Fair Interval Scheduling of Indivisible Chores

Sarfaraz Equbal
IIT Bombay
sequbal@cse.iitb.ac.in

Rohit Gurjar
IIT Bombay
rgurjar@cse.iitb.ac.in

Yatharth Kumar
IIT Delhi
cs1200413@iitd.ac.in

Swaprava Nath
IIT Bombay
swaprava@cse.iitb.ac.in

Rohit Vaish
IIT Delhi
rvaish@iitd.ac.in

Abstract

We study the problem of fairly assigning a set of discrete tasks (or chores) among a set of agents with additive valuations. Each chore is associated with a start and finish time, and each agent can perform at most one chore at any given time. The goal is to find a fair and efficient schedule of the chores, where fairness pertains to satisfying *envy-freeness up to one chore* (EF1) and efficiency pertains to *maximality* (i.e., no unallocated chore can be feasibly assigned to any agent). Our main result is a polynomial-time algorithm for computing an EF1 and maximal schedule for two agents under *monotone* valuations when the conflict constraints constitute an *arbitrary* interval graph. The algorithm uses a coloring technique in interval graphs that may be of independent interest. For an arbitrary number of agents, we provide an algorithm for finding a fair schedule under identical dichotomous valuations when the constraints constitute a path graph. We also show that stronger fairness and efficiency properties, including envy-freeness up to any chore (EFX) along with maximality and EF1 along with Pareto optimality, cannot be achieved.

1 Introduction

Fair allocation of indivisible resources has become a significant area of study within economics, operations research, and computer science [Brams and Taylor, 1996, Brandt et al., 2016, Moulin, 2019]. The main objective is to distribute a set of discrete resources among agents with differing preferences such that the outcome satisfies rigorous guarantees of fairness and economic efficiency. This field has generated extensive theoretical and practical interest in recent years. On the theoretical front, multiple fairness notions have been formulated and a rich set of algorithmic techniques have emerged [Amanatidis et al., 2023]. As for practical applications, there are various areas of use, such as course allocation [Budish et al., 2017], public housing [Benabbou et al., 2020], and inheritance division [Goldman and Procaccia, 2015].

It is worth noting that while the aforementioned settings involve resources that are considered *desirable* (also known as *goods*), there are many real-world situations where fair distribution of resources that are *undesirable*, also known as *chores*, is needed [Gardner, 1978]. Common examples of such situations include the division of household tasks like cooking or cleaning [Igarashi and Yokoyama, 2023], as well as the distribution of responsibilities for tackling global issues like climate change among countries [Traxler, 2002].

The problem of fair division of indivisible chores involves a set of discrete resources for which agents have non-positive values. The goal is to assign each chore to exactly one agent such that the final allocation is fair. A well-studied notion of fairness is *envy-freeness* [Gamow and Stern, 1958, Foley, 1967] which requires that each agent weakly prefers its bundle over any other agent's. However, due to the discrete nature of the tasks, an envy-free allocation may not always exist. This has led to the study of approximations such as *envy-freeness up to one chore* (EF1) which bounds the pairwise envy by the removal of some chore in the envious agent's bundle [Budish, 2011, Aziz et al., 2019]. Unlike exact envy-freeness, an EF1 allocation of chores is guaranteed to exist even under general monotone valuations [Lipton et al., 2004, Bhaskar et al., 2021].

A common assumption in the fair division literature is that any item can be feasibly assigned to any agent. This assumption may not hold in many settings of interest. For example, in course allocation, a student can only attend at most one course at any given time. Similarly, in assigning volunteers to conference sessions, temporal overlaps may need to be taken into account. In such settings, it is more natural to model *conflicts* among the items and allow only feasible (or non-conflicting) allocations.

We formalize the problem of fair and efficient scheduling of indivisible chores under conflict constraints. Each chore is associated with a start time and a finish time. Indivisibility dictates that a chore can be assigned to at most one agent. An agent can perform at most one chore at a time; furthermore, a chore once started must be performed until its completion. By modeling the chores as vertices of a graph and capturing temporal conflicts with edges, we obtain the problem of dividing the vertices of an *interval* graph among agents such that each agent gets an independent subset. Note that due to conflicts, it may not be possible to allocate all chores. Thus, we ask for schedules to be *maximal*, i.e., it should not be possible to assign any agent an unallocated chore without creating a conflict.

Our Contributions

We initiate the study of fair and efficient interval scheduling of indivisible chores under conflict constraints and make the following contributions:

Non-existence results. In Section 3, we show that the strongest approximation notion—envy-freeness up to *any* chore (EFX)—may not be compatible with maximality. By weakening the fairness requirement to envy-freeness up to one chore (EF1) but strengthening the efficiency requirement to Pareto optimality, we again obtain a non-existence result (see Figure 1). Notably, our negative results hold even for two agents with identical valuations and even when the conflict graph is a path graph. Thus, we focus on EF1 and maximality in search of positive results. In Section 4, we highlight the limitations of algorithms from the unconstrained setting—specifically, round robin and envy-cycle elimination—in computing an EF1 and maximal schedule.

Algorithms for two agents. In Section 5, we prove our main result: A polynomial-time algorithm for finding an EF1 and maximal schedule for two agents under general *monotone* valuations and for *any* interval graph. Our analysis develops a novel notion of *adjacent* schedules and uses a *coloring* technique that may be of independent interest.

Algorithms for an arbitrary number of agents. In Section 6, we consider the case of an arbitrary number of agents. While we are unable to settle the existence of an EF1 and maximal schedule in this setting even for three agents, we show that under *restricted* valuations (specifically, identical dichotomous valuations), an EF1 and maximal schedule always exists for a path graph for four or

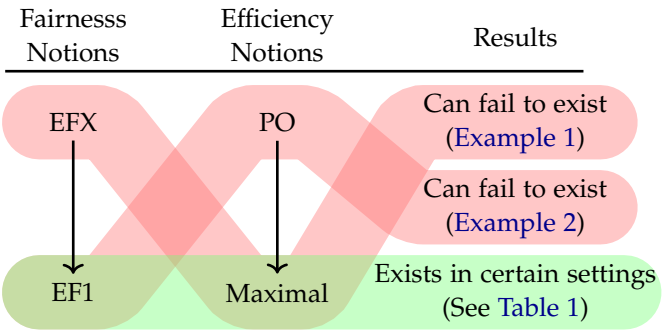


Figure 1: *Summary of our results. The arrows denote logical implications between fairness and efficiency notions. The positive and negative results are shown in green and red, respectively.*

No. of agents	Path graph	Interval graph
$n = 2$	✓ for monotone valuations (Theorem 1)	✓ for monotone valuations (Theorem 3)
arbitrary n	✓ for identical dichotomous valuations and $n \geq 4$ (Theorem 4)	✓ for identical valuations and bounded components (Theorem 5)

Table 1: *Summary of our results for* EF1 *and maximality. In each cell, a* ✓ *denotes that an* EF1 *and maximal schedule always exists and is computable in polynomial time under the assumptions on the number of agents (rows) and the conflict graph (column).*

more agents.[1] Furthermore, for identical valuations (not necessarily dichotomous), we show that EF1 and maximality can be simultaneously achieved for a more general class of graphs, namely, any graph in which each connected component is of size at most n, where n is the number of agents.

Related Work

Fair allocation of indivisible resources has been predominantly studied in the *unconstrained* setting wherein there are no feasibility constraints on the resources. This problem is a special case of our model when the conflict graph contains no edges. For indivisible *goods*, it is known that an allocation that is envy-free to up one good (EF1) always exists. Furthermore, the round robin algorithm returns such an allocation in polynomial time under additive valuations, while the envy-cycle elimination algorithm finds an EF1 allocation for the larger class of monotone valuations [Lipton et al., 2004]. Additionally, under additive valuations, an allocation that is simultaneously EF1 and Pareto optimal (PO) is known to always exist [Caragiannis et al., 2019] and such an allocation can be computed in pseudopolynomial time [Barman et al., 2018]. The existence of an allocation satisfying envy-freeness up to any good (EFX) remains unresolved for four or more agents [Chaudhury et al., 2020]; though, it is known that EFX can be incompatible

[1]For two agents, our result in Section 5 applies while the three agent case remains unresolved even for identical dichotomous valuations and a path graph.

with Pareto optimality [Plaut and Roughgarden, 2020].

For indivisible *chores*, an allocation satisfying envy-freeness up to one chore (EF1) is again known to exist under monotone valuations [Bhaskar et al., 2021]. However, for additive valuations, it is not known if EF1 and Pareto optimality can be simultaneously achieved for four or more agents [Garg et al., 2023] or if envy-freeness up to any chore (EFX) can be satisfied for three or more agents.[2] Just like goods, EFX and Pareto optimality are known to be incompatible for chores [Plaut and Roughgarden, 2020].

A growing line of research in fair division has studied *feasibility constraints* on the resources [Suksompong, 2021]. For example, several works model the items as vertices of a graph and require each agent's bundle to constitute a connected subgraph [Bouveret et al., 2017, 2019, Bilò et al., 2022]. By contrast, our work requires the bundles to be independent sets.

Hummel and Hetland [2022] study fair allocation of items on a graph when each agent gets an independent subset of the vertices. Their model differs from ours in two ways: First, they require *complete* allocations whereas we focus on maximal allocations.[3] Secondly, the items in their model are goods whereas we consider chores. Chiarelli et al. [2023] study a similar model as Hummel and Hetland [2022] but allow for partial allocations. However, they focus on maximizing the egalitarian welfare (i.e., maximizing the minimum utility) as opposed to satisfying approximate envy-freeness. Biswas et al. [2023] generalize the model of Hummel and Hetland [2022] by incorporating capacity constraints for agents and items.

Assigning independent subsets to agents naturally corresponds to a partial coloring of the conflict graph. A seminal result in this line of work is the *Hajnal-Szemerédi theorem* [Hajnal and Szemerédi, 1970], which states that any graph with maximum degree Δ admits an equitable coloring when there are at least $\Delta + 1$ colors.[4] The distinguishing point with our work is that this result only focuses on equalizing the *number* of vertices in each color class, whereas our model accounts for the (possibly differing) *valuations* assigned by the agents to the vertices.

Our work is closely related to that of Li et al. [2021], who studied fair scheduling in the context of indivisible *goods* (as opposed to chores). Their framework is somewhat more general than ours as they allow for *flexible* intervals, that is, the processing time of a job can be strictly less than its finish time minus its start time. They study approximate maximin fair share (MMS) and EF1 notions. Notably, they show that a schedule maximizing Nash social welfare (i.e., the geometric mean of agents' utilities) satisfies $1/4$-EF1 and Pareto optimality. For the case of chores, however, maximizing or minimizing the product of (absolute value of) agents' utilities can be shown to violate EF1 or Pareto optimality.

In the literature on scheduling problems [Leung, 2004], various other fairness criteria have been studied such as minimizing the maximum deviation from a desired load [Ajtai et al., 1998], minimizing the ℓ_p norm of flow times [Im and Moseley, 2020], and analyzing the welfare degradation due to imposition of fairness constraints [Bilò et al., 2016]. These notions differ from the "up to one item" style approximations studied in our work.

[2] Though, positive results are known for dichotomous valuations [Garg et al., 2022, Ebadian et al., 2022, Zhou and Wu, 2022].

[3] In our model, a complete allocation may fail to be EF1 even for a star graph.

[4] An *equitable* coloring is a proper coloring in which the color classes are almost balanced. That is, each vertex is assigned a color such that no pair of adjacent vertices have the same color and any two color classes differ in size by at most one.

2 Preliminaries

Given any $r \in \mathbb{N}$, let $[r] \coloneqq \{1, 2, \ldots, r\}$.

Problem instance. An instance of the *chore scheduling problem* (CSP)[5] is given by a tuple $\langle \mathcal{A}, \mathcal{C}, \mathcal{T}, \mathcal{V} \rangle$, where $\mathcal{A} \coloneqq \{a_1, \ldots, a_n\}$ is the set of n *agents* (or machines), $\mathcal{C} \coloneqq \{c_1, \ldots, c_m\}$ is the set of m indivisible *chores* (or jobs), $\mathcal{T}$ is the set containing the *timing information* of all chores, and $\mathcal{V}$ is the *valuation profile*. The sets $\mathcal{T}$ and $\mathcal{V}$ are formally defined below.

Timing information. We will model the time axis as consisting of disjoint units $[0, 1)$, $[1, 2)$, $[2, 3)$, and so on. Any fixed $t \in \mathbb{N} \cup \{0\}$ denotes a *time instant* and the interval $[t, t+1)$ refers to the $(t+1)^{\text{th}}$ *time slot*. Each chore $c_j \in \mathcal{C}$ is associated with a *start time* $s_j \in \mathbb{N} \cup \{0\}$ and a *finish time* $f_j \in \mathbb{N}$ ($f_j > s_j$). The set $\mathcal{T}$ contains the tuple (s_j, f_j) for every chore $c_j \in \mathcal{C}$, i.e., $\mathcal{T} \coloneqq \{(s_j, f_j)_{c_j \in \mathcal{C}}\}$.

Feasibility. An agent can perform at most one chore in any time slot. A chore c_j is performed successfully if it is performed by an agent during the interval $[s_j, f_j)$. A set of chores $C \subseteq \mathcal{C}$ is said to be *feasible* for an agent a_i if all chores in C can be performed successfully by a_i without overlap. We will write $\mathcal{F}_i$ to denote the *feasibility set* of agent a_i, i.e., the set of all feasible subsets of chores for agent a_i. Observe that any subset of pairwise non-overlapping chores is feasible for any agent; thus, we have that $\mathcal{F}_1 = \mathcal{F}_2 = \cdots = \mathcal{F}_n$.

Valuation functions. The valuation function $v_i : \mathcal{F}_i \to \mathbb{Z}_{\leq 0}$ specifies the value (or utility) derived by agent a_i for every feasible set of chores. Notice that the valuations are non-positive integers. A valuation function v_i is *monotone* if for any two feasible subsets of chores $C, C' \in \mathcal{F}_i$ such that $C \subseteq C'$, we have $v_i(C) \geq v_i(C')$. We say that the valuations are *additive* if for any feasible set of chores $C \in \mathcal{F}_i$, $v_i(C) \coloneqq \sum_{c_j \in C} v_i(\{c_j\})$. For simplicity, we will write $v_{i,j}$ to denote $v_i(\{c_j\})$. The valuation profile $\mathcal{V} \coloneqq \{v_1, v_2, \ldots, v_n\}$ specifies the valuation functions of all agents.

Special classes of valuations. We say that agents have *identical* valuations if the valuation functions $v_1(\cdot), v_2(\cdot), \ldots, v_n(\cdot)$ are the same, i.e., for any chore c_j, we have $v_{i,j} = v_{k,j}$ for any pair of agents a_i and a_k. The valuations are said to be *dichotomous* if each agent has one of two distinct values for any chore. Formally, there exist two distinct non-positive integers H and L such that for every agent a_i and every chore c_j, we have $v_{i,j} \in \{H, L\}$.

Schedule. A *schedule* (or *allocation*) $\mathbf{X} \coloneqq (X_1, X_2, \ldots, X_n)$ is an ordered n-partition of a subset of chores in $\mathcal{C}$ where X_i denotes the set of chores (or *bundle*) assigned to agent a_i; thus, for every pair of agents $a_i, a_k \in \mathcal{A}$, we have $X_i \cap X_k = \emptyset$ and $X_1 \cup \cdots \cup X_n \subseteq \mathcal{C}$. A schedule $\mathbf{X}$ is said to be *feasible* if for every agent a_i, we have $X_i \in \mathcal{F}_i$. We will write $\mathcal{F}$ to denote the space of all feasible schedules; notice that $\mathcal{F} \subseteq \mathcal{F}_1 \times \mathcal{F}_2 \times \cdots \times \mathcal{F}_n$.

Complete and maximal schedule. A schedule $\mathbf{X} \coloneqq (X_1, X_2, \ldots, X_n)$ is called *complete* if no chore is left unassigned, i.e., $X_1 \cup \cdots \cup X_n = \mathcal{C}$. Note that a complete schedule may not be feasible in general.[6] A *maximal* schedule is one that is feasible and has the additional property that assigning any unallocated chore to any agent makes it infeasible. Unless explicitly stated otherwise, we will use the term 'schedule' to refer to a feasible (but possibly incomplete and possibly non-maximal) schedule.

[5] Not to be confused with Constraint Satisfaction Problem.

[6] For example, for two agents and three chores that have identical start times and finish times, no complete schedule is feasible.

Figure 2: *(a) A scheduling instance and (b) its conflict graph.*

Conflict graph. Given any CSP instance $\langle \mathcal{A}, \mathcal{C}, \mathcal{T}, \mathcal{V} \rangle$, we will find it convenient to define its *conflict graph* $G = (\mathcal{C}, E)$ as follows [Hummel and Hetland, 2022, Chiarelli et al., 2023]: The set of vertices is the set of chores, and for any pair of vertices $c_i, c_j \in \mathcal{C}$, there is an undirected edge $\{c_i, c_j\}$ if and only if c_i and c_j overlap; see Figure 2 for an illustration. Note that a maximal schedule corresponds to a subpartition into independent sets in the conflict graph. Also, observe that the conflict graphs correspond to the class of *interval* graphs [West, 2001, Sec 5.1.15].

Envy-freeness. A schedule $\mathbf{X} := (X_1, X_2, \ldots, X_n)$ is said to be (a) *envy-free* (EF) [Gamow and Stern, 1958, Foley, 1967] if for every pair of agents $a_i, a_k \in \mathcal{A}$, we have $v_i(X_i) \geq v_i(X_k)$; (b) *envy-free up to any chore* (EFX) [Caragiannis et al., 2019, Aziz et al., 2019] if for every pair of agents $a_i, a_k \in \mathcal{A}$ such that $X_i \neq \emptyset$ and for every chore $c \in X_i$, we have $v_i(X_i \setminus \{c\}) \geq v_i(X_k)$, and (c) *envy-free up to one chore* (EF1) [Budish, 2011, Aziz et al., 2019] if for every pair of agents $a_i, a_k \in \mathcal{A}$ such that $X_i \neq \emptyset$, we have $v_i(X_i \setminus \{c\}) \geq v_i(X_k)$ for some chore $c \in X_i$. It is easy to verify that EF $\Rightarrow$ EFX $\Rightarrow$ EF1 and that all implications are strict.

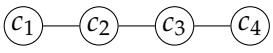
Pareto optimality. A schedule $\mathbf{X} := (X_1, X_2, \ldots, X_n)$ is said to Pareto dominate another schedule $\mathbf{Y} := \{Y_1, Y_2, \ldots, Y_n\}$ if $v_i(X_i) \geq v_i(Y_i)$ for every agent a_i and $v_k(X_k) > v_k(Y_k)$ for some agent a_k. A *Pareto optimal* schedule is one that is maximal and is not Pareto dominated by any other maximal schedule. The maximality requirement is helpful in ruling out trivial solutions; in particular, an empty schedule that leaves all chores unassigned cannot be Pareto optimal according to this definition.

3 Non-Existence Results

In this section, we will show that various combinations of fairness and efficiency notions can fail to exist in the chore scheduling problem. Interestingly, all of our counterexamples involve two agents with identical valuations and the conflict graph is a path graph.

Let us start with a negative result for EFX and maximality.

Example 1 (EFX and maximal schedule may not exist)**.** *Consider an instance with two agents a_1 and a_2 and four chores c_1, c_2, c_3, and c_4 that are identically valued by the agents at -1, -1, -1, and -4, respectively. The conflict graph is shown below:*

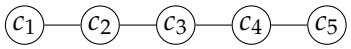


Let $\mathbf{X}$ be the desired EFX *and maximal schedule. Observe that due to maximality, the chore c_4 cannot remain unallocated under $\mathbf{X}$. This is because if the neighboring chore c_3 is assigned to one of the agents, say a_1, then the chore c_4 must be assigned to the other agent a_2. Similarly, if c_3 is unassigned, then c_4 can be assigned to either of the agents without creating any conflict. Thus, we can assume, without loss of generality, that c_4 is assigned to agent a_1.*

In order for the schedule $\mathbf{X}$ *to satisfy* EFX*, agent* a_1 *cannot be assigned any other chore. Furthermore, feasibility dictates that the other agent* a_2 *can be given at most two of the three remaining chores* c_1*,* c_2*, and* c_3*. If agent* a_2 *gets exactly two chores, then it must be given* c_1 *and* c_3*; however, then* c_2 *must be assigned to agent* a_1*, violating* EFX*. On the other hand, if agent* a_2 *gets at most one chore, then once again by maximality of* $\mathbf{X}$*, agent* a_1 *will be required to get at least one chore out of* c_1*,* c_2*, and* c_3*, again violating* EFX*. Thus, an* EFX *and maximal schedule does not exist in the above instance.* □

The non-existence of EFX motivates the consideration of a weaker approximation such as EF1. Our next example shows that EF1 and Pareto optimality can be mutually incompatible even for two agents with identical valuations on a path graph.

Example 2 (EF1 and Pareto optimal schedule may not exist)**.** *Consider an instance with two agents* a_1*,* a_2 *and five chores* $c_1, \ldots, c_5$ *that are identically valued by the agents at* -2*,* -10*,* -1*,* -10*, and* -2*, respectively. The conflict graph is shown below:*



Any maximal schedule that allocates a "heavy" chore (c_2 *or* c_4*) to one of the agents, say* a_1*, can be shown to be Pareto dominated by a schedule that assigns the extreme chores (*c_1 *and* c_5*) to* a_1 *and the middle chore* c_3 *to* a_2*. Therefore, any maximal schedule that leaves both heavy chores unassigned must be of the form* $(\{c_1, , c_5\}, \{c_3\})$ *or* $(\{c_3\}, \{c_1, , c_5\})$*, neither of which satisfy* EF1*.* □

Note that a Pareto optimal schedule (without EF1) always exists; in particular, a schedule that maximizes the sum of agents' utilities (i.e., the utilitarian social welfare maximizing schedule) over the space of all maximal schedules is Pareto optimal.

Another notion of efficiency is *completeness* which asks that no chore should be left unassigned. A complete schedule may not exist (consider two agents and a triangle conflict graph). However, even when a complete schedule exists, no such schedule may satisfy EF1.

Example 3 (EF1 and complete schedule may not exist)**.** *Consider an instance with two agents and four chores* c_1, c_2, c_3, c_4 *such that the odd index chores are valued at* -1 *each and the even index chores are valued at* -3 *each by both agents. The conflict graph is a path graph, similar to the one in Example 1. Any complete schedule must assign all odd index chores to one agent and all even index chores to the other. It is easy to see that* EF1 *is violated from the perspective of the agent with even index chores.* □

The failure of EF1 and completeness means that we must focus on EF1 and maximality in search of positive results. Note that for three or more agents and a path graph, a schedule is maximal if and only if it is complete.

4 Limitations of Algorithms from the Unconstrained Setting

Part of what makes the chore scheduling problem challenging is that algorithmic techniques from unconstrained fair division do not automatically extend to the constrained problem. We will illustrate this challenge through two examples that demonstrate that the well-known round robin and envy-cycle elimination algorithms, which satisfy EF1 for unconstrained items, fail to do so in the presence of conflicts.

The round robin algorithm iterates over the agents according to a fixed permutation, and each agent, on its turn, picks its favorite remaining item. This algorithm is known to find an EF1 allocation under additive valuations in the unconstrained problem. However, when the conflict

graph is a path graph, round robin can fail to achieve EF1 even for two agents with identical valuations.

Example 4 (Round robin fails EF1)**.** *Consider an instance with eight chores $c_1, \ldots, c_8$ and two agents a_1, a_2 with identical valuations. The valuations profile and the scheduling constraints are shown in Figure 3.*

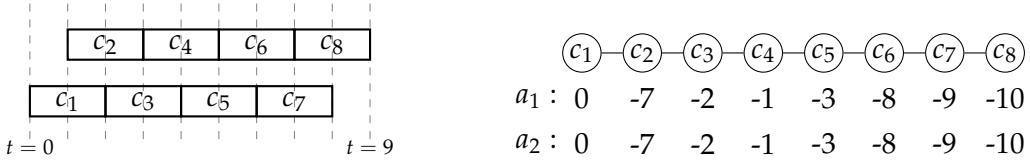


Figure 3: *The instance used in Example 4 (left) and its conflict graph and valuations (right).*

Suppose agent a_1 goes first in the round robin algorithm. Then the induced schedule is $X_1 = \{c_1, c_3, c_5, c_7\}$ and $X_2 = \{c_4, c_2, c_6, c_8\}$. Note that $v_2(X_2 \setminus c) < v_2(X_1)$ for every $c \in X_2$, implying that the schedule fails EF1*.*

The reason why round robin fails EF1 *in the scheduling model is that after picking c_4 in the first round, agent a_2 can no longer pick c_5 in its next turn due to feasibility constraint. Thus, it has to pick its favorite among the feasible chores, which is c_2. This results in a "build up" of envy in each round, which cannot be compensated even after removing the worst chore c_8 in its bundle.* □

The envy-cycle elimination algorithm is known to find an EF1 allocation for indivisible goods in the unconstrained setting. For chores, a modification of this algorithm called the top-trading envy-cycle elimination algorithm [Bhaskar et al., 2021] provides similar guarantees. The latter algorithm works as follows: At each step, the algorithm assigns a chore to a "sink" vertex in the envy graph.[7] If a sink does not exist, there must exist a cycle of "most envied edges", which, when resolved (i.e., by a cyclic swap of bundles), creates a sink vertex.

Example 5 (Envy-cycle elimination fails EF1)**.** *Consider an instance with two agents a_1, a_2 and five chores $c_1, \ldots, c_5$ whose conflict graph is a path graph (similar to Example 1). The chores are valued at -10, -1, -10, -3, and -2, respectively, by both agents. Note that since the valuations are identical, an envy cycle can never occur. Thus, a sink agent always exists.*

At each step of the algorithm, we allow a sink agent to pick its favorite among the feasible chores. Thus, agent a_1 starts by picking the chore c_2, which makes a_2 the sink. Next, agent a_2 picks the chore c_5, which again makes a_1 the sink, and so on. It can be checked that the induced schedule is $(\{c_2, c_4\}, \{c_1, c_3, c_5\})$. However, EF1 *is violated from a_2's perspective as it gets both heavy chores.* □

Another challenge with the chore scheduling problem is that the set of maximal schedules seems to lack any "easy to exploit" structure. In particular, one might wonder whether maximal schedules constitute a *matroid* [Oxley, 2022]. If so, it would be natural to use known algorithms for finding an EF1 allocation under matroid constraints [Biswas and Barman, 2019, Dror et al., 2023]. However, any maximal schedule lacks the downward closedness property; in particular, unassigning an item from a maximal schedule does not maintain maximality. Similarly, the exchange axiom may also fail for a maximal schedule. To see this, consider a star conflict graph

[7] The envy graph associated with a given schedule is a directed graph whose vertices are the agents and there is an edge (i, j) if agent i envies agent j in the given schedule. A sink vertex refers to an agent who does not envy any other agent.

as shown in Figure 2. Suppose the center chore is assigned to agent a_1 and all leaf chores are assigned to agent a_2. Agent a_1's bundle has fewer chores, but transferring any of agent a_2's chores to agent a_1 creates infeasibility. Thus, it is not clear if ideas from matroid theory can be applied to the chore scheduling problem.

5 Results for Two Agents

In this section, we will discuss our algorithmic results for two agents. We will start with the case when the conflict graph is a *path* graph (Theorem 1). As we have seen in Sections 3 and 4, the setting of two agents and a path graph is already quite nontrivial as all of our counterexamples hold even in this special case. Our algorithm will use a novel *coloring* technique and a construct called *adjacent schedules* that will help us circumvent the limitations discussed previously. We will then show that, building on our coloring technique, a more sophisticated algorithm can find an EF1 and maximal schedule for two agents for any *interval* graph (Theorem 3).

Notably, both of our algorithms apply to general *monotone* valuations, which is a significantly broader class that contains additive valuations. In light of the non-existence results in Section 3, the result in Theorem 3 is the most general positive result for two agents that one can expect in our problem.

Let us start with our algorithm for path graphs.

Theorem 1 (Two agents and path graph)**.** *There is a polynomial-time algorithm that, given any* CSP *instance with two agents with monotone valuations and an arbitrary path graph, returns an* EF1 *and maximal schedule.*

Our algorithm will use the idea of *adjacent schedules* which is defined below.

Definition 1 (Adjacent schedules)**.** *Two schedules* $\mathbf{X} = (X_1, X_2)$ *and* $\mathbf{Y} = (Y_1, Y_2)$ *are said to be adjacent if for any* $i \in \{1,2\}$*,* Y_i *is obtained from* X_i *by adding at most one element and removing at most one element, that is,*

$$|Y_i \setminus X_i| \leq 1 \text{ and } |X_i \setminus Y_i| \leq 1 \text{ for } i \in \{1,2\}.$$

For example, an adjacent schedule can be obtained by the two agents exchanging a pair of chores, or by assigning an unallocated chore to agent 1 and transferring another chore from agent 1 to agent 2, etc. The following lemma shows that if we have two adjacent schedules and an agent is envious in one but not in the other, then one of the two schedules or the schedules obtained after swapping the bundles satisfies EF1.

Lemma 1. *Let* $\mathbf{X} = (X_1, X_2)$ *and* $\mathbf{Y} = (Y_1, Y_2)$ *be two adjacent schedules such that*

- *in schedule* $\mathbf{X}$*, agent* a_1 *envies agent* a_2*, and*
- *in schedule* $\mathbf{Y}$*, agent* a_1 *does not envy agent* a_2*.*

Then, at least one of the following four schedules must be EF1*:* $\mathbf{X}$*,* $\mathbf{Y}$*, or the schedules obtained by swapping the agents' bundles, i.e.,* $\mathbf{X}' := (X_2, X_1)$*, or* $\mathbf{Y}' := (Y_2, Y_1)$*.*

Note that while the conditions of the lemma only require the envy to be considered from agent a_1's perspective, the EF1 implication holds for both agents. Also, as will become clear, the proof of Lemma 1 works for monotone valuations. Furthermore, the lemma can also be shown to hold for *goods* (i.e., when all valuations are non-negative) under monotone valuations.

Proof. (of Lemma 1) For the sake of contradiction, assume that none of the four schedules is EF1. We can assume that a_2 does not envy a_1 in schedule $\mathbf{X}$, because otherwise $\mathbf{X}'$ would be EF. Similarly, we can assume that a_2 does not envy a_1 in schedule $\mathbf{Y}'$, because otherwise a_2 would not envy a_1 in $\mathbf{Y}$ and $\mathbf{Y}$ would be EF. Now, since we assumed $\mathbf{X}$ and $\mathbf{Y}'$ are not EF1, in both schedules agent a_1 must envy agent a_2 even after giving up any single chore. By adjacent property of $\mathbf{X}$ and $\mathbf{Y}$, we know that $|X_1 \setminus Y_1| \leq 1$ and $|Y_2 \setminus X_2| \leq 1$. Hence, we can write

$$\text{For } \mathbf{X} : v_1(X_1 \setminus (X_1 \setminus Y_1)) < v_1(X_2), \text{ and}$$
$$\text{For } \mathbf{Y}' : v_1(Y_2 \setminus (Y_2 \setminus X_2)) < v_1(Y_1).$$

Note that $X_1 \setminus \{X_1 \setminus Y_1\} \subseteq Y_1$ and $Y_2 \setminus \{Y_2 \setminus X_2\} \subseteq X_2$. Putting this together with the above two equations, we get

$$v_1(Y_1) \leq v_1(X_1 \setminus \{X_1 \setminus Y_1\}) < v_1(X_2) \text{ and}$$
$$v_1(X_2) \leq v_1(Y_2 \setminus \{Y_2 \setminus X_2\}) < v_1(Y_1).$$

The two inequalities give us a contradiction. □

Lemma 1 implies that in order to find an EF1 and maximal schedule, it suffices to find a pair of adjacent and maximal schedules such that agent a_1 is envious under one but not the other. Towards this goal, we will construct a *sequence* of maximal schedules, starting with a schedule $\mathbf{X} = (X_1, X_2)$ and ending with the swapped bundles $\mathbf{X}' = (X_2, X_1)$, such that any two consecutive schedules in the sequence are adjacent. Note that agent a_1 cannot envy agent a_2 in both X and X'. Hence, there will be two consecutive schedules (i.e., a switchover point) in this sequence that will satisfy the condition required in Lemma 1 and will give us the desired EF1 schedule.

For our next result (Lemma 2), we will find it convenient to identify agent a_1 with red and agent a_2 with blue color.

Lemma 2. *For any* CSP *instance with two agents and a path graph, there exists a sequence of feasible schedules* $(\mathbf{X}_1 = (R_1, B_1), \mathbf{X}_2 = (R_2, B_2), \ldots, \mathbf{X}_m = (R_m, B_m))$*, where* $m \in \mathbb{N}$ *is the number of chores, such that*

1. *The last schedule is obtained by swapping the two bundles in the first schedule. That is,* $R_1 = B_m$ *and* $B_1 = R_m$*.*
2. *For any* $2 \leq i \leq m$*, the two schedules* $\mathbf{X}_i$ *and* $\mathbf{X}_{i-1}$ *are adjacent.*
3. *For any* $1 \leq i \leq m$*, the schedule* $\mathbf{X}_i$ *is maximal.*
4. *The sequence* $\mathbf{X}_1, \ldots, \mathbf{X}_m$ *can be constructed in polynomial time.*

Note that Theorem 1 follows readily from Lemmas 1 and 2. Indeed, the sequence of schedules returned by the algorithm in Lemma 2 consists only of maximal schedules. The extreme schedules are swapped versions of each other, therefore there must exist a pair of consecutive schedules, say $\mathbf{X}_i$ and $\mathbf{X}_{i+1}$, in the sequence where the envy of agent a_1 switches. From Lemma 1, at least one of $\mathbf{X}_i$, $\mathbf{X}_{i+1}$, or the swapped versions of these must satisfy EF1.

Proof. (of Lemma 2) We will prove the lemma by means of a coloring technique. Let the chores, in the increasing order of their finish times, be denoted by $c_1, c_2, \ldots, c_m$. We define the m schedules $\mathbf{X}_i = (R_i, B_i)$ for $1 \leq i \leq m$, as follows (see Figure 4):

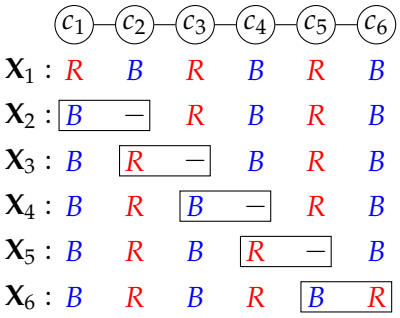


Figure 4: *Illustrating the coloring technique on a path graph.*

- $R_1 = \{c_h : h \text{ is odd}\}$ and $B_1 = \{c_h : h \text{ is even}\}$.
- For any $i \in \{2, 3, \ldots, m-1\}$,
  - R_i contains $\{c_h : 1 \leq h < i, h \text{ is even}\} \cup \{c_h : i < h \leq k, h \text{ is odd}\}$,
  - B_i contains $\{c_h : 1 \leq h < i, h \text{ is odd}\} \cup \{c_h : i < h \leq k, h \text{ is even}\}$,
  - and the chore c_i is unassigned c_{i+1}.
- $R_k = \{c_h : h \text{ is even}\}$ and $B_k = \{c_h : h \text{ is odd}\}$.

Feasibility: It is easy to see that each schedule $\mathbf{X}_i$ in the above construction is feasible. Indeed, any colored chore c_h intersects with at most one colored chore with earlier finish time, which can only be c_{h-1}. But the two chores c_h and c_{h-1} are not assigned to the same agent in any of the schedules.

Proof of part (1): From the construction, $R_1 = B_m$ and $B_1 = R_m$ immediately follows.

Proof of part (2): For any even $2 \leq i \geq m$, observe that $B_i \setminus B_{i-1} = c_{i-1}$, and $B_{i-1} \setminus B_i = c_i$. Moreover, $R_{i-1} \setminus R_i = \emptyset$ and $R_i \setminus R_{i-1}$ is empty for $i < m$ and is equal to $\{c_m\}$ for $i = m$. Hence, $\mathbf{X}_{i-1}$ and $\mathbf{X}_i$ are adjacent. For odd i, a similar argument works.

Proof of part (3): For any schedule $\mathbf{X}_i$, there is at most one unassigned chore. Any such chore is sandwiched between two chores that are assigned to different agents, implying maximality.

Proof of part (4): The number of schedules is equal to the number of chores m. Furthermore, at each step, assigning or unassigning the chores takes constant time. Hence, the coloring algorithm runs in polynomial time, as desired. □

Having shown that an EF1 and maximal schedule exists for two agents on a path graph, let us now turn to the more general class of interval graphs. Here, we will first show that a generalization of the coloring technique for path graphs gives a weaker fairness guarantee of EF2 (instead of EF1) along with maximality for interval graphs.[8] Subsequently, we will provide a more sophisticated algorithm that gives a stronger guarantee of EF1.

Theorem 2 (EF2 for two agents and interval graph)**.** *There is a polynomial-time algorithm that, given any* CSP *instance with two agents with monotone valuations and an arbitrary interval graph, returns an* EF2 *and maximal schedule.*

[8] A schedule $\mathbf{X} = (X_1, X_2)$ satisfies envy-freeness up to two chores (EF2) if for every pair of agents a_i, a_k, there exist two chores $c, c' \in X_i$ such that $v_i(X_i \setminus \{c, c'\}) \geq v_i(X_k)$.

Towards proving Theorem 2, we will once again construct a sequence of schedules, only this time with a slight relaxation of maximality, as described in the next lemma.

Lemma 3. *For any* CSP *instance with two agents and an interval graph, there exists a sequence of feasible schedules* $(\mathbf{X}_1 = (R_1, B_1), \mathbf{X}_2 = (R_2, B_2), \ldots, \mathbf{X}_k = (R_k, B_k))$ *for some* $k \in \mathbb{N}$ *such that*

1. *The last schedule is obtained by swapping the two bundles in the first schedule. That is,* $R_1 = B_k$ *and* $B_1 = R_k$.
2. *For any* $2 \leq i \leq k$, *the two schedules* $\mathbf{X}_i$ *and* $\mathbf{X}_{i-1}$ *are adjacent.*
3. *For any* $1 \leq i \leq k$, *the schedule* $\mathbf{X}_i$ *is either maximal or can be made maximal by including one unassigned chore in either of its two bundles* R_i *or* B_i.
4. *The sequence* $\mathbf{X}_1, \ldots, \mathbf{X}_k$ *can be constructed in polynomial time.*

Proof. (of Lemma 3) We will classify each chore as colored or uncolored as follows: Consider the chores in the increasing order of their finish times (In the event of a tie, we may resolve it by subtracting a small value, such as i/n^2, from the finish time of the ith chore. This adjustment ensures an increasing order of finish times for the chores). A chore c is classified as *uncolored* if and only if its time interval overlaps with two or more chores which have earlier finish time than c and which are classified as colored.

Now, we will construct the desired sequence of schedules. The uncolored chores will not be assigned to any agent in any of the schedules. Let the colored chores be $c_{j_1}, c_{j_2}, \ldots, c_{j_k}$ in increasing order of their finish times. We define the k schedules $\mathbf{X}_i = (R_i, B_i)$ for $1 \leq i \leq k$, as follows.

- $R_1 = \{c_{j_h} : h \text{ is odd}\}$ and $B_1 = \{c_{j_h} : h \text{ is even}\}$.
- For any $i \in \{2, 3, \ldots, k-1\}$,
  - R_i contains $\{c_{j_h} : 1 \leq h < i, h \text{ is even}\} \cup \{c_{j_h} : i < h \leq k, h \text{ is odd}\}$,
  - B_i contains $\{c_{j_h} : 1 \leq h < i, h \text{ is odd}\} \cup \{c_{j_h} : i < h \leq k, h \text{ is even}\}$,
  - and the chore c_{j_i} is assigned or unassigned in the schedule $\mathbf{X}_i$ depending on whether the interval of c_{j_i} intersects with the intervals of $c_{j_{i-1}}$ and $c_{j_{i+1}}$. Note that one of chores $c_{j_{i-1}}$ and $c_{j_{i+1}}$ is in R_i and the other one is in B_i. If the interval of c_{j_i} intersects with both, then c_{j_i} is unassigned. If it intersects with only $c_{j_{i+1}}$, then it is assigned to the same agent as $c_{j_{i-1}}$. And otherwise, it is assigned to the same agent as $c_{j_{i+1}}$.
- $R_k = \{c_{j_h} : h \text{ is even}\}$ and $B_k = \{c_{j_h} : h \text{ is odd}\}$.

Feasibility: By the definition of colored chores, any colored chore c_{j_h} can intersect with at most one colored chore with an earlier finish time, which can be only $c_{j_{h-1}}$. But, the two chores c_{j_h} and $c_{j_{h-1}}$ are not assigned to the same agent in any of the schedules, if their intervals intersect. Moreover, the uncolored chores are unassigned in all the schedules. Hence, each schedule $\mathbf{X}_i$ is feasible.

Proof of part (1): From the construction, $R_1 = B_k$ and $B_1 = R_k$ immediately follows.

Proof of part (2): For any $i \geq 2$, observe that the only chores that can possibly be in $R_i \setminus R_{i-1}$ are $c_{j_{i-1}}$ and c_{j_i}. Moreover, if $c_{j_{i-1}}$ is in $R_i \setminus R_{i-1}$, then $i-1$ must be even. In that case, c_{j_i} must be in R_{i-1}, and hence cannot be in $R_i \setminus R_{i-1}$. Thus, we can conclude that $|R_i \setminus R_{i-1}| \leq 1$.

Similarly, the only chores that can possibly be in $R_{i-1} \setminus R_i$ are $c_{j_{i-1}}$ and c_{j_i}. Moreover, if c_{j_i} is in R_{i-1}, then i must be odd. In that case, $c_{j_{i-1}}$ must be in R_i, and hence cannot be in $R_{i-1} \setminus R_i$. Thus, we can conclude that $|R_{i-1} \setminus R_i| \leq 1$.

Similarly, one can argue that $|B_i \setminus B_{i-1}| \leq 1$ and $|B_{i-1} \setminus B_i| \leq 1$. Thus, the two schedules $\mathbf{X}_{i-1}$ and $\mathbf{X}_i$ are adjacent for any $i \geq 2$.

Proof of part (3): Consider the schedule $\mathbf{X}_i$ for some i. We will argue that there is at most one unassigned chore that can be assigned to some agent without violating feasibility. First, consider the chore c_{j_i}. It is unassigned only if it intersects with two chores, one in R_i and the other in B_i. Hence, it cannot be assigned to any agent.

The other unassigned chores are the uncolored chores. Let us partition the uncolored chores into k buckets as follows: For $1 \leq h \leq k$, let U_h be the set of uncolored chores whose finish time are between those of c_{j_h} and $c_{j_{h+1}}$. Note that U_1 is empty. By the design of the coloring scheme, any chore in set U_h intersects with $c_{j_{h-1}}$ and c_{j_h}, and possibly some other colored chores with earlier finish time. For any h different from i and $i+1$, observe that chores $c_{j_{h-1}}$ and c_{j_h} have been assigned to different agents in the schedule $\mathbf{X}_i$. Hence, any chore in U_h cannot be assigned to any agent for $h \neq i$ and $h \neq i+1$.

It remains to consider the uncolored chores in the sets U_i and U_{i+1}. We first claim that any chore in U_i cannot be assigned to any agent in the schedule $\mathbf{X}_i$. There are two cases:

(i) when c_{j_i} and $c_{j_{i+1}}$ do not intersect, then they both are assigned to the same agent, while $c_{j_{i-1}}$ is assigned to the other agent. Thus, we get that any chore in U_i intersects with two chores which are with different agents, and hence cannot be assigned to any agent.

(ii) when c_{j_i} and $c_{j_{i+1}}$ intersect, then $c_{j_{i+1}}$ must also intersect with every chore in U_i, because chores in U_i have later finish time than c_{j_i}. In this case, any chore in U_i intersects with $c_{j_{i-1}}$ and $c_{j_{i+1}}$ which are with different agents. Hence, it cannot be assigned to any agent.

Now, we consider the chores in U_{i+1}. Any chore in U_{i+1} intersecting with $c_{j_{i-1}}$ cannot be assigned to any agent, because $c_{j_{i-1}}$ and $c_{j_{i+1}}$ are with different agents. In case c_{j_i} and $c_{j_{i+1}}$ are assigned to different agents, we can say that no chore in U_{i+1} can be assigned to any agent.

Consider the case when c_{j_i} and $c_{j_{i+1}}$ are with the same agent or if c_{j_i} is unassigned. Let c^* be the chore in U_{i+1} with the earliest finish time that does not intersect with $c_{j_{i-1}}$. If c^* intersects with $c_{j_{i+2}}$, then so do the all the following chores in U_{i+1}. In that case, the schedule is already maximal. Otherwise, we can assign c^* to the same agent as $c_{j_{i+2}}$. After this, the remaining chores in U_{i+1} cannot be assigned to any agent because they all intersect with $c_{j_{i+1}}$ and c^*. Hence, we will have a maximal schedule after assigning c^*. □

From Lemma 3, we obtain a feasible schedule which is EF1 and can be made maximal by assigning one of its unassigned chores to some agent. The resulting schedule, therefore, satisfies EF2, completing the proof of Theorem 2.

Finally, we note that by a more sophisticated coloring argument, it can be shown that an EF1 and maximal schedule always exists for interval graphs.

Theorem 3 (EF1 for two agents and interval graph)**.** *There is a polynomial-time algorithm that, given any* CSP *instance with two agents with monotone valuations and an arbitrary interval graph, returns an* EF1 *and maximal schedule.*

To prove Theorem 3, it will suffice to show the following result.

Lemma 4. *For any* CSP *instance with two agents and an interval graph, there exists a sequence of feasible schedules* $(\mathbf{X}_0 = (R_0, B_0), \mathbf{X}_1 = (R_1, B_1), \ldots, \mathbf{X}_k = (R_k, B_k))$ *for some* $k \in \mathbb{N}$ *such that*

1. *The last schedule is obtained by swapping the two bundles in the first schedule. That is,* $R_0 = B_k$ *and* $B_0 = R_k$.
2. *For any* $1 \leq i \leq k$, *the two schedules* $\mathbf{X}_i$ *and* $\mathbf{X}_{i-1}$ *are adjacent.*
3. *For any* $0 \leq i \leq k$, *the schedule* $\mathbf{X}_i$ *is maximal.*
4. *The sequence* $\mathbf{X}_0, \ldots, \mathbf{X}_k$ *can be constructed in polynomial time.*

Theorem 3 follows readily from Lemmas 1 and 4. Thus, in the rest of this section, we will focus on the proof of Lemma 4.

Proof. (of Lemma 4) To create the intended sequence of schedules, we employ a three-phase procedure. In the first phase, we construct a maximal schedule, just as in the proof of Lemma 3. The idea is then to gradually move chores from one bundle to another and reach a schedule with swapped bundles. However, as seen in the proof of Lemma 3, the straightforward way of moving chores violates maximality. Instead, we first do a preparatory phase 2, at the end of which, we guarantee enough number of assigned chores overlapping with each unassigned chore. This ensures that we can then move chores between bundles in a natural way and maintain maximality, which is our phase 3.

Phase 1: Begin by arranging the chores in the increasing order of their finish times (In the event of a tie, we may resolve it by subtracting a small value, such as i/n^2, from the finish time of the ith chore. This adjustment ensures an increasing order of finish times for the chores). A chore c is classified as *unmarked* if and only if its time interval overlaps with two or more chores that have earlier finish time than c and which are classified as *marked*. Let the marked chores be denoted as $c_1, c_2, \ldots, c_m$, in the order of finish time. Let A represent a set of all c_is. There can be multiple unmarked chores situated between two marked chores. Let U_i denote the set of unmarked chores between c_i and c_{i+1} for any $1 \leq i \leq m-1$, and $U = \bigcup U_i$.

We define the first schedule, $\mathbf{X}_0 = (R_0, B_0)$, as follows.

- $R_0 = \{c_h : h \text{ is odd}\}$ and $B_0 = \{c_h : h \text{ is even}\}$.

  Consequently, the last schedule, $\mathbf{X}_k$ must be swapping of the bundles, resulting in:

- $R_k = \{c_h : h \text{ is even}\}$ and $B_k = \{c_h : h \text{ is odd}\}$.

  We refer to the first schedule $\mathbf{X}_0$ as the source schedule and the last Schedule $\mathbf{X}_k$ as the target schedule. The bundles to which any chore c_h belongs in the schedule $\mathbf{X}_0$ and in the $\mathbf{X}_k$ schedule are called the source bundle and the target bundle of c_h, respectively.

In Figure 5, it is clear that in the schedule $\mathbf{X}_0$, all the marked chores are assigned to the alternating bundles represented as R and B. Additionally, unassigned chores are indicated as N. In the target schedule, denoted as $\mathbf{X}_k$, all the marked chores will be assigned to the opposite

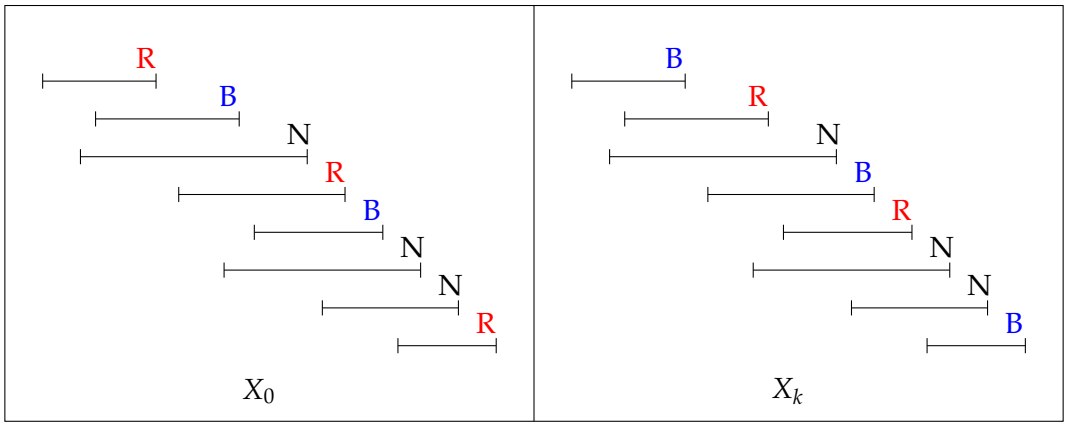


Figure 5: *Pictorial representation of first and the last schedule of the sequence*

bundle, while unmarked chores will retain their unassigned status.

Feasibility and maximality of $\mathbf{X}_0$ *:* According to the definition of a marked chore, any chore c_h can overlap with at most two other marked chores, namely, c_{h-1} and c_{h+1}. If h is even, then $h-1$ and $h+1$ must be odd, and vice versa. Therefore, in the $\mathbf{X}_0$ schedule, it follows that c_{h-1} and c_{h+1} are not in the same bundle as c_h, affirming the feasibility of schedule $\mathbf{X}_0$. Additionally, every unmarked chore must necessarily overlap with chores c_h and c_{h-1} for some h, which belong to different bundles. Consequently, $\mathbf{X}_0$ is also maximal. Similarly, $\mathbf{X}_k$ will also be both feasible and maximal because it is just a complement of $\mathbf{X}_0$.

Phase 2: In this phase, we will construct the first part of the desired sequence of maximal schedules, starting from $\mathbf{X}_0$. This will be done by modifying the status of certain chores, transitioning them from assigned to unassigned, and vice versa.

Definition 1 (Supported chores)**.** *For any given schedule* (R, B)*, we say that an unassigned chore* c *is* supported*, if it satisfies at least one of the following three conditions.*

1. *c overlaps with at least three assigned chores with finish times earlier than c.*
2. *if $c \in U_i$ and c_i is assigned, say, in bundle R (or B), then c also overlaps with an assigned chore from bundle B (or R) with a finish time later than c.*
3. *c overlaps with two assigned chores whose finish times are later than c.*

Now, we will start identifying the unsupported chores and make them supported one by one from right to left. In the process, we will construct a sequence of maximal schedules $\mathbf{X}_i = (R_i, B_i)$ for $1 \leq i \leq m-1$. For any $m \geq i \geq 2$, we construct $\mathbf{X}_{m-i+1}$ from $\mathbf{X}_{m-i}$ as follows. If in schedule $\mathbf{X}_{m-i}$, there is no unsupported chore from U_i, then define $\mathbf{X}_{m-i+1} = \mathbf{X}_{m-i}$. Otherwise, let $u_i^* \in U_i$ be a chore that is unsupported in schedule $\mathbf{X}_{m-i}$ and has the latest finish time.

- Case (i): If c_{i-1} and c_i themselves overlap, then perform color reassignment as follows: $u_i^* \to color(c_{i-1})$ and $c_{i-1} \to$ `unassigned` (see Figure 6-i). That is, if c_i is in B_{m-i} (the other case is analogous) then define

$$B_{m-i+1} = B_{m-i} \text{ and } R_{m-i+1} = R_{m-i} + u_i^* - c_{i-1}.$$

- Case (ii): If c_{i-1} and c_i do not overlap, and c_{i-1} and c_{i-2} also do not overlap, then reassign colors as follows: $u_i^* \to color(c_{i-1})$ and $c_{i-1} \to color(c_i)$ (see Figure 6-ii). That is, if c_i is in B_{m-i} (the other case is analogous) then define

$$B_{m-i+1} = B_{m-i} + c_{i-1} \text{ and } R_{m-i+1} = R_{m-i} + u_i^* - c_{i-1}.$$

- If c_{i-1} and c_i do not overlap, but c_{i-1} and c_{i-2} overlap with each other, consider the following scenarios:

  – Case (iii(a)): If there exist a chore in U_i which is unsupported in $\mathbf{X}_{m-i}$ and which does not overlap with any assigned chore with a later finish time, let $u_i' \in U_i$ be such a chore with the latest finish time. Then reassign colors as follows: $u_i' \to color(c_i)$ and $c_i \to color(c_{i-1})$ (see Figure 6-iii(a)). That is, if c_i is in R_{m-i} (the other case is analogous) then define

$$B_{m-i+1} = B_{m-i} + c_i \text{ and } R_{m-i+1} = R_{m-i} + u_i' - c_i.$$

  – Case (iii(b)): If there is no such unsupported chore in U_i that meets the aforementioned criteria, then carry out color reassignment as follows: $c_i \to color(c_{i-1})$ (see Figure 6-iii(b)). That is, if c_i is in R_{m-i} (the other case is analogous) then define

$$B_{m-i+1} = B_{m-i} + c_i \text{ and } R_{m-i+1} = R_{m-i} - c_i.$$

  Case (iii(c)): It is possible that there are some supported chores in U_i that overlap with chores c_{i-2}, c_{i-1}, c_i, but do not overlap with any assigned chore with a later finish time. Let $u_i' \in U_i$ be such a chore with the latest finish time. Then further reassign colors as follows: $u_i' \to color(c_{i-2})$ and $c_{i-2} \to \texttt{unassigned}$ (see Figure 6-iii(c)). That is, if c_i is now in B_{m-i+1} (the other case is analogous) then define

$$B_{m-i+2} = B_{m-i+1} \text{ and } R_{m-i+2} = R_{m-i+1} + u_i' - c_{i-2}.$$

This finishes the construction of the first part of the desired sequence. From the construction, it is evident that any two consecutive schedules are adjacent. Before moving on to phase 3, we prove certain desired properties of the sequence $(\mathbf{X}_0, \mathbf{X}_1, \ldots, \mathbf{X}_{m-1})$.

Claim 1. *For any i, h, if a chore c_h is unassigned in the schedule $\mathbf{X}_{m-i+1}$, then c_h overlaps with two assigned chores in $\mathbf{X}_{m-i+1}$ whose finish times are later than c_h. Hence, c_h is supported in $\mathbf{X}_{m-i+1}$.*

Claim 2. *For any i, h, if the chore c_h is not in its source bundle in schedule $\mathbf{X}_{m-i+1}$, then for any $j \geq h$, all the chores in U_j are supported in $\mathbf{X}_{m-i+1}$.*

Claim 3. *For any i, the schedule $\mathbf{X}_{m-i+1}$ is feasible and maximal.*

Proof. We will prove the three claims through an induction on i.

Base case: $i = m + 1$. In schedule $\mathbf{X}_0$, each c_h is assigned and is in its source bundle by definition. Hence, Claims 1 and 2 are vacuously true. And we have already argued feasibility and maximality of $\mathbf{X}_0$.

Induction hypothesis: the three statements are true for the schedule $\mathbf{X}_{m-i}$.

Induction step: Recall that if the schedule $\mathbf{X}_{m-i}$ has no unsupported chores from U_i, then $\mathbf{X}_{m-i+1} = X_{m-i}$. And thus, the three statements will be true for $\mathbf{X}_{m-i+1}$ as well.

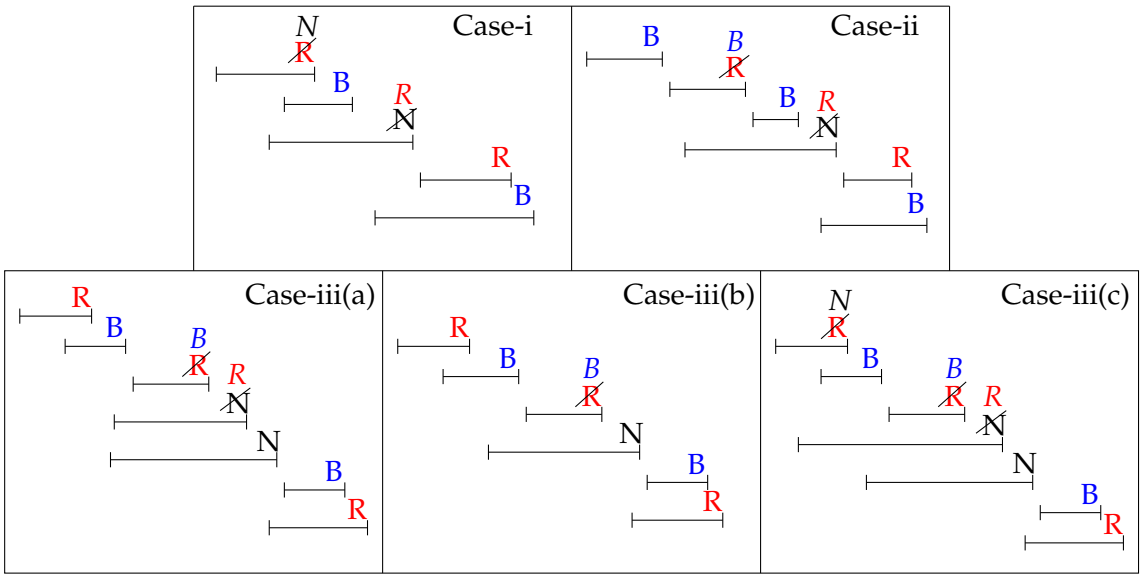


Figure 6: *Visual illustration of all five scenarios along with the reassignment guidelines for unsupported chores. Letters with stock represent the color previously assigned to the chore, while the color mentioned above it reflects the assignment after reassignment.*

Now, consider the case when $\mathbf{X}_{m-i}$ has unsupported chores from U_i. Then from, induction hypothesis for Claim 2, we know that c_i, c_{i-1}, and c_{i-2} are in their source bundles in $\mathbf{X}_{m-i}$. Also all chores in U_i must be unassigned in $\mathbf{X}_{m-i}$ because it is feasible (from Claim 3). Now, we go over various cases considered in the construction of $\mathbf{X}_{m-i+1}$. Recall that $u_i^* \in U_i$ is the chore that is unsupported in schedule $\mathbf{X}_{m-i}$ and has the latest finish time.

- Case (i): c_i and c_{i-1} also overlap each other. In this case, we had the following reassignment: $u_i^* \to color(c_{i-1})$ and $c_{i-1} \to \texttt{unassigned}$.

  For claim 1, observe that c_{i-1} overlaps with c_i and u_i^*, which are both assigned in $\mathbf{X}_{m-i+1}$. The status of any other c_h in $\mathbf{X}_{m-i+1}$ remains same as in $\mathbf{X}_{m-i}$. Hence, the claim follows.

  For claim 2, observe that any chore $u \in U_{i-1}$ overlaps with c_i and u_i^*, and hence satisfies support condition (3). Any chore $u \in U_i$ overlaps with c_i and u_i^* (which are in opposite bundles), and hence satisfies support condition (2). Any $u \in U_j$ for $j > i$ that was supported in X_{m-i} by overlapping with c_{i-1}, c_i, c_{i+1}, is now supported in $\mathbf{X}_{m-i+1}$ by overlapping with three assigned chores c_i, u_i^*, c_{i+1}. Any other $u \in U_j$ has the same status in $\mathbf{X}_{m-i+1}$ and $\mathbf{X}_{m-i}$. Hence, the claim follows.

  For claim 3, observe that since u_i^* is unsupported in $\mathbf{X}_{m-i}$, we know it does not overlap with any assigned chore with a later finish time, which is in the same bundle as c_{i-1}. Hence, the assignment of $color(c_{i-1})$ to u_i^* is feasible. Leaving c_{i-1} unassigned is also maximal, as it overlaps with both bundles: c_i, which is in the opposite bundle of c_{i-1}, and u_i^*, which is in the bundle of c_{i-1}. Any chore $u \in U_{i-1}$ or $u \in U_i$ other than u_i^*, overlap with u_i^* and c_i. Hence, their being unassigned is fine. For any other chores, their status w.r.t. maximality are same in $\mathbf{X}_{m-i}$ and $\mathbf{X}_{m-i+1}$.

- Case (ii): c_i and c_{i-1} do not overlap with each other, and further, there is no overlap between

c_{i-1} and c_{i-2}. We proceeded with the following color reassignment: $u_i^* \to color(c_{i-1})$ and $c_{i-1} \to color(c_i)$. Thus, in schedule $\mathbf{X}_{m-i+1}$, u_i^* is in the opposite bundle of c_i and c_{i-1}.

Claim 1 is true because no new c_h is unassigned.

For Claim 2, observe that any $u \in U_{i-1}$ is supported because it overlaps with c_i and u_i^*, which are in opposite bundles. Any $u \in U_i$ (other than u_i^*) overlaps with c_i and the chore u_i^* from the opposite bundle and with later finish time, and hence is supported. And any $u \in U_j$ for $j > i$ has the same status in $\mathbf{X}_{m-i+1}$ as in $\mathbf{X}_{m-i}$.

For Claim 3, observe that the assignment of c_{i-1} in bundle of c_i is feasible because c_{i-1} does not overlap with c_i and c_{i-2}, and thus, does not overlap with any assigned chores. Since u_i^* is unsupported in $\mathbf{X}_{m-i}$, it does not overlap with any chore with a later finish time which is in the same bundle as c_{i-1}. Hence, putting u_i^* in the earlier bundle of c_{i-1} is fine. As argued in the previous paragraph, any $u \in U_{i-1}$ or $u \in U_i$ (other than u_i^*) overlap with two chores from opposite bundles. And hence, their being unassigned is fine.

- Case (iii(a)): c_{i-1} and c_i do not overlap each other, but c_{i-1} and c_{i-2} overlap with each other. Further there exists an unsupported chore u_i' that does not overlap with any assigned chore with a later finish time. We proceeded with the reassignment $u_i' \to color(c_i)$ and $c_i \to color(c_{i-1})$.

  Claim 1 is true because no new c_h is unassigned.

  For Claim 2, consider any chore $u \in U_i$. If u has a finish time earlier than u_i', then it is supported because it overlaps with c_i and u_i' with a later finish time and from the opposite bundle. If u has a finish time later than u_i', then it is supported because it overlaps with three assigned chores with earlier finish times, c_{i-1}, c_i, u_i'. Consider any chore $u \in U_j$ for $j > i$. If u was supported in $\mathbf{X}_{m-i}$ due to its overlap with c_i, now in $\mathbf{X}_{m-i+1}$, u_i' can take place of c_i.

  For Claim 3, observe that u_i' is unsupported in $\mathbf{X}_{m-i}$, hence only overlaps with two assigned chores c_{i-1} and c_i from opposite bundles. And c_i does not overlap with c_{i-1}. Thus, the reassignment keeps the schedule feasible. Any chore $u \in U_{i-1}$ or $u \in U_i$ (other than u_i') overlaps with c_{i-1} and u_i' from opposite bundles, and hence can remain unassigned.

- Case (iii(b)): c_i, c_{i-1}, and c_{i-2} are positioned like case iii (a). Additionally, there is no unsupported $u \in U_i$ that does not overlap with any assigned chore with a later finish time. We proceeded with the color reassignment: $c_i \to color(c_{i-1})$.

  Claim 1 is true because no new c_h is unassigned.

  For Claim 2, consider any chore $u \in U_i$. If u was supported in $\mathbf{X}_{m-i}$ by overlapping with three assigned chores with earlier finish time, then that will also hold true in $\mathbf{X}_{m-i+1}$. Consider $u \in U_i$ which is unsupported in $\mathbf{X}_{m-i}$ and overlaps with an assigned chore with a later finish time, say c'. Because u is unsupported, c' and c_i must be in the same bundle in X_{m-i} and u must not overlap any other chore from later finish time and in the opposite bundle. We can conclude that any unsupported chore $u \in U_i$ must overlap with c'. In schedule X_{m-i+1}, c' and c_i are in opposite bundles, and hence, u is supported. Similarly, any chore $u \in U_{i+1}$ overlaps with c' and c_i, and hence, is supported.

  For Claim 3, observe that in schedule $\mathbf{X}_{m-i}$, the chore c_i neither overlaps with c_{i-1}, nor with any other chore in the same bundle. Hence, it is feasible to put it in the bundle of c_{i-1}. As

argued in the previous paragraph, any $u \in U_i$ overlaps with two chores in opposite bundles, hence, can be left unassigned.

- Case iii(c): c_i, c_{i-1}, and c_{i-2} are positioned like case iii (a). Additionally, there is no unsupported $u \in U_i$ that does not overlap with any assigned chore with a later finish time. Further there is a chore $u_i' \in U_i$ that overlap with chores c_{i-2}, c_{i-1}, c_i, but does not overlap with any assigned chore with a later finish time. Then we reassigned colors as follows: $u_i' \rightarrow color(c_{i-2})$ and $c_{i-2} \rightarrow \texttt{unassigned}$.

  For Claim 1, observe that c_{i-2} is the newly unassigned chore and it overlaps with c_{i-1} and u_i' from opposite bundles. Hence, c_{i-2} is supported.

  For Claim 2, observe that any chore $u \in U_{i-2}$ overlaps with c_{i-1} and u_i', which have later finish times, and hence is supported. Any chore $u \in U_{i-1}$ overlaps with c_{i-1} and u_i' from opposite bundles, and hence, is supported. Any chore $u \in U_i$ whose finish time is between c_i and u_i' is similarly supported by overlapping with c_i and u_i'. Any chore $u \in U_i$ whose finish time is later than u_i' is supported because it overlaps with assigned chores c_{i-1}, c_i, u_i' with earlier finish times. For any chore $u \in U_j$ for $j < i$, if it was supported in $\mathbf{X}_{m-i}$ by overlap with c_i, then u_i' can take the place of c_i.

  For Claim 3, observe that since c_i has been given $color(c_{i-1})$ and c_{i-2} is unassigned, it is feasible to give $color(c_{i-2})$ to u_i'. Similarly, we can leave c_{i-2} unassigned because it overlaps with u_i' and c_{i-1}, which are in opposite bundles. For any unassigned $u \in U_{i-2}, U_{i-1}, U_i$, we have argued in the previous paragraph that they overlap with two chores in opposite bundles.

□

From the three claims, it is clear that at the end of phase 2, there will be no unsupported unassigned chores.

Following the reassignment, several changes occur: some assigned chores become unassigned, some unassigned chores get assigned, and certain assigned chores achieve their target bundles. We label the chores which are in the same bundle in $\mathbf{X}_m$ as their target bundles and the chores that are unassigned in both $\mathbf{X}_m$ and $\mathbf{X}_k$ as "targeted chores". All other chores are labeled as "untargeted chores". Before proceeding to phase 3, it is essential to establish one more claim, which will assist us in verifying the feasibility of constructed schedules in phase 3.

Claim 4. *Let c_i be an untargeted chore and let c be another chore with a later finish time, which is in the target bundle of c_i in schedule X_m. If c overlaps with c_i, then c must be untargeted and must be the immediately next untargeted chore after c_i (in order of finish times).*

Proof. Assuming there are no unsupported chores in $\mathbf{X}_0$, this implies that no reassignment of chores happens and we have $\mathbf{X}_m = \mathbf{X}_0$. It follows that all unassigned chores are targeted, while all assigned chores are untargeted. In the $\mathbf{X}_0$ schedule, the only assigned chore with a later finish time that overlaps with c_i is c_{i+1}.

However, in cases involving unsupported chores, color reassignments are performed. We will verify the claim for each individual case one by one.

1. The case where c_i and c_{i-1} overlap with each other, we perform the following color reassignments: c_{i-1} is left unassigned, and u_i^* is assigned the color of c_{i-1}. In this scenario, the untargeted assigned chore is c_{i-1}, which overlaps with c_i (of the target color), making it an immediately next untargeted chore. Notably, any unassigned chore between c_{i-1} and c_i retains its status and remains unassigned, effectively becoming targeted. Therefore, the immediately next untargeted chore after c_{i-1} is c_i. Furthermore, u_i^* becomes assigned but with a color that is not the target color of c_{i-1}. Importantly, u_i^* does not overlap with any assigned chore other than c_i and c_{i-1}. Additionally, c_i gains an untargeted chore immediately adjacent to it. Since u_i^* does not overlap with c_{i+1}, it follows that c_i also does not overlap with c_{i+1}. Therefore, the untargeted chore coming immediately after c_i is only u_i^*. Thus, the claim holds for this case.

2. In the case where c_i and c_{i-1} are not overlapping with each other and furthermore, c_{i-1} does not overlap with c_{i-2}, the situation is similar to the first case. None of the unassigned chores between c_i and c_{i-1} change their status; they all remain targeted. Since c_{i-1} is assigned the target color, it is also considered targeted. The only untargeted assigned chore is c_i, which overlaps with u_i^* (of the target color). However, c_i must be an immediately next untargeted chore since every chore between these two becomes supported after this assignment, and, as per the definition of unsupported u_i^*, c_i must not overlap with c_{i+1}. This confirms the claim being true for this case.

3. In the last case where c_i and c_{i-1} are not overlapping each other and c_{i-1} overlaps with c_{i-2}, the first and second situation does not yield any untargeted assigned chores. In the last situation, it is analogous to the first case when c_i and c_{i-1} overlap each other. We can observe the claim in the same way as the first case. This completes the proof that, after phase 2, every untargeted assigned chore overlaps only with the immediately next untargeted chore of the target color from the later finish time.

□

Phase 3: In this phase, we will construct the second part of the desired sequence of maximal schedules, starting from $\mathbf{X}_m$. This will be done by assigning the target bundle to the untargeted chores. We will start identifying the untargeted chores and assign them to their target bundle one by one from left to right. In the process, we will construct a sequence of maximal schedules $\mathbf{X}_i = (R_i, B_i)$ for $m \leq i \leq k$. For any $0 \geq i \geq k - m - 1$, we construct $\mathbf{X}_{m+i+1}$ from $\mathbf{X}_{m+i}$ as follows. If in schedule $\mathbf{X}_{m+i}$, there is no untargeted chore then we are done. Otherwise, let c be a chore that is untargeted in schedule $\mathbf{X}_{m+i}$ and has the earliest finish time. Then assign the target bundle to c and assign a feasible bundle to the untargeted chore immediately next to c. Here feasible bundle means if the chore can get any assignment without violating any feasible condition then assign that bundle otherwise leave it unassigned.

Since in each step, we are assigning at least one chore to its target bundle, it follows that the process will conclude within a linear number of steps.

Feasibility and maximality in Phase 3: Now, we shall establish the feasibility and maximality of Phase 3. After completing phase 2, we have achieved a schedule $\mathbf{X}_m$ that is both feasible and maximal. In Phase 3, we identify the untargeted chore c from the earliest finish time and reassign their color. Further, we assign a feasible color to their immediately next untargeted chore.

ALGORITHM 1: Algorithm for EF1 and maximality for interval graphs

Input: A CSP instance $\langle \mathcal{A}, \mathcal{C}, \mathcal{T}, \mathcal{V} \rangle$ with an interval graph
Output: A sequence of schedules $\mathbf{X}_1, \ldots, \mathbf{X}_k$

▷ `Phase 1: Initial Coloring`

Order the chores in increasing order of their finish time (In the case of a tie, we may resolve it by subtracting a small value, such as i/n^2, from the finish time of the ith chore)
Classify the chores assigned and unassigned
while *Scan from left to right* **do**
    if *A chore overlaps two assigned chores from an earlier finish time* **then**
        Put it in unassigned set U
    else
        Put it in assigned set A
Let $c_1, c_2, \ldots, c_m$, be the chores in set in increasing order of finish times. Now, assign c_h to bundle R if h is odd otherwise assign to bundle B

▷ `Phase 2: Removal of unsupported chores`

while *Scan from right to left* **do**
    if *An unassigned chore is Unsupported* **then**
        Reassign the colors according to one of the five conditions

▷ `Phase 3: Swap the colors of untargeted chores`

while *Scan from left to right* **do**
    Pick the first two untargeted chores
    Assign the target color to the first chore
    Assign a feasible color to the second chore
return *Final sequence of schedules*

If $c \in U_i$, assigning its target color is maximal. Because, in the target schedule($\mathbf{X}_k$), every unassigned chore in U_i overlaps with at least two assigned chores from different bundles from earlier finish times chores. As all earlier finish time chores have been appropriately set to their targeted colors, assigning the target bundle to c becomes automatically maximal.

On the other hand, if $c \in A$, assigning it a target color is feasible, as per the claim 4. This claim established that every untargeted assigned chore only overlaps with its immediately next untargeted chore, which can share the same color as its target color. Since phase 3 involves reassigning colors to the earliest finish time untargeted chores, and the immediate next chore that can have the same color as the target. ensuring the feasibility of assigning the target color to c. Assigning a color to the immediately next untargeted chore poses no issues, as we can assign any feasible color that meets the criteria.

Now, we must demonstrate that all unassigned chores will satisfy the maximality criteria after the color reassignments. As we show after Phase 2, every unassigned chore will be supported. That means all unassigned chores must follow one of the criteria of being supported. Further, we will demonstrate for each case how it will satisfy maximality criteria.

1. In the first scenario, where an unassigned chore c is supported due to the first condition only, i.e. overlaps with three or more assigned chores whose finish time is earlier than the c.

If c also meets any of the other conditions of supported chores then we will demonstrate how c satisfies maximality criteria due to other conditions in the subsequent section. We can consider the simplest case where all three assigned chores are from A, and they are assigned to their source bundles which means they have alternating assigned bundles. In this scenario, it is straightforward to confirm that during each step of phase 3, c will overlap with two assigned chores from different bundles. Now, let's consider another scenario in which all three assigned chores are from A, but their colors are modified, signifying that they do not belong to alternating bundles. This scenario arises due to the color reassignment in case 3(b) only, where we reassign the color to $c_i \in A$ but do not assign a color to $u^* \in U_i$. In this case, we may encounter different color sequences of the three assigned chores, but a careful examination reveals that we will not obtain any supported unassigned chore due to the overlap of three chores with sequences of B, B, B, or R, R, R. This is because, in such a situation, u^* will lose its feasibility. However, we have already demonstrated that we consistently maintain feasibility in phase 2. Therefore, the only situation where an issue arises is when the color sequence of the assigned chores is B, R, R and R, B, B. To avert this scenario, we reassign the color to such supported unassigned chores, which is addressed in reassignment case 3(c).

Moving on, another scenario arises when at least one of the three assigned chores belongs to set U. Let's assume this specific assigned chore is $u' \in U_i$. This reassignment can be attributed to any case except 3(b). Furthermore, we will elucidate how maximality is upheld in each case. If the reassignment is due to case 1, the unassigned chore c will also overlap with assigned c_i and unassigned c_{i-1} with respect to assigned u'. A careful analysis reveals that, during phase 3, two of these three chores must be assigned to the opposite bundles, ensuring the maximality condition for the unassigned chore c. If the reassignment is due to case 2, even in this scenario, c will overlap with assigned c_i and assigned c_{i-1} with respect to assigned u'. Similarly, a close examination indicates that two of these three chores must consistently be assigned to opposite bundles in phase 3, which ensures the maximality condition of c.
If the reassignment is due to case 3(a), then c must overlap with four assigned chores, where three are from the set A and one is from U This occurs because, in case 3(a), we reassigned the colors of $u^* \in U_i$ to the color of c_i Thus, if we replace c_i with u^*, the color sequence of the three originally assigned chores will maintain alternate color sequence. That will help to preserve the maximality condition for c. Finally, in the case of 3(c), this case is very similar to 3(a) the only difference is when assigned U_i is the last chore of the sequence of three assigned chores overlap c. Nevertheless, even for this case, the unassigned chore will maintain maximality condition because it overlaps three chores from A having earlier finish times than c. Therefore, before c is set to unassigned all the three As will get their target color. That will ensure the maximality of unassigned chores in phase 3. This covers all the sub-cases of unassigned chores that overlap with three or more chores.

2. In the second scenario, an unassigned chore $c \in U_i$ overlaps with two assigned chores from the earlier finish time. Additionally, c overlaps c_i from bundle R(or B) and also overlaps with one assigned chore from a later finish time from bundle B(or R) respectively. In phase 3, where we perform color reassignments from earlier finish times chores, there are no concerns regarding the chores from the late finish time. However, the color of c_i may

undergo modification. If c_i already has its target color, then we refrain from altering its color during phase 3. Conversely, if c_i does not have its target color, it implies that the color of c_i matches the target color of c_{i-1}. So before changing the color of c_i, we have already set c_{i-1} to its target color. This ensures that c satisfies the maximality condition. Likewise, before reassigning colors to the chore that overlapped c from later finish time chores, we have already set the target color for all earlier finish time chores. This guarantees the maximality condition for all such supported chores.

3. In the final scenario, where an unassigned chore c overlaps with two assigned chores whose finish times are later than c, it is apparent that both assigned chores must belong to different bundles. This distinction is necessary since they must overlap with each other. In phase 3, our color reassignment efforts are focused on the earlier finish time chores, and thus, we do not alter the status of these particular assigned chores until all the earlier finish time chores are set to their target colors. In this situation, if $c \in U_i$, it must overlap with at least two assigned chores from the earlier finish time chores, in accordance with the target schedule requirements. Conversely, if $c \in A$, it will get assigned its target color and will not require maximality.

This concludes the verification of the schedule's feasibility and maximality across all steps.

Adjacent: After completing phase 1, we obtain a feasible and maximal schedule. In each step of phase 2, we reassign colors based on five different cases. It's worth noting that each of these five cases is adjacent to the previously assigned schedule. Moving on to phase 3, in each step, we reassign colors to only two chores. If these two chores have different colors, then the reassigned schedules will also be adjacent to the previous schedule.

Two adjacent chores cannot have the same color because if they overlap, assigning the same color to both is not the feasible color for the second chore. If they do not overlap, there is no need to reassign a color to the second chore, as the initially given color would be feasible for it. This ensures that each phase of the steps is adjacent to the preceding one, leading to the conclusion that each $\mathbf{X}_i$ is adjacent to $\mathbf{X}_{i+1}$. Therefore, the sequence of schedules is adjacent. □

6 Results for Arbitrary Number of Agents

We will now turn to the case of an arbitrary number of agents. Here, we are unable to settle the existence question for EF1 and maximal schedules even on a path graph. Nevertheless, we show positive results for two restricted settings: For identical dichotomous valuations and a path graph (Theorem 4) and for identical valuations and a graph with bounded connected components (Theorem 5).

We will start with our result for path graphs. Recall that under identical and dichotomous valuations, each chore is either "heavy" (highly disliked) or "light" (less disliked) for all agents. We will assume that heavy chores are valued at H while the light chores are valued at L by each agent; thus, $H < L \leq 0$. We show that for four or more agents with identical dichotomous valuations and a path graph, an EF1 and maximal schedule can be efficiently computed.

Theorem 4 (Path graph and identical dichotomous vals)**.** *For any $n \geq 4$, there is a polynomial-time algorithm that, given any* CSP *instance with n agents with identical, dichotomous, and additive valuations and an arbitrary path graph, returns an* EF1 *and maximal schedule.*

ALGORITHM 2: Algorithm for Identical Dichotomous Valuations

Input: A CSP instance $\langle \mathcal{A}, \mathcal{C}, \mathcal{T}, \mathcal{V} \rangle$ with a path graph
Output: A schedule $\mathbf{X}$

▷ `Phase 1: Grouping agents into meta agents`

1 If n is even: For $i \in [n/2]$, let the meta agent S_i represent the pair of agents a_{2i-1} and a_{2i}.
2 If n is odd: Let the meta agent S_1 represent the triple of agents a_1, a_2, a_3 and for $i \in \{2, 3, \ldots, (n-1)/2\}$, let the meta agent S_i represent the agents a_{2i} and a_{2i+1}.

▷ `Phase 2: Weighted round robin for meta agents`

3 If n is even: First allocate heavy chores from left to right via round robin among the meta agents; specifically, we use the picking sequence $\langle S_1, S_2, \ldots, S_{n/2}, S_1, S_2, \ldots, S_{n/2} \rangle$. Then allocate the light chores from left to right according to the same sequence, starting after the meta agent who last picked a heavy chore.
4 If n is odd: First allocate heavy chores from left to right via weighted round robin among meta agents according to the picking sequence $\langle S_1, S_2, \ldots, S_{(n-1)/2}, S_1, S_2, \ldots, S_{(n-1)/2}, S_1 \rangle$. Then allocate light chores from left to right according to the same sequence, starting after the meta agent who last picked a heavy chore.
5 Assign dummy chores (by adding isolated vertices in the conflict graph) to equalize the number of heavy (similarly, light) chores for all meta agents representing pairs of agents. If n is odd, the meta agent S_1 receives 1.5 times as many chores of each type.

▷ `Phase 3: Assigning chores to individual agents`

6 Solve the 2-agent and 3-agent sub-problems by giving each meta agent's chores to its constituent agents.
7 Remove the dummy chores.
8 **return** *Current schedule*

Our algorithm for establishing Theorem 4 consists of three phases (see Algorithm 2):

Phase 1: Grouping agents into meta agents The algorithm starts by partitioning the n agents into pairs (if n is even) or pairs and one triple (if n is odd). We refer to each such group as a *meta agent*.

Phase 2: Weighted round robin for meta agents In the second phase, the algorithm runs a *weighted round-robin* procedure for the meta agents. If the number of agents n is even, the procedure is the same as the standard round robin algorithm.[9] However, if n is odd, we need to adjust the turn-taking so that the meta agent corresponding to the triple picks 1.5 times as many chores as any other meta agent (see Line 4 of Algorithm 2).[10]

The weighted round-robin step runs in two phases: A *heavy phase*, where, at its turn, each meta agent picks the leftmost available heavy chore. This is followed by a *light phase*, where, starting after the meta agent who was the last to pick in the previous phase, each meta agent picks the leftmost available light chore. Note that it is possible for a meta agent to pick two neighboring chores in the path graph.

[9] For ease of analysis, we execute each "round" as a combination of two copies of the same permutation of the agents, i.e., using the picking sequence $\langle S_1, S_2, \ldots, S_{n/2}, S_1, S_2, \ldots, S_{n/2} \rangle$, where $S_1, S_2, \ldots, S_{n/2}$ are the meta agents. Doing so ensures that each meta agent picks an even number of chores in each round.

[10] We use the picking sequence $\langle S_1, S_2, \ldots, S_{(n-1)/2}, S_1, S_2, \ldots, S_{(n-1)/2}, S_1 \rangle$ among the meta agents, which ensures that the number of chores picked by each meta agent representing a pair of agents is a multiple of 2 while the number of chores picked by meta agent S_1 is a multiple of 3.

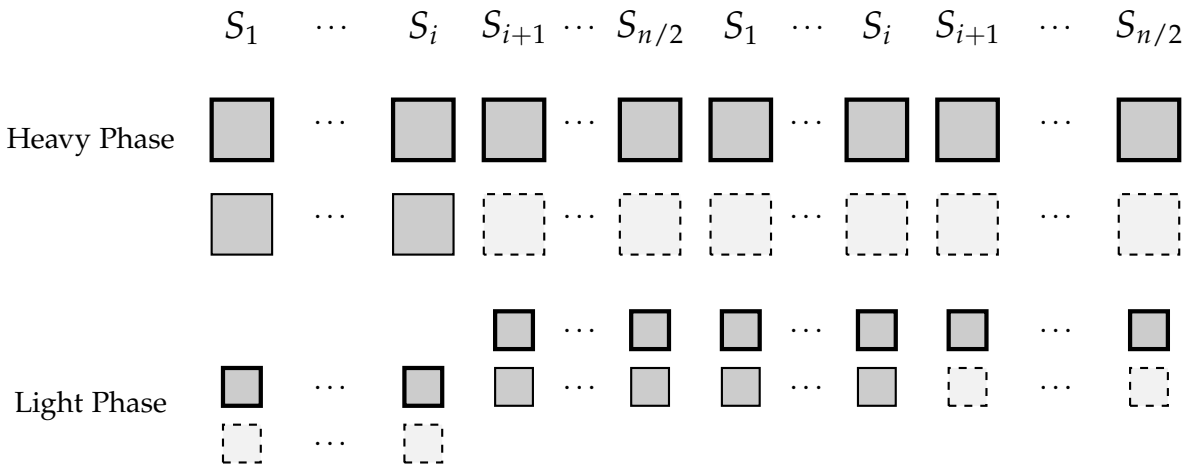


Figure 7: *Illustrating the weighted round-robin algorithm when the number of agents is even. The large and small squares denote the heavy and light chores, respectively. The strongly (respectively, lightly) shaded squares denote the chores that are originally present (respectively, the dummy chores). Thick borders around the squares denote that in that round, all agents receive an original chore, while the squares with regular borders denote the round where some agents picked original chores while others did not because the original chores were consumed.*

By adding an appropriate number of *dummy* chores (heavy and light), the algorithm ensures that each meta agent representing a pair of agents gets the same number of heavy (similarly, light) chores, and the meta agent representing a triple gets 1.5 times as many chores of each type.

Figures 7 and 8 provide a visual illustration of the weighted round-robin step in our algorithm (Algorithm 2).

Phase 3: Assigning chores to individual agents In the third phase, the algorithm distributes the chores of each meta agent among its constituent agents; in other words, the algorithm solves the 2-agent and 3-agent subproblems. At the end of this phase, each individual agent has the same number of heavy (similarly, light) chores. Finally, the algorithm discards the dummy chores and the resulting schedule is returned.

It is easy to see that the algorithm runs in polynomial time. To argue correctness, we will show that before the dummy chores are removed in Phase 3, each agent gets an equal number of heavy (similarly, light) chores; in other words, before the removal of dummy chores, the schedule is envy-free.

Note that since there are at least four agents ($n \geq 4$), there must be at least two meta agents. If n is even, a meta agent never picks two heavy (respectively, light) chores that are adjacent in the path graph (recall that each meta agent picks the *leftmost* available chore). Thus, the set of chores picked by a meta agent must constitute a disjoint union of edges (between one heavy and one light chore) and isolated vertices. Lemma 5 formalizes this observation.

Lemma 5. *For any meta agent S_i that represents a pair of agents, the set of chores picked by S_i is a disjoint union of edges (between one heavy and one light chore) and isolated vertices. Furthermore, these chores can be divided among the two constituent agents such that both agents get the same number of heavy (similarly, light) chores and their bundles are conflict-free.*

Proof. Since there are at least four agents ($n \geq 4$), there must be at least two meta agents. Thus,

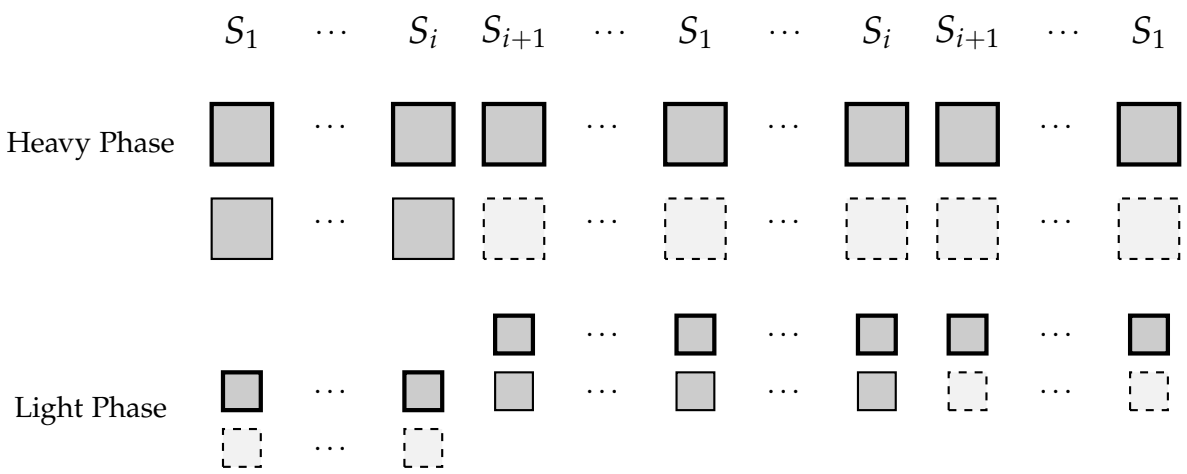


Figure 8: *Illustrating the weighted round-robin algorithm when the number of agents is odd. The drawing convention is the same as described in the caption of Figure 7.*

no meta agent can pick two adjacent heavy (respectively, light) chores in the path graph. Thus, the set of chores picked by any meta agent, say S_i, that represents a pair of agents is a disjoint collection of paths such that each path is of length at most 2. In other words, the chores picked up by the meta agent S_i consist of isolated vertices and disjoint edges, where each edge connects a heavy and a light chore.

By the choice of picking sequences in our algorithm (Lines 3 and 4 in Algorithm 2), any meta agent that represents a pair of agents picks two chores in each round. Thus, the number of heavy (similarly, light) chores in each meta agent's bundle is even. This includes the dummy chores.

Let x denote the number of edges and y denote the number of isolated vertices in the bundle of meta agent S_i. We know that $2x + y$ is even, implying that y is even. Let a_j and a_{j+1} denote the constituent agents of S_i.

If x is even, then there are x heavy chores each connected to a unique light chore. For $x/2$ of the edges, the heavy chore can be given to a_j and the light chore to a_{j+1}. For the other $x/2$ edges, the heavy chore can be given to a_{j+1} and the light chore to a_j. Among the y isolated chores, there should again be an even number of heavy (respectively, light) chores, which can be equitably distributed between the two agents.

If x is odd, then for $\lfloor x/2 \rfloor$ of the edges, the heavy chore can be given to a_j and the light chore to a_{j+1}. For the other $\lceil x/2 \rceil$ edges, the heavy chore can be given to a_{j+1} and the light chore to a_j. Thus, so far, agent a_{j+1} has an extra heavy chore, and agent a_j has an extra light chore. Now, among the y isolated chores, there should be an odd number of heavy and an odd number of light chores. By giving an extra heavy chore to a_j and an extra light chore to a_{j+1} among the isolated chores, and allocating the remaining chores evenly, once again an equitable distribution of heavy and light chores can be achieved. □

When the number of agents is odd, the conclusion of Lemma 5 still holds true for the meta agents representing pairs of agents. However, for the meta agent S_1 which represents a triple of agents, the set of chores constitutes a disjoint union of paths each of length at most 4. Here, a more careful argument is needed to show an equitable distribution of chores among the three constituent agents of the meta agent S_1.

Lemma 6. *For the meta agent S_1 that represents a triple of agents, the set of chores picked by S_1 is a disjoint union of paths of length at most 4. Furthermore, these chores can be divided among the three constituent agents such that all three agents get the same number of heavy (similarly, light) chores and their bundles are conflict-free.*

Proof. We will use an argument similar to that in the proof of Lemma 5. By the choice of the picking sequence in our algorithm (Line 4 in Algorithm 2), the meta agent S_1 can never pick three consecutive heavy (respectively, light) chores in each round. Thus, the set of chores picked by S_1 is a disjoint union of paths each of length at most 4. Furthermore, the number of heavy (similarly, light) chores in S_1's bundle is a multiple of 3. This includes the dummy chores.

Let a_1, a_2, a_3 denote the constituent agents of S_1. We will show that each of these agents can be assigned an equal number of heavy (similarly, light) chores such that no two chores in an agent's bundle are adjacent. Recall that the total number of heavy (similarly, light) chores is a multiple of 3. Thus, it will suffice to argue that, after feasibly allocating the chores in $i-1$ components, the chores in the i^{th} component can also be feasibly assigned such that for any pair of agents, the number of heavy (similarly, light) chores in their bundles differs by at most 1.

To show that the number of heavy (similarly, light) chores is almost balanced in the manner stated above, let us consider the different components that arise in the meta agent's bundle:

- Path of length 3: Note that any path of length 3 contains at most two heavy and at most two light chores. We always assign the three chores in such a path to three different agents.
- Path of length 4: A path of length 4 must contain two heavy and two light chores. For any arrangement of these chores, it is always possible to assign to some agent one heavy and one light chore, and assign to the other two agents the remaining heavy and light chore.
- Path of length 2: In this case, we assign the two chores to two different agents.
- Isolated vertices: The isolated vertices do not pose any constraints on feasibility.

To compute the desired assignment of the chores, we follow a component-wise strategy. At each step, we maintain the following invariant: For any pair of agents, the number of heavy (similarly, light) chores in their bundles differs by at most 1. By case analysis for each component, it can be checked that the chores in that component can be assigned while maintaining this invariant. The lemma follows from the invariant and the multiples-of-3 property. □

Lemmas 5 and 6 show that the two-agent and three-agent subproblems can be solved so that all chores (including the dummy chores) are allocated, i.e., the schedule is complete and hence maximal. Furthermore, each individual agent gets the same number of heavy (similarly, light) chores. Thus, before removing the dummy chores, the schedule is envy-free.

After the removal of dummy chores, the schedule remains EF1. This is because the number of dummy chores assigned to any pair of agents differ by at most 1, and, moreover, the number of heavy (respectively, light) dummy chores assigned to any pair of agents also differ by at most 1. Therefore, removing the dummy chores preserves the EF1 property. Also, note that removing the dummy chores maintains the completeness of the schedule and therefore its maximality. This proves Theorem 4.

Our next result shows that for n agents with identical (but not necessarily dichotomous) and additive valuations, and for any graph each of whose connected components has at most n vertices, an EF1 and maximal schedule can be efficiently computed.

Theorem 5 (Bounded connected components and identical vals)**.** *There is a polynomial-time algorithm that, given any* CSP *instance with n agents with identical additive valuations and an arbitrary interval graph with each connected component of size at most n, returns an* EF1 *and maximal schedule.*

Our algorithm for Theorem 5 proceeds in a component-wise manner. All chores in a connected component are allocated using the round-robin algorithm. The picking order for component i is decided based on the topological ordering of the envy graph of the schedule resulting from the previous $i-1$ components. The envy graph is always acyclic because of identical valuations and thus the topological ordering is well-defined.

We note that our algorithm resembles that of Hummel and Hetland [2022, Proposition 2], who showed a result similar to Theorem 5 for indivisible goods. Notably, their result *does not* require the valuations to be identical. Their proof relies on the fact that for indivisible goods, resolving an *arbitrary* envy cycle preserves the EF1 guarantee [Lipton et al., 2004]. For chores, however, resolving arbitrary envy cycles can interfere with EF1 [Bhaskar et al., 2021], which makes it challenging to extend a similar analysis to our setting.

7 Concluding Remarks

We formulated the problem of fair and efficient scheduling of indivisible chores under conflict constraints. Despite the limited applicability of techniques from the unconstrained setting, we showed, by means of a novel coloring technique, that an EF1 and maximal schedule can be computed for two agents. Resolving the existence of EF1 and maximal schedules for three or more agents for path graphs (and, more generally, for interval graphs) is an exciting open problem. It would also be interesting to consider schedules that "minimize wastage" (i.e., leave the fewest chores unassigned) or satisfy other notions of fairness such as proportionality, equitability, or maximin fair share.

When all chores are equally and identically valued, the EF1 requirement boils down do assigning an almost balanced number of chores to the agents. From the Hajnal-Szemerédi theorem [Hajnal and Szemerédi, 1970], it follows that given any conflict graph with maximum degree Δ, a complete and cardinality-wise balanced schedule always exists if there are $\Delta+1$ agents. But what happens when the number of agents is fixed, and only a maximal schedule is desired? Such a schedule can be shown to exist for interval graphs (by considering the chores in the increasing order of finish times), however, the problem seems nontrivial for general graphs.

Acknowledgments

We are thankful to anonymous reviewers of AAMAS 2024 for their helpful feedback, and to Amit Kumar for valuable discussions during the early stages of this work. SN gratefully acknowledges the support of a MATRICS grant (MTR/2021/000367) and a Core Research Grant (CRG/2022/009169) from SERB, Govt. of India, a TCAAI grant (DO/2021-TCAI002-009), and a TCS grant (MOU/CS/10001981-1/22-23). RV acknowledges support from DST INSPIRE grant no. DST/INSPIRE/04/2020/000107 and SERB grant no. CRG/2022/002621.

References

Miklos Ajtai, James Aspnes, Moni Naor, Yuval Rabani, Leonard J Schulman, and Orli Waarts. Fairness in Scheduling. *Journal of Algorithms*, 29(2):306–357, 1998.

Georgios Amanatidis, Haris Aziz, Georgios Birmpas, Aris Filos-Ratsikas, Bo Li, Hervé Moulin,

Alexandros A Voudouris, and Xiaowei Wu. Fair Division of Indivisible Goods: Recent Progress and Open Questions. *Artificial Intelligence*, page 103965, 2023.

Haris Aziz, Ioannis Caragiannis, Ayumi Igarashi, and Toby Walsh. Fair Allocation of Indivisible Goods and Chores. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 53–59, 2019.

Siddharth Barman, Sanath Kumar Krishnamurthy, and Rohit Vaish. Finding Fair and Efficient Allocations. In *Proceedings of the 19th ACM Conference on Economics and Computation*, pages 557–574, 2018.

Nawal Benabbou, Mithun Chakraborty, Xuan-Vinh Ho, Jakub Sliwinski, and Yair Zick. The Price of Quota-Based Diversity in Assignment Problems. *ACM Transactions on Economics and Computation*, 8(3):1–32, 2020.

Umang Bhaskar, AR Sricharan, and Rohit Vaish. On Approximate Envy-Freeness for Indivisible Chores and Mixed Resources. In *Proceedings of the 24th International Conference on Approximation Algorithms for Combinatorial Optimization Problems*, 2021.

Vittorio Bilò, Angelo Fanelli, Michele Flammini, Gianpiero Monaco, and Luca Moscardelli. The Price of Envy-Freeness in Machine Scheduling. *Theoretical Computer Science*, 613:65–78, 2016.

Vittorio Bilò, Ioannis Caragiannis, Michele Flammini, Ayumi Igarashi, Gianpiero Monaco, Dominik Peters, Cosimo Vinci, and William S Zwicker. Almost Envy-Free Allocations with Connected Bundles. *Games and Economic Behavior*, 131:197–221, 2022.

Arpita Biswas and Siddharth Barman. Matroid Constrained Fair Allocation Problem. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, volume 33, pages 9921–9922, 2019.

Arpita Biswas, Yiduo Ke, Samir Khuller, and Quanquan C Liu. An Algorithmic Approach to Address Course Enrollment Challenges. In *Proceedings of the Fourth Symposium on Foundations of Responsible Computing*, pages 8:1–8:23, 2023.

Sylvain Bouveret, Katarína Cechlárová, Edith Elkind, Ayumi Igarashi, and Dominik Peters. Fair Division of a Graph. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 135–141, 2017.

Sylvain Bouveret, Katarína Cechlárová, and Julien Lesca. Chore Division on a Graph. *Autonomous Agents and Multi-Agent Systems*, 33:540–563, 2019.

Steven J Brams and Alan D Taylor. *Fair Division: From Cake-Cutting to Dispute Resolution*. Cambridge University Press, 1996.

Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel D Procaccia. *Handbook of Computational Social Choice*. Cambridge University Press, 2016.

Eric Budish. The Combinatorial Assignment Problem: Approximate Competitive Equilibrium from Equal Incomes. *Journal of Political Economy*, 119(6):1061–1103, 2011.

Eric Budish, Gérard P Cachon, Judd B Kessler, and Abraham Othman. Course Match: A Large-Scale Implementation of Approximate Competitive Equilibrium from Equal Incomes for Combinatorial Allocation. *Operations Research*, 65(2):314–336, 2017.

Ioannis Caragiannis, David Kurokawa, Hervé Moulin, Ariel D Procaccia, Nisarg Shah, and Junxing Wang. The Unreasonable Fairness of Maximum Nash Welfare. *ACM Transactions on Economics and Computation*, 7(3):12, 2019.

Bhaskar Ray Chaudhury, Jugal Garg, and Kurt Mehlhorn. EFX Exists for Three Agents. In *Proceedings of the 21st ACM Conference on Economics and Computation*, pages 1–19, 2020.

Nina Chiarelli, Matjaž Krnc, Martin Milanič, Ulrich Pferschy, Nevena Pivač, and Joachim Schauer. Fair Allocation of Indivisible Items with Conflict Graphs. *Algorithmica*, 85(5):1459–1489, 2023.

Amitay Dror, Michal Feldman, and Erel Segal-Halevi. On Fair Division under Heterogeneous Matroid Constraints. *Journal of Artificial Intelligence Research*, 76:567–611, 2023.

Soroush Ebadian, Dominik Peters, and Nisarg Shah. How to Fairly Allocate Easy and Difficult Chores. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, pages 372–380, 2022.

Duncan Foley. Resource Allocation and the Public Sector. *Yale Economic Essays*, pages 45–98, 1967.

George Gamow and Marvin Stern. Puzzle-Math. *Viking Press*, 1958.

Martin Gardner. *aha! Insight*. W.H.Freeman and Company, 1978.

Jugal Garg, Aniket Murhekar, and John Qin. Fair and Efficient Allocations of Chores under Bivalued Preferences. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence*, volume 36, pages 5043–5050, 2022.

Jugal Garg, Aniket Murhekar, and John Qin. New Algorithms for the Fair and Effcient Allocation of Indivisible Chores. In *Proceedings of the 32nd International Joint Conference on Artificial Intelligence*, pages 2710–2718, 2023.

Jonathan Goldman and Ariel D Procaccia. Spliddit: Unleashing Fair Division Algorithms. *ACM SIGecom Exchanges*, 13(2):41–46, 2015.

A. Hajnal and E. Szemerédi. Proof of a Conjecture of P. Erdős. In *Combinatorial Theory and Its Applications, I-III (Proc. Colloq., Balatonfüred, 1969)*, volume 4 of *Colloq. Math. Soc. János Bolyai*, pages 601–623. North-Holland, Amsterdam-London, 1970.

Halvard Hummel and Magnus Lie Hetland. Fair Allocation of Conflicting Items. *Autonomous Agents and Multi-Agent Systems*, 36, 2022.

Ayumi Igarashi and Tomohiko Yokoyama. Kajibuntan: A House Chore Division App. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 16449–16451, 2023.

Sungjin Im and Benjamin Moseley. Fair Scheduling via Iterative Quasi-Uniform Sampling. *SIAM Journal on Computing*, 49(3):658–680, 2020.

Joseph YT Leung. *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. CRC press, 2004.

Bo Li, Minming Li, and Ruilong Zhang. Fair Scheduling for Time-Dependent Resources. *Advances in Neural Information Processing Systems*, 34:21744–21756, 2021.

Richard J Lipton, Evangelos Markakis, Elchanan Mossel, and Amin Saberi. On Approximately Fair Allocations of Indivisible Goods. In *Proceedings of the 5th ACM Conference on Electronic Commerce*, pages 125–131, 2004.

Hervé Moulin. Fair Division in the Internet Age. *Annual Review of Economics*, 11:407–441, 2019.

James Oxley. Matroid Theory. In *Handbook of the Tutte Polynomial and Related Topics*, pages 44–85. Chapman and Hall/CRC, 2022.

Benjamin Plaut and Tim Roughgarden. Almost Envy-Freeness with General Valuations. *SIAM Journal on Discrete Mathematics*, 34(2):1039–1068, 2020.

Warut Suksompong. Constraints in Fair Division. *ACM SIGecom Exchanges*, 19(2):46–61, 2021.

Martino Traxler. Fair Chore Division for Climate Change. *Social Theory and Practice*, 28(1):101–134, 2002.

Douglas Brent West. *Introduction to Graph Theory*, volume 2. Prentice Hall Upper Saddle River, 2001.

Shengwei Zhou and Xiaowei Wu. Approximately EFX Allocations for Indivisible Chores. In *Proceedings of the 31st International Joint Conference on Artificial Intelligence*, pages 783–789, 2022.