P_1, C_1), \dots, (P_k, C_k)) \in \mathrm{PC}[t, r]$ such that $P_i \geq P$ and $C_i \leq B$ for each $i \in [k]$. If such an element exists in $\mathrm{PC}[t, r]$ then we conclude that $\mathcal{I}$ is a yes-instance of BCFEA. The corresponding partition can be obtained by backtracking over the subproblems.

Correctness. We say that the entry $C[t, f]$ is computed *correctly* if it contains the set of all possible PC-values which can be obtained from a k-coloring of $G[V_t]$ that agrees with the k-coloring f on β_t. We begin our analysis by observing that the correctness of the recurrence for a leaf node and the root follows trivially from the definition. Hence, our main analysis will focus on the introduce, forget, and join nodes.

When t is a introduce node: For an introduce node t, we show that $\mathrm{PC}[t, f] = \{\mathrm{b} + \Delta_i(p_v, c_v) \mid \mathrm{b} \in \mathrm{PC}[t', f']\}$. Note that t has exactly one child t' such that $\beta_t = \beta_{t'} \cup \{v\}$, where $v \notin V_{t'}$. Consider a coloring f of β_t and let the introduced vertex v be colored i, that is, $f(v) = i$. Clearly, any k coloring h of V_t that agrees with the coloring f of β_t, also agrees with the coloring f' of β_t, where $f' = f|_{\beta_{t'}}$ ($f'(v) = f(v)$ for each $v \in \beta_{t'}$).
$$\mathrm{PC}[t, f] \subseteq \{\mathrm{b} + \Delta_i(p_v, c_v) \mid \mathrm{b} \in \mathrm{PC}[t', f'], i = f(v)\}.$$
Now, let f' be a k-coloring of $\beta_{t'}$ such that $f'^{-1}(i) \cap N_G(v) = \emptyset$. That is, no neighbor of v is colored i by f'. Also, using Theorem **??**, we note that v is not adjacent to any vertex $u \in V_t \setminus \beta_t$. Therefore, in this case, any k-coloring h of $V_{t'}$ that agrees with f' on $\beta_{t'}$, also agrees with the k coloring f of β_t where $f(u) = f'(u)$ for all $u \in \beta_t \setminus \{v\}$ and $f(v) = i$.
$$\{\mathrm{b} + \Delta_i(p_v, c_v) \mid \mathrm{b} \in \mathrm{PC}[t', f'], i = f(v)\} \subseteq \mathrm{PC}[t, f].$$
This proves that $C[t, f]$ has been computed correctly for an introduce node. Observe that, here we proved a one-to-one relation between the set of k coloring h of $G[V_t]$ that agrees with k coloring f of β_t and set of k coloring h' of $G[V_{t'}]$ that agrees with k coloring f' on $\beta_{t'}$, where $f' = f|_{\beta_{t'}}$.

Time complexity: Suppose the treewidth of the conflict graph G in the input instance is tw. Then there are at most $O(\mathrm{tw} \cdot n)$ nodes in nice tree decomposition. At each node t, we are considering all possible colorings of β_t. Thus, there are $k^{|\beta_t|}$ possible k-colorings of β_t. Furthermore, we observe that for any partition $(X_1, X_2, \dots, X_k)$ of V_t, we have, $(p(X_i), c(X_i)) \in \{0, 1, \dots, \alpha\} \times \{0, 1, \dots, \gamma\}$. Therefore, each set $\mathrm{PC}[t, f]$ has at most $(\alpha \cdot \gamma)^k$ entries. The adjacency checking can be done in constant time. The computation time of $\mathrm{PC}[t, f]$ at each node depends on the type of node. For leaf nodes the $\mathrm{PC}[t, f]$ can be computed in constant time. At introduce node t, the $\mathrm{PC}[t, f]$ is computed using already computed values of corresponding $\mathrm{PC}[t', f']$ which takes time at most $(\alpha \cdot \gamma)^k$. For computing $\mathrm{PC}[t, f]$ when t is a forget node, in $O(k)$-time, we first check for all the color class which does not contain any neighbor of v. Then we can compute $\mathrm{PC}[t, f]$ in time $O(k \cdot (\alpha \cdot \gamma)^k)$. At the join node for each entry of $\mathrm{PC}[t_1, f]$, we may have to go through all values in the set $\mathrm{PC}[t_2, f]$. Therefore, at join node t, $\mathrm{PC}[t, f]$ can be computed in time $O((\alpha \cdot \gamma)^{2k})$. Hence, the total running time of the algorithm is $O(\mathrm{tw}^{k+1} \cdot (\alpha \cdot \gamma)^{2k} \cdot n)$, where n denotes the number of vertices in the conflict graph.

This concludes that the BCFEA admits a pseudo-polynomial time algorithm when k is constant and the conflict graph in the input instance has bounded treewidth. □

Here are some nice observations that we can infer from Theorem 5. Let $\mathcal{I} = (G, k, p, c, P, B)$ be an instance of conflict graph, where $k = O(1)$ and G is a chordal graph. Let $(T, \{\beta_t\}_{t \in V(T)})$ be a tree decomposition of G. It is known that a graph G is a chordal graph if and only if it admits a tree decomposition where each bag induces a maximal clique in G, see [8]. Such tree a decomposition of a chordal graph G can be computed in linear-time [34]. Using these facts and Observation 1, we note that if the treewidth of G in $\mathcal{I}$ is more than $k - 1$, then $\mathcal{I}$ is a no-instance of BCFEA. Further, if the treewidth of G is less than k, then we can use the algorithm given in Theorem 5 to decide if $\mathcal{I}$ is a yes-instance of BCFEA. Moreover, when G is an interval graph (a subclass of chordal graph) the corresponding tree T in the tree decomposition of G is a path. Thus we do not have join node in case of interval graphs. From the discussion above, we have the following corollary.

Corollary 6. *Let $\mathcal{I} = (G, k, p, c, P, B)$ be an instance of* BCFEA. *When k is a constant and the utility and the cost values are in unary, we have the following:*

(a) *if G has constant treewidth,* BCFEA *admits a polynomial time algorithm;*
(b) *if G is chordal then* BCFEA *admits a polynomial time algorithm;*
(c) *if G interval graph then* BCFEA *admits an algorithm with an improved exponent in the running time;*

When the type is a constant. Note that Remark 1 implies that BCFEA cannot have an FPT(k + tw) or an FPT($k + \lambda$) algorithm. Hence, it is worthwhile to consider (k + tw + λ).

Theorem 7. *When the type is a constant,* BCFEA *admits an* FPT(k + tw) *algorithm.*

Let $\mathcal{I} = (G, k, p, c, P, B)$ be an instance of BCFEA. From Theorem 7, we infer that, if k and treewidth of conflict graph in $\mathcal{I}$, are constant then BCFEA can be solved in polynomial-time. As we discussed, if G is a chordal graph and $\text{tw}(G) \geq k$ then $\mathcal{I}$ is a no-instance of BCFEA. Furthermore if $\text{tw}(G) \leq k - 1$ then for constant k, BCFEA can be solved in polynomial-time. Thus we have the following corollaries.

Corollary 8. *When type is a constant and the conflict graph is chordal,* BCFEA

(a) *is* FPT *with respect to k; and*
(b) *when k is also a constant it can be solved in $O(n)$ time.*

Exact Exponential Algorithm. We will present an exact exponential algorithm that runs in time $O^\star(2^n)$. It is based on the principle of subset convolution and is implemented via polynomial algebra, specifically FFT based polynomial multiplication, which guarantees that we can multiply two polynomials of degree d in at most $O(d \log d)$ time,[31]. The basic idea is that we build our solution, $(S_1, \ldots, S_k)$, where each S_i is a piece, round by round, with each round keeping track of the subsets that could give rise to a piece in the (hypothetical) solution of BCFEA. More specifically, in the first round we define monomials that correspond to subsets of items that satisfy the utility and cost constraint and therefore could be the first piece in the solution; in the second round we define a polynomial in which each monomial represents a solution of the subproblem defined for two agents, and so on. After k rounds, we have a polynomial which is not identically zero if and only if there is a solution for BCFEA.

Theorem 9. *For any $k \geq 1$,* BCFEA *can be solved in $O^\star(2^n)$-time.*

Proof. We present an algorithm for BCFEA using the subset convolution technique in the specified time. Our objective is to compute a k-partition of the vertices of G such that each set S in the partition satisfies the following properties; (i) S is an independent set, (ii) $c(S) \leq B$, and (iii) $p(S) \geq P$. We define the following indicator function f such that $f : 2^V \to \{0, 1\}$. For a set $S \in 2^V$, $f(S) = 1$ if S is an independent set, $c(S) \leq B$, and $p(S) \geq P$, and 0, otherwise. We use polynomial multiplication technique to find such a k-partition of V. Let $V = \{v_1, \ldots, v_n\}$. For a subset $S \subseteq V$, a characteristic vector $\chi(S)$ is defined as follows, $\chi(S)[i] = 1$, if $u_i \in S$, and 0 otherwise.

Observe that for a set $S \subseteq V$, such that $\chi(S)$ is an n-length binary vector. Two strings S_1, and S_2 are said to be disjoint if $\forall i \in [n]$, $\chi(S_1)[i] \neq \chi(S_2)[i]$. The *Hamming weight* of a binary string is the number of ones in the binary representation. For a binary string S, let $\mathtt{Ham}(S)$ denote the Hamming weight of S. The Hamming weight of a monomial x^i is the Hamming weight of i (expressed as a binary vector). The following lemma captures the relationship between disjointness of sets and Hamming weights.

Lemma 1. [10] *Subsets $S_1, S_2 \subseteq V$ are disjoint if and only if Hamming weight of $\chi(S_1) + \chi(S_2)$ is $|S_1|+|S_2|$.*

We use $\mathtt{Ham}_l(F(X))$ to denote the *Hamming projection* of a polynomial $F(x)$ to s which is the sum of all monomials in $F(X)$ having Hamming weight s. We use $\mathcal{R}(F(X))$ to denote the *representative polynomial* of $F(X)$, the coefficient of the monomial is one if the coefficient of the monomial is non-zero. Given an instance $\mathcal{I} = (G, k, p, c, P, B)$, our objective is to find a polynomial which is non-zero if and only if $\mathcal{I}$ is a yes-instance. Our algorithm finds the polynomial iteratively. We define a polynomial $F_s^1(x)$ of type 1 as follows for each $s \in [n]$,

$$F_s^1(x) = \sum_{Y \subseteq U, |Y|=s} f(Y) x^{\chi(Y)}$$

Observe that the polynomial $F_s^1(x)$ contains all subsets of fixed size s for which the indicator function f is satisfied. Further, all the polynomials in $F^1(x) = \{F_s^1(x) \mid s \in [n]\}$ together contain information about all possible subsets of V that are independent and satisfy the utility and cost constraints. For each $s \in [n]$, we define the polynomials of type $j \in [k] \setminus [1]$ as follows

$$F_s^j(x) = \sum_{s_1, s_2 \in [n] : s_1 + s_2 = s} \mathcal{R}(\mathtt{Ham}_s(F_{s_1}^1(x) \times F_{s_2}^{j-1}(x)))$$

Thus, a polynomial of type j is obtained by multiplying polynomial of type 1 with polynomial of type $j - 1$. Observe that $\mathtt{Ham}(\cdot)$ function ensures that the subsets corresponding to each monomial are formed by disjoint union of smaller set, and $\mathcal{R}(\cdot)$ function ensures that coefficients of all non-zero monomials is one. The solution to $\mathcal{I}$ reduces to checking whether $F_n^k(x)$ is a non-zero polynomial. If $F_n^k(x)$ is a non-zero polynomial, then the given instance is a yes-instance; otherwise a no-instance. □

We know from the following influential result that k-Coloring has a lower bound unless ETH fails.

Proposition 1. [27] *Unless* ETH *fails, k-*Coloring *cannot be solved in $2^{o(n)}$ time.*

Thus, Theorem 9 is asymptotically tight.

Parameterized Algorithm. Due to Remark 1, we cannot expect an FPT algorithms with respect to parameters $k + P + B$, or s, where s denotes the maximum bundle size. However, as shown in the next result, we can expect one with respect to $k + s$. This is because $ks \geq n$ and so an exhaustive search yields an FPT($k + s$) algorithm.

Theorem 10. BCFEA *admits an* FPT($k + s$) *algorithm.*

For the special case when a bundle size is at most two, we can reduce the question to finding a matching in an auxiliary graph, where each edge represents a feasible bundle that satisfies the profit and cost constraints. Hence, the problem is polynomial-time solvable, in contrast to the hardness for the case $s = 3$, Remark 1.

Theorem 11. *When* $s = 2$, BCFEA *admits a polynomial-time algorithm.*

Approximation Algorithm. Next, we present an FPTAS for BCFEA by repurposing the dynamic programming approach of Theorem 5 and the standard bucketing technique to split the range of the utility and cost values into smaller "buckets" so as to approximately estimate the value of each subproblem. Specifically, we show that for any positive real numbers $\epsilon, \omega > 0$ and a k-independent partition $(X_1, \ldots, X_k)$ with PC-value $((P_1, C_1), \ldots, (P_k, C_k))$ obtained in Theorem 5, we can obtain an approximate PC-value $((\widehat{P_1}, \widehat{C_1}), \ldots, (\widehat{P_k}, \widehat{C_k}))$ such that $P_j \leq (1+\epsilon)\widehat{P_i}$ and $\widehat{C_j} \leq (1+\omega)C_i$ for each $j \in [k]$.

Theorem 12. BCFEA *admits an algorithm that in* FPT$(k + \text{tw})$ *time outputs a solution in which every agent has profit at least* $P/(1+\epsilon)$ *and cost at least* $(1+\omega)B$.

Proof. Let $\mathcal{I} = (G, k, p, c, P, B)$ be an instance of BCFEA, where k is a constant. Recall that α and γ are the aggregate utility and cost of all the items, respectively. For a constant k, the pseudo-polynomial algorithm in Theorem 5 works as follows. At a node t and for a coloring f of β_t, the defined subproblem PC$[t, f]$ is the set of all possible PC-values of every k-coloring of $G[V_t]$ that agrees with f on β_t. Thus, at the root node r, we have all possible PC-values of any k-coloring of G. Then we check which all PC-values satisfy the utility and cost constraints. If none of them meet the constraints, then $\mathcal{I}$ is a no-instance.

Indeed, at a node t, we may be updating $(\tau\kappa)^k$ (at most) PC-values for coloring f of $G[\beta_t]$ which is a primary challenge in achieving a polynomial-time algorithm. Consequently, at a node t and coloring f of β_t, we store approximate PC-values of a k-independent partition of $G[V_t]$ that agrees with f on β_t. Towards the end, for positive real numbers $\epsilon, \omega > 0$ and for any k-independent partition $(X_1, \ldots, X_k)$ with PC-value $((P_1, C_1), \ldots, (P_k, C_k))$, we want to obtain a *relaxed* PC-value $((\widehat{P_1}, \widehat{C_1}), \ldots, (\widehat{P_k}, \widehat{C_k}))$, such that $P_j \leq (1+\epsilon)\widehat{P_i}$ and $\widehat{C_j} \leq (1+\omega)C_i$ for each $j \in [k]$.

To achieve this, we divide the range of utility and cost value. Note that, in k-independent partition $(X_1, X_2, \ldots, X_k)$ of $V(G)$, the utility and the cost of a set X_i ranges from 0 to τ and 0 to κ, respectively. For a given positive value $0 < \epsilon \leq 1$, we split the utility value of each set X_i into intervals whose endpoints vary by a factor of $(1+\epsilon)$:

$$(0, (1+\epsilon)^{\frac{1}{n}}], \ldots, ((1+\epsilon)^{\frac{i-1}{n}}, (1+\epsilon)^{\frac{i}{n}}], \ldots$$

where, $i \leq \lceil n \log_{1+\epsilon} \alpha \rceil$. Similarly, for $\omega \in (0, 1)$, we split the cost value of each set X_i into $s = \lceil n \cdot \log_{(1+\omega)} \gamma \rceil$ intervals whose end points vary by a factor of $(1+\omega)$:

$$(0, (1+\omega)^{\frac{1}{n}}], \ldots, ((1+\omega)^{\frac{i-1}{n}}), (1+\omega)^{\frac{i}{n}})], \ldots$$

Note that at an introduce node t, we look for the possibility of assigning vertex v to some set in a k-partition $(X_1, X_2, \ldots, X_k)$ of V_t and the PC-value corresponding to this partition gets updated. Let $((P_1, C_1), \ldots, (P_k, C_k))$ denote the PC value before assigning v to the set X_j. Thus, after update, the resulting PC-value is $(\ldots, (P_j + p_v, C_j + c_v), \ldots, (P_k, C_k))$. Similar arguments hold for the other nodes as well. For every operation, we can pin-point the sets in the PC value that is affected. This allows us to bound the change in value of the PC-values in an approximate manner.

Specifically, for the introduce node, the updated utility of X_j is rounded down to the nearest interval endpoint to which the value below belongs. That is, suppose that $P_j + p_v \in [(1+\epsilon)^{\frac{i}{n}}, (1+\epsilon)^{\frac{i+1}{n}})$ for some $i \leq \lceil n \log_{(1+\epsilon} \tau \rceil$, then the approximate utility value is at least $(1+\epsilon)^{\frac{i}{n}}$. Similarly, the updated cost value is rounded up to the nearest interval endpoint. Hence, $C_j + c_v$ is at most $(1+\omega)^{\frac{i'}{n}}$ for some $i' \leq \lceil n \log_{(1+\omega)} \gamma \rceil$. Similar argument hold for the forget and join nodes as well.The utility (and cost) values of the sets that are affected by our operation can rounded down (and up), as described above. Through the next claim, we establish a relationship between the standard PC-value computed by the exact algorithm (of Theorem 5) and the relaxed PC-value computed by the approximation algorithm at every step. The factor of $(1+\epsilon, 1+\omega)$ follows as a result.

Claim 1. *Consider a PC-value,* $((P_1, C_1), \ldots, (P_k, C_k))$, *generated by the algorithm Theorem 5 after assigning the first* $i - 1$ *vertices. Suppose that the* i^{th} *vertex affects the* j^{th} *set. Then, there exists a relaxed PC-value* $((\widehat{P_1}, \widehat{C_1}), \ldots, (\widehat{P_k}, \widehat{C_k}))$ *such that* $P_j \leq (1+\epsilon)^{i/n}\widehat{P_j}$ *and* $\widehat{C_j} \leq (1+\omega)^{i/n}C_j$.

Note that each set PC$[t, f]$ in the approximation algorithm is consists of at most $r \cdot s$ entries. Thus the running time of the algorithm is $O(\text{tw}^k \cdot (rs)^{2k} \cdot n)$, where where, $r = \lceil n \log_{1+\epsilon} \alpha \rceil$ and $s = \lceil n \cdot \log_{(1+\omega)} \gamma \rceil$. Hence, when k is a constant, the result follows from Theorem 5 and Claim 1. □

Corollary 13. *When* k *is a constant and the conflict graph has constant treewidth,* BCFEA *admits an* FPTAS.

3.3 About general utility and cost functions

Throughout the paper, we have presented algorithms for identical utility and cost functions (between agents). These results can be extended to arbitrary additive utility and cost functions. Consider the polynomial-time algorithm in Theorem 5. In case of arbitrary additive utility and cost functions we replace α with $\widehat{\alpha} = \max_{i \in [k]} \max\{p_i(j) | j \in [n]\}$ and γ with $\widehat{\gamma} = \max_{i \in [k]} \max\{c_i(j) | j \in [n]\}$. Moreover, while updating the PC-values of a k-independent partition $(X_1, X_2, \ldots, X_k)$ of V_t at any node t, to update the utility and cost of X_i we use the valuation and cost function of agent j. Similarly, in Theorem 7, we can redefine the *type* of an instance as the union of set of different utility and cost pairs of each agent. Then, the type of elements in X_j can be updated according to the functions of agent j. Thus, the results of Theorems 5 and 7 and their corollaries can be extended to general valuation and cost functions.

FUTURE DIRECTION

We initiated the study of conflict-free fair division under budget constraints. This has applications to various real-life scenarios. We considered the egalitarian fair division as the fairness criteria for our study. Below we describe few future research directions that we view as important and promising: (i) other fairness notions, such as envy-freeness, NSW, maximin, pareto optimalility, etc, (ii) dependencies between the items, e.g connectivity (iii) two-sided preferences.

Acknowledgement: Sushmita Gupta, Pallavi Jain, and Vikash Tripathi acknowledge the support from SERB-SUPRA grant number SPR/2021/000860. Sushmita Gupta acknowledges support from SERB's MATRICS grant (MTR/2021/000869). Pallavi Jain acknowledges support from IITJ Seed Grant grant I/SEED/PJ/20210119.

REFERENCES

[1] M. M. Ahmadian, M. Khatami, A. Salehipour, and T. C. E. Cheng. 2021. Four decades of research on the open-shop scheduling problem to minimize the makespan. *European Journal of Operational Research* 295, 2 (2021), 399–426.

[2] G. Amanatidis, H. Aziz, G. Birmpas, A. Filos-Ratsikas, Bo. Li, H. Moulin, A. A. Voudouris, and X. Wu. 2023. Fair Division of Indivisible Goods: Recent Progress and Open Questions. *Artificial Intelligence* 322 (2023).

[3] N. Bansal and M. Sviridenko. 2006. The Santa Claus problem. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing (STOC)*. 31–40.

[4] S. Barman, A. Khan, S. Shyam, and K. V. N. Sreenivas. 2023. Finding Fair Allocations under Budget Constraints. In *Proceedings of the 37th AAAI Conference on Artificial Intelligence (AAAI)*. 5481–5489.

[5] S. Barman and P. Verma. 2021. Existence and Computation of Maximin Fair Allocations Under Matroid-Rank Valuations. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*. 169–177.

[6] I. Bezáková and V. Dani. 2005. Allocating indivisible goods. *ACM SIGecom Exchanges* 5, 3 (2005), 11–18.

[7] A. Biswas, Y. Ke, S. Khuller, and Q. C. Liu. 2023. An Algorithmic Approach to Address Course Enrollment Challenges. In *4th Symposium on Foundations of Responsible Computing, FORC 2023, (LIPIcs, Vol. 256)*. 8:1–8:23.

[8] J. R. S. Blair and B. Peyton. 1993. An introduction to chordal graphs and clique trees. In *Graph theory and sparse matrix computation*. IMA Vol. Math. Appl., Vol. 56. Springer, 1–29.

[9] S. Bouveret, K. Cechlárová, E. Elkind, A. Igarashi, and D. Peters. 2017. Fair Division of a Graph. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*. 135–141.

[10] Marek C. and Marcin P. 2010. Exact and approximate bandwidth. *Theoretical Computer Science* 411, 40 (2010), 3701–3713.

[11] S. Cheng and Y. Mao. 2022. Restricted Max-Min Allocation: Integrality Gap and Approximation Algorithm. *Algorithmica* (2022), 1–40.

[12] N. Chiarelli, M. Krnc, M. Milanič, U. Pferschy, N. Pivač, and J. Schauer. 2022. Fair Allocation of Indivisible Items with Conflict Graphs. *Algorithmica* (2022), 1–31.

[13] E. G. Coffman Jr., J. Csirik, G. Galambos, S. Martello, and D. Vigo. 2013. *Bin Packing Approximation Algorithms: Survey and Classification*. Springer, 455–531.

[14] M. Cygan, F. V. Fomin, Ł. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. 2015. *Parameterized algorithms*. Springer.

[15] A. Darmann, U. Pferschy, J. Schauer, and G. J. Woeginger. 2011. Paths, trees and matchings under disjunctive constraints. *Discrete Applied Mathematics* 159, 16 (2011), 1726–1735.

[16] A. Deligkas, E. Eiben, R. Ganian, T. Hamm, and S. Ordyniak. 2021. The Parameterized Complexity of Connected Fair Division.. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*. 139–145.

[17] B. L. Deuermeyer, D. K. Friesen, and M. A. Langston. 1982. Scheduling to maximize the minimum processor finish time in a multiprocessor system. *SIAM Journal on Algebraic Discrete Methods* 3, 2 (1982), 190–196.

[18] R. Diestel. 2012. *Graph Theory, 4th Edition*. Graduate texts in mathematics, Vol. 173. Springer.

[19] J. Dippel, M. D. l. Tour, April Niu, and A. Vetta. 2023. Gerrymandering Planar Graphs. arXiv:2312.14721 [cs.GT]

[20] S. Ebadian, D. Peters, and N. Shah. 2022. How to Fairly Allocate Easy and Difficult Chores. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 372–380.

[21] L. Epstein and A. Levin. 2008. On bin packing with conflicts. *SIAM Journal on Optimization* 19, 3 (2008), 1270–1298.

[22] G Even, M. M. Halldórsson, L. Kaplan, and D. Ron. 2009. Scheduling with conflicts: Online and offline algorithms. *Journal of Scheduling* 12, 2 (2009), 199–224.

[23] P. Factorovich, I. Méndez-Díaz, and P. Zabala. 2020. Pickup and delivery problem with incompatibility constraints. *Computers & Operations Research* 113 (2020), 104805, 17.

[24] M.R. Garey and D.S. Johnson. 1979. *Computers and intractability*. Vol. 174. freeman.

[25] J. Garg, M. Hoefer, and K. Mehlhorn. 2018. Approximating the Nash Social Welfare with Budget-Additive Valuations. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2326–2340.

[26] H. Hummel and M. L. Hetland. 2022. Fair allocation of conflicting items. *Autonomous Agents and Multi-Agent Systems* 36, 1 (2022), 1–33.

[27] R. Impagliazzo, R. Paturi, and F. Zane. 2001. Which Problems Have Strongly Exponential Complexity? *J. Comput. System Sci.* 63, 4 (2001), 512–530.

[28] H. Kellerer, U. Pferschy, and D. Pisinger. 2004. *Knapsack problems*. Springer-Verlag. xx+546 pages.

[29] D. Kurokawa, A. D. Procaccia, and J. Wang. 2018. Fair enough: Guaranteeing approximate maximin shares. *Journal of the ACM (JACM)* 65, 2 (2018), 1–27.

[30] J. Madathil. 2023. Fair Division of a Graph into Compact Bundles. arXiv:2305.06864 [cs.GT]

[31] R. T. Moenck. 1976. Practical fast polynomial multiplication. In *Proceedings of SYMSAC'76*. 136–148.

[32] U. Pferschy and J. Schauer. 2009. The knapsack problem with conflict graphs. *Journal of Graph Algorithms and Applications* 13, 2 (2009), 233–249.

[33] U. Pferschy and J. Schauer. 2017. Approximation of knapsack problems with conflict and forcing graphs. *Journal of Combinatorial Optimization* 33, 4 (2017), 1300–1323.

[34] J. P. Spinrad. 2003. *Efficient Graph Representations.: The Fields Institute for Research in Mathematical Sciences*. Vol. 19. American Mathematical Society.

[35] H. Steinhaus. 1948. The problem of fair division. *Econometrica* 16 (1948), 101–104.

[36] W. Suksompong. 2021. Constraints in fair division. *ACM SIGecom Exchanges* 19, 2 (2021), 46–61.

[37] G. J. Woeginger. 1997. A polynomial-time approximation scheme for maximizing the minimum machine completion time. *Operations Research Letters* 20, 4 (1997), 149–154.