Preprint – IJCAI 2025: This is the accepted version made available for conference attendees.
Do not cite. The final version will appear in the IJCAI 2025 proceedings.

EFX Feasible Scheduling for Time-dependent Resources

Jiazhu Fang[1], **Qizhi Fang**[1,2], **Minming Li**[3*] and **Wenjing Liu**[1,2]

[1] School of Mathematical Sciences, Ocean University of China, Qingdao, Shandong, China.
[2] Laboratory of Marine Mathematics, Ocean University of China, Qingdao, Shandong, China.
[3] Department of Computer Science, City University of Hong Kong, Kowloon Tong, Hong Kong, China.
fjz@stu.ouc.edu.cn, qfang@ouc.edu.cn, minming.li@cityu.edu.hk, liuwj@ouc.edu.cn

Abstract

In this paper, we study a fair resource scheduling problem involving the assignment of a set of interval jobs among a group of heterogeneous machines. Each job is associated with a release time, a deadline, and a processing time. A machine can process a job if the entire processing period falls within the release time and deadline of the job. Each machine can process at most one job at any given time, and different jobs yield different utilities for the machine. The goal is to find a fair and efficient schedule of the jobs. We discuss the compatibility between envy-freeness up to any item (EFX) and various efficiency concepts. Additionally, we present polynomial-time algorithms for various settings.

1 Introduction

The fair division problem aims to address how to allocate limited resources among multiple agents with individual preferences in a manner that is both fair and efficient. Classical literature on fair division primarily focuses on divisible resources [Deng *et al.*, 2012; Aziz and Mackenzie, 2016]. Recently, the fair division of indivisible resources has gained significant attention, where resources must be integrally allocated to agents. This has become an important research area in economics, operations research, and computer science [Brams and Taylor, 1996; Brandt *et al.*, 2016; Moulin, 2019]. It has numerous real-world applications, such as the fair division of courses [Budish *et al.*, 2017], public housing [Benabbou *et al.*, 2020], food donations [Aleksandrov *et al.*, 2015], and inheritance [Goldman and Procaccia, 2015].

An important fairness concept in fair division is *envy-freeness* (EF) [Varian, 1974], which requires that no agent prefers another agent's bundle over their own. However, EF allocation does not always exist for indivisible resources. For example, if there is only one resource and two agents, an EF allocation cannot be achieved. A significant relaxation of EF is the concept of *envy-freeness up to one item* (EF1) [Richard *et al.*, 2004], which allows for envy between two agents as long as there exists an item in the envied agent's bundle that, when removed, can eliminate the envy. [Caragiannis *et al.*, 2019b] proposed another notable fairness concept, *envy-freeness up to any good* (EFX), which balances between the stronger concept of EF and the weaker concept of EF1. EFX assumes that removing any positive value item from the envied agent's bundle can eliminate the envy. Unlike the universal existence of EF1, the existence of EFX allocations remains uncertain. Indeed, establishing the existence of EFX allocations is widely regarded as one of the core open problems in the fair division of indivisible resources.

In addition to fairness requirements, resource allocators also seek efficient allocations. A commonly used efficiency criterion is Pareto optimality (PO), where an allocation is Pareto optimal if no other allocation can make someone better off without making someone else worse off. [Caragiannis *et al.*, 2019b] analyzed the properties of allocations that maximize Nash social welfare (MaxNSW) and demonstrated that for additive valuation functions, there exist MaxNSW allocations that satisfy both EF1 and PO, where Nash social welfare is defined as the geometric mean of all agents' valuations [Kaneko and Nakamura, 1979]. However, since computing a MaxNSW allocation is NP-hard, this result only establishes the existence of EF1+PO allocations. Consequently, extensive research has aimed to design algorithms that can simultaneously optimize both fairness and efficiency [Barman *et al.*, 2018a; Barman and Krishnamurthy, 2019; Garg and Murhekar, 2023].

In most of the literature on fair division, any subset of items can be feasibly allocated to any agent. However, this assumption is not applicable in many scenarios. For example, in the Student Affairs Office (SAO) problem, SAO staff needs to allocate jobs to students applying for work, where each job has a release time, a deadline, and a consecutive processing time and students earn compensation by getting jobs. A feasible job allocation requires that the jobs assigned to a student can be scheduled without overlapping in time. Motivated by this scenario, [Li *et al.*, 2021] introduced the fair interval scheduling problem (FISP) where a set of interval jobs is allocated to heterogeneous machines controlled by agents. Each job is associated with a release time, a deadline, and a processing time, and it can be processed if its processing period falls between its release time and deadline. Each machine can only process one job at any given time, and different machines may derive different utilities from processing the same job.

[*] Corresponding Authors.

Additionally, all jobs assigned to the same machine must be processed without overlapping in time. They investigated the existence and computability of approximate maximin share fairness (MMS) and EF1 schedules.

However, the fairness guarantee of EF1, limited to the most valuable item, is often too weak. This paper continues the study of FISP and explores a stronger fairness concept, EFX. It aims to investigate the computability of schedules that satisfy EFX and various efficiency concepts.

1.1 Our Contribution

We study the fair interval scheduling problem (FISP), where fairness is captured by EFX. We also consider several efficiency concepts: MaxNSW, PO, and Weakly Individual Optimal (WIO).

Our main contributions are summarized as follows.

EFX + MaxNSW + PO. We first consider binary valuation functions, where agents' valuations for jobs are restricted to 1 or 0. We explore the characteristics of MaxNSW schedules, focusing on fairness guarantees (approximate EFX) and efficiency guarantees (PO).

Main Result 1: For any instance of FISP with binary valuations, there exists a MaxNSW schedule that is $\frac{1}{2}$-EFX and PO. Additionally, there exist instances where no MaxNSW schedule can guarantee $(\frac{1}{2}+\varepsilon)$-EFX for any $\varepsilon > 0$.

Main Result 2: For any instance of FISP with binary valuations and jobs with unit processing times, there exists a MaxNSW schedule that is EFX and PO, and it can be computed in polynomial time.

Next, we find that under general valuation functions, the approximation guarantee of EFX is related to the non-zero range parameter γ, defined as the ratio of the maximum valuation to the minimum non-zero valuation among all agents.

Main Result 3: For any instance of FISP, there exists a MaxNSW schedule that is $\frac{1}{\gamma^2}$-EFX and PO. When all jobs have unit processing times, there exists a MaxNSW schedule that is $\frac{1}{\gamma+1}$-EFX and PO. Additionally, in both settings, there exist instances where no MaxNSW schedule can guarantee $(\frac{1}{\gamma}+\varepsilon)$-EFX for any $\varepsilon > 0$.

EFX + WIO. [Li *et al.*, 2021] proved the incompatibility between Individual Optimal (IO) (no one envies the union of their assigned jobs and the unassigned jobs) and EF1, even for FISP with identical valuation functions. This implies that IO is also incompatible with the stronger fairness concept, EFX. Therefore, we consider the relaxed version of IO, known as WIO, which means that no one envies the unassigned jobs. We first provide an algorithm framework demonstrating the compatibility between WIO and EFX for all instances of FISP. Then, we show that finding a WIO schedule is NP-hard. Finally, we present a polynomial-time algorithm that approximates both EFX and WIO.

Main Result 4: There exists an algorithm that can return a feasible schedule that is both EFX and WIO for all FISP instances.

Main Result 5: For any $0 < \varepsilon < 1$, there exists a polynomial-time algorithm that can return a feasible schedule that is both $0.644(1-\varepsilon)$-EFX and 0.644-WIO for all FISP instances. The running time is polynomial in $|J|$, $|A|$, and $\frac{1}{\varepsilon}$, where $|J|$ is the number of jobs and $|A|$ is the number of agents.

1.2 Related Work

The scheduling problem. [Johnson and Garey, 1979] showed that computing the maximum feasible set of jobs is NP-hard. Subsequently, various approximation algorithms have been proposed [Chuzhoy *et al.*, 2006; Berman and DasGupta, 2000], with the currently best-known approximation ratio being 0.644 [Im *et al.*, 2020]. For instances with rigid jobs, [Schrijver, 1998] provided a polynomial-time algorithm to solve the problem. Various fairness criteria have also been proposed, including minimizing the maximum deviation from a desired load [Ajtai *et al.*, 1998], minimizing the ℓ_p norm of flow time [Im *et al.*, 2020], and analyzing the welfare degradation resulting from the imposition of fairness constraints [Bilò *et al.*, 2016].

EFX + MaxNSW/PO. We primarily review the literature on the compatibility of EFX with various efficiency concepts. For binary additive valuations, any MaxNSW allocation is EFX [Amanatidis *et al.*, 2021]. [Garg and Murhekar, 2023] showed that for instances with binary additive valuations, EFX and PO allocations can be computed in polynomial time, but for instances with three distinct values, EFX and PO are incompatible. [Babaioff *et al.*, 2021] proved that under submodular and dichotomous valuations, any MaxNSW allocation is EFX. [Caragiannis *et al.*, 2019a] proved that under additive valuation functions, there exist partial allocations that are EFX and $\frac{1}{2}$-MaxNSW (where some items may remain unallocated). [Garg *et al.*, 2023] showed that even under subadditive valuation functions, there exist complete allocations that are $\frac{1}{2}$-EFX and $\frac{1}{2}$-MaxNSW. [Feldman *et al.*, 2024] provided optimal trade-offs between EFX and MaxNSW for additive and subadditive valuation functions. [Dai *et al.*, 2024] investigated the relationship between EFX and MaxNSW under budget constraints, proving that for binary valuation functions, MaxNSW can guarantee $\frac{1}{4}$-EFX and PO. However, even under unconstrained additive valuations, computing a MaxNSW allocation is NP-hard [Ramezani and Endriss, 2010]. [Barman *et al.*, 2018b] showed that for binary additive valuations, a MaxNSW allocation that is both EFX and PO can be found in polynomial time. [Benabbou *et al.*, 2021] proved that for matroid rank functions, MaxNSW can be found in polynomial time.

The most relevant works to ours are [Li *et al.*, 2021] and [Kumar *et al.*, 2024]. [Li *et al.*, 2021] proposed the fair interval scheduling problem, and investigated fairness concepts of MMS and EF1. They showed that any MaxNSW schedule is $\frac{1}{4}$-EF1 and PO, and for jobs with unit processing times, it is $\frac{1}{2}$-EF1 and PO. They also introduced the efficiency concept of individual optimality (IO), and considered the compatibility of EF1 and IO. When switching EF1 to EFX, IO is difficult to achieve. Therefore, we consider a relaxed version of IO, known as weakly individual optimality (WIO), which is referred to as bounded charity in some literature [Barman *et al.*, 2023]. [Kumar *et al.*, 2024] considered the fair interval scheduling problem for indivisible chores by constructing interval graphs and studying the existence and computability of schedules that are both EF1 and maximal.

2 Preliminaries

2.1 Fair Interval Scheduling Problem

Problem instance. We follow the notation used by [Li *et al.*, 2021]. An instance I of the fair interval scheduling problem (FISP) is given by a tuple $(J, A, \mathbf{u}_A)$, where $J = \{j_1, \dots, j_n\}$ represents a set of n indivisible jobs and $A = \{a_1, \dots, a_m\}$ is a set of m agents (machines). The timeline consists of disjoint unit time slots $[0, 1)$, $[1, 2)$, $[2, 3)$, and so on. For any $t \in \mathbb{N}_+$, let $[t, t+1)$ denote the t-th time slot. Each $j_i \in J$ is associated with *release time* $r_i \in \mathbb{N}_+$, *deadline* $d_i \in \mathbb{N}_+$, and *processing time* $p_i \in \mathbb{N}_+$ such that $p_i \le d_i - r_i + 1$. We refer to $[r_i, d_i]$ as a job interval, which can be viewed as a set of contiguous time slots from r_i to d_i, denoted as $\{r_i, \dots, d_i\}$. If p_i contiguous time slots within $[r_i, d_i]$ are allocated to job j_i, then job j_i can be successfully processed. An agent can process at most one job in any given time slot. A set of jobs $J' \subseteq J$ is called *feasible* if all jobs in J' can be processed on a single machine without overlapping. Define $u_i : 2^J \to \mathbb{R}_{\ge 0}$ as the *valuation function* of agent $a_i \in A$, and $\mathbf{u}_A = \{u_1, \dots, u_m\}$ as the *valuation profile* of the m agents. We call these $u_i(\cdot)$ interval scheduling (IS) functions. For a job $j_k \in J$, if j_k is successfully processed by agent a_i, then agent a_i receives a utility $u_i(\{j_k\}) \ge 0$. For simplicity, denote $u_i(\{j_k\})$ as $u_i(j_k)$. For a feasible job set S, the utility of agent a_i is additive, i.e., $u_i(S) = \sum_{j_k \in S} u_i(j_k)$. For any infeasible job set S, the utility of agent a_i is the maximum value obtainable by processing a feasible subset of S, i.e.,

$$u_i(S) = \max_{S' \subseteq S : S' \text{ is feasible}} \sum_{j_k \in S'} u_i(j_k).$$

Non-zero range parameter. Part of our approximation guarantees for the fairness concept is related to the non-zero range parameter $\gamma \ge 1$, which depends on the range of non-zero valuations. Formally, for any given instance $I = (J, A, \mathbf{u}_A)$, the *non-zero range parameter* is defined as

$$\gamma := \frac{\max_{a_i \in A, j_k \in J} u_i(j_k)}{\min_{j_k \in J, a_i : u_i(j_k) > 0} u_i(j_k)}.$$

Schedule (Allocation). A schedule $\mathbf{X} = (X_1, \cdots, X_m)$ is defined as an ordered m-partition of a subset of J, where X_i is the set of jobs (or bundle) assigned to agent a_i. Thus, we have $X_1 \cup \cdots \cup X_m \subseteq J$ and $X_i \cap X_j = \emptyset$ for every pair of agents $a_i, a_j \in A$. Let $X_0 = J \setminus \bigcup_{i \in [m]} X_i$ denote all unscheduled jobs, which can be considered as donated to a *charity*. A schedule $\mathbf{X}$ is called *feasible* if X_i is feasible for all $a_i \in A$, i.e., all jobs in X_i can be successfully processed by a_i. A schedule $\mathbf{X}$ is called *non-wasteful* if $j_k \in X_i$ implies $u_i(j_k) > 0$ for all $j_k \in J, a_i \in A$, and *wasteful* otherwise.

Special instance class. Regarding agents' valuations, we consider (1) *Binary*: $u_i(j_k) \in \{0, 1\}$ for all $a_i \in A, j_k \in J$; (2) *Identical*: $u_i(j_k) = u_r(j_k)$ for all $a_i, a_r \in A, j_k \in J$; (3) *General*: $u_i(j_k) \ge 0$ without any restrictions. Regarding jobs, we consider (1) *Unit*: $p_i = 1$, for all $j_i \in J$, i.e., all jobs have unit processing time; (2) *Rigid*: $r_i + p_i - 1 = d_i$, for all $j_i \in J$, i.e., each job needs to occupy the entire time interval between release time and deadline; (3) *Flexible*: $r_i + p_i - 1 \le d_i$, for all $j_i \in J$. Note that unit jobs may not be rigid and rigid jobs may not be unit either. We use "FISP with <valuation type, job type>" to denote a special instance class of FISP.

2.2 Solution Concepts

We define the fairness and efficiency concepts considered in this paper as follows.

Fairness Concepts

Definition 1 (α-EF1 Schedule)**.** *For $0 < \alpha \le 1$, a feasible schedule $\mathbf{X} = (X_1, \cdots, X_m)$ is called α-approximate envy-free up to one item (α-EF1) if for any two agents $a_i, a_k \in A$, we have*

$$u_i(X_i) \ge \alpha \cdot u_i(X_k \setminus \{j\}) \text{ for some } j \in X_k.$$

Definition 2 (α-EFX Schedule)**.** *For $0 < \alpha \le 1$, a feasible schedule $\mathbf{X} = (X_1, \cdots, X_m)$ is called α-approximate envy-free up to any item (α-EFX) if for any two agents $a_i, a_k \in A$, we have*

$$u_i(X_i) \ge \alpha \cdot u_i(X_k \setminus \{j\}) \text{ for any } j \in X_k.$$

Definition 3 (α-EFX Envy)**.** *Given a feasible schedule $\mathbf{X} = (X_1, \cdots, X_m)$, for any two agents $a_i, a_k \in A$ and $0 < \alpha \le 1$, we say a_i α-EFX envies a_k if*

$$u_i(X_i) < \alpha \cdot u_i(X_k \setminus \{j\}) \text{ for some } j \in X_k.$$

Efficiency Concepts

Definition 4 (MaxNSW Schedule)**.** *A feasible schedule $\mathbf{X} = (X_1, \cdots, X_m)$ is called MaxNSW schedule if and only if*

$$\mathbf{X} \in \arg\max_{\mathbf{X}' \in \mathcal{F}} \left(\prod_{i=1}^{m} u_i(X'_i)\right)^{\frac{1}{m}}$$

where $\mathcal{F}$ is the set of all feasible schedules and $\mathbf{X}' = (X'_1, \cdots, X'_m)$.

Definition 5 (PO Schedule)**.** *A feasible schedule $\mathbf{X} = (X_1, \cdots, X_m)$ is called Pareto optimal (PO) if there does not exist an alternative feasible schedule $\mathbf{X}' = (X'_1, \cdots, X'_m)$ such that $u_i(X'_i) \ge u_i(X_i)$ for all $a_i \in A$, and $u_k(X'_k) > u_k(X_k)$ for some $a_k \in A$.*

Definition 6 (α-IO Schedule [Li *et al.*, 2021])**.** *For $0 < \alpha \le 1$, a feasible schedule $\mathbf{X} = (X_1, \cdots, X_m)$ with $X_0 = J \setminus \bigcup_{i \in [m]} X_i$ is called α-approximate individual optimal (α-IO) if $u_i(X_i) \ge \alpha \cdot u_i(X_0 \cup X_i)$ for all $a_i \in A$. When $\alpha = 1$, $\mathbf{X}$ is called an IO schedule.*

[Li *et al.*, 2021] proved the incompatibility between EF1 and IO even for FISP with <Identical, Rigid>. Therefore, we consider the relaxed version of IO, called WIO.

Definition 7 (α-WIO Schedule)**.** *For $0 < \alpha \le 1$, a feasible schedule $\mathbf{X} = (X_1, \cdots, X_m)$ with $X_0 = J \setminus \bigcup_{i \in [m]} X_i$ is called α-approximate weakly individual optimal (α-WIO) if $u_i(X_i) \ge \alpha \cdot u_i(X_0)$ for all $a_i \in A$. When $\alpha = 1$, $\mathbf{X}$ is called a WIO schedule.*

3 Approximately EFX and MaxNSW Scheduling

In this section, we establish interesting connections between EFX and MaxNSW schedules by analyzing the performance of the non-wasteful MaxNSW schedules. We also explore the conditions under which a schedule can be both EFX and PO.

Clearly, if an agent's valuation for a job is 0, a natural idea is not to assign that job to him, as doing so would not only waste machine resources but also fail to generate any social welfare. Therefore, we primarily concentrate on the non-wasteful MaxNSW schedules. It is noteworthy that for an arbitrarily wasteful schedule, we can always transfer the 0-valued jobs assigned to agents to the charity, resulting in a non-wasteful schedule with the same Nash social welfare. This also implies that the non-wasteful MaxNSW schedules must exist.

3.1 <Binary, Flexible>

We first consider the relationship between EFX and the non-wasteful MaxNSW schedules in binary valuation instances. Our main result shows that for any binary valuation instance I, there exists a MaxNSW schedule that is both $\frac{1}{2}$-EFX and PO. Moreover, the $\frac{1}{2}$-approximation is the best EFX guarantee achievable among all MaxNSW schedules.

The following results provide the worst-case EFX approximation guaranteed by the non-wasteful MaxNSW schedules on any instance I with MaxNSW$(I) > 0$. All the omitted proofs are presented in the full version of our paper.

Theorem 1. *Given an arbitrary instance I of FISP with <Binary, Flexible>, if MaxNSW$(I) > 0$, then any non-wasteful MaxNSW schedule is $\frac{1}{4}$-EFX and PO.*

However, for an instance I with MaxNSW$(I) = 0$, it is possible for a non-wasteful MaxNSW schedule to have an unbounded EFX approximation. We now consider the best possible EFX approximation guarantee achievable by the non-wasteful MaxNSW schedules.

We first establish the relationship between instances where the maximum Nash social welfare is greater than 0 and those where the maximum Nash social welfare equals 0, allowing us to focus solely on instances where MaxNSW$(I) > 0$ when analyzing the relationship between EFX and MaxNSW schedules.

Lemma 1. *For any FISP with <valuation type, job type>, if for all instances I with MaxNSW$(I) > 0$, there exists a non-wasteful MaxNSW schedule that is α-EFX and PO, then for any instance I' with MaxNSW$(I') = 0$, there must exist a non-wasteful MaxNSW schedule that is α-EFX and PO, where $0 < \alpha \leq 1$.*

Then, before presenting the main results of this subsection, we provide a key lemma that guarantees the elimination of $\frac{1}{2}$-EFX envy between agents for any binary valuation instance.

Lemma 2. *For an arbitrary instance I with MaxNSW$(I) > 0$ of FISP with <Binary, Flexible> and any non-wasteful MaxNSW schedule $\mathbf{X} = (X_1, \ldots, X_m)$, if there exist $i, k \in [m]$ such that a_i $\frac{1}{2}$-EFX envies a_k, then there exist a set $T \subset X_i$ containing $|X_i| - 1$ jobs and a set $S \subset X_k$ containing 2 jobs that a_i values non-zero, such that $T \cup S$ is feasible.*

Theorem 2. *Given an arbitrary instance I of FISP with <Binary, Flexible>, there exists a non-wasteful MaxNSW schedule which is $\frac{1}{2}$-EFX and PO.*

Proof. By Lemma 1, we only need to show that the conclusion holds for all instances I with MaxNSW$(I) > 0$. Theorem 1 has already proven that any non-wasteful MaxNSW schedule is PO. Below, we present the procedure to find a non-wasteful MaxNSW schedule that satisfies $\frac{1}{2}$-EFX.

For any non-wasteful MaxNSW schedule $\mathbf{X} = (X_1, \ldots, X_m)$ with $X_0 = J \setminus \cup_{i \in [m]} X_i$, if $\mathbf{X}$ is $\frac{1}{2}$-EFX, then the proof is complete. Otherwise, there must exist $i, k \in [m]$ such that a_i $\frac{1}{2}$-EFX envies a_k, i.e.,

$$u_i(X_i) < \frac{1}{2} \cdot u_i(X_k \setminus \{j_p\}), \exists j_p \in X_k. \tag{1}$$

By Lemma 2, there exist a set $T \subset X_i$ containing $|X_i| - 1$ jobs and a set $S \subset X_k$ containing 2 jobs that a_i values non-zero, such that $T \cup S$ is feasible for a_i.

Now we construct a new schedule $\mathbf{X}' = (X'_1, \ldots, X'_m)$, where $X'_r = X_r, \forall r \in [m], r \neq i, k$; $X'_i = T \cup S$; $X'_k = X_k \setminus S$; and $X'_0 = J \setminus \bigcup_{i \in [m]} X'_i$. We can observe that in the new schedule $\mathbf{X}'$, the utility of a_i increases by 1, the utility of a_k decreases by 2, and the utilities of other agents remain unchanged. Specifically, $u_i(X'_i) = u_i(X_i) + 1, u_k(X'_k) = u_k(X_k) - 2, u_r(X'_r) = u_r(X_r), \forall r \in [m], r \neq i, k$. Also, since

$$\begin{aligned} u_k(X_k) = u_k(X_k \setminus \{j_p\}) + 1 &\geq u_i(X_k \setminus \{j_p\}) + 1 \\ &\geq 2 \cdot u_i(X_i) + 2, \end{aligned}$$

where the first equality and the first inequality hold due to $\mathbf{X}$ being non-wasteful, and the last inequality holds due to inequality (1) and the fact that $u_i(X_i \setminus \{j_p\})$ is an integer, we have

$$\begin{aligned} u_i(X'_i) \cdot u_k(X'_k) &= (u_i(X_i) + 1) \cdot (u_k(X_k) - 2) \\ &= u_i(X_i) \cdot u_k(X_k) + u_k(X_k) \\ &\quad - (2 \cdot u_i(X_i) + 2) \\ &\geq u_i(X_i) \cdot u_k(X_k). \end{aligned}$$

Thus, by iterating this process, we can eliminate the $\frac{1}{2}$-EFX envy between a_i and a_k without losing Nash social welfare. In the same manner, we can eliminate $\frac{1}{2}$-EFX envy between any pair of agents, thereby obtaining a non-wasteful MaxNSW schedule that satisfies $\frac{1}{2}$-EFX. □

Remark 1. In fact, Theorem 2 provides a conversion procedure from any MaxNSW schedule to a $\frac{1}{2}$-EFX MaxNSW schedule. Each step that eliminates $\frac{1}{2}$-EFX envy results in an increase in the number of jobs allocated to charity, implying that a non-wasteful MaxNSW with the maximum number of charity jobs must be $\frac{1}{2}$-EFX. However, finding a MaxNSW schedule is NP-hard, and thus, we cannot implement this procedure efficiently.

We now provide an example to show that the $\frac{1}{2}$-approximation of EFX is optimal.

Theorem 3. *There exists an instance I of FISP with <Binary, Rigid> such that no MaxNSW schedule of I is $(\frac{1}{2}+\varepsilon)$-EFX, where $\varepsilon > 0$.*

3.2 <Binary, Unit>

[Li *et al.*, 2021] showed that EF1 and PO are incompatible even if jobs are rigid and valuations are unary, i.e., $u_i(j_k) = 1, \forall a_i \in A, \forall j_k \in J$. This implies that no algorithm can return a feasible schedule that is both EFX and PO for all instances of FISP with <Binary, Rigid>. Fortunately, we find that for any instance of FISP with <Binary, Unit>, there exists a non-wasteful MaxNSW schedule that is both EFX and PO and it can be found in polynomial time.

Theorem 4. *Given an arbitrary instance I of FISP with <Binary, Unit>, if MaxNSW$(I) > 0$, then any non-wasteful MaxNSW schedule is EFX and PO; if MaxNSW$(I) = 0$, then there exists a non-wasteful MaxNSW schedule that is EFX and PO.*

For any instance I of FISP with <Binary, Unit>, we present below a polynomial-time algorithm to find a MaxNSW schedule that is both EFX and PO. Algorithm 1 is based on the proof of Lemma 1. Specifically, it first maximizes the number of agents with non-zero utility by finding the maximum matching. Subsequently, agents not matched in the maximum matching are assigned empty bundles. For the sub-instance I' formed by removing the unmatched agents from I, it is clear that MaxNSW$(I') > 0$. According to the proof of Lemma 1 and Theorem 4, it suffices to find a non-wasteful MaxNSW schedule for I' in polynomial time.

For I', starting with any feasible schedule, the algorithm adopts the simplest schedule update method, where in each update, each agent can add at most one job and remove at most one job, represented by constructing a directed graph. At each update step, the algorithm selects the update that maximizes the Nash social welfare increment among all feasible updates following this method. Lemma 4 quantifies the Nash social welfare increment achieved by each such schedule update. Theorem 5 proves that by performing at most $(2m-1)\cdot n \cdot \ln\frac{4n^2}{m}$ scheduling updates, we can obtain a MaxNSW schedule that satisfies both EFX and PO. Lemma 3 guarantees that our algorithm can be completed in polynomial time.

A polynomial-time algorithm. The detailed description of the algorithm is as follows: Firstly, construct a bipartite graph $G(A \cup J, E)$, where an edge (a_i, j_k) exists if and only if $u_i(j_k) = 1$. Then, compute a maximum matching $M = A_M \cup J_M$ of G. If agent a_i is not matched, an empty bundle is assigned, i.e., $X_i = \emptyset$. Thus, we obtain a sub-instance $I' = (J, A_M, \mathbf{u}_{A_M})$ with MaxNSW$(I') > 0$. Finally, find a non-wasteful MaxNSW schedule for instance I'. To simplify notation, in the following, we still let $A_M = A$.

For the sub-instance I', each agent initially receives the job in the maximum matching M. In each subsequent iteration, the algorithm greedily finds a feasible schedule update that increases Nash social welfare (NSW). Specifically, for any iteration t, the algorithm construct a directed graph $G'(\mathbf{X}^{t-1})$ based on the current schedule $\mathbf{X}^{t-1}$ and $X_0^{t-1} = J \setminus \cup_{i\in[m]} X_i^{t-1}$, where the vertex set is $V = V_1 \cup V_2$. Here, $V_1 = \{v_0, v_1, \dots, v_m\}$ represents charity and all agents, and vertices in V_1 are referred to as *agent vertices*. $V_2 = \{j_{0,1} \dots j_{0,d_0}, j_{1,1} \dots j_{1,d_1} \dots j_{m,1} \dots j_{m,d_m}\}$, where $\forall j_{k,i} \in V_2$ represents the i-th job in bundle X_k^{t-1} of agent a_k (job indices in X_k^{t-1} are arbitrary). Specifically, $j_{0,i}$ represents the i-th job in charity bundle X_0^{t-1}. Vertices in V_2 are referred to as *job vertices*.

- A directed edge $(j_{k,i}, a_d)$, where $d \neq 0, k \neq d$ exists if and only if $u_d(j_{k,i}) = 1$ and $X_d^{t-1} \cup \{j_{k,i}\}$ is feasible.
- A directed edge $(j_{k,i}, a_0)$ exists if and only if $j_{k,i} \notin X_0^{t-1}$, i.e. $k \neq 0$.
- A directed edge $(a_d, j_{k,i})$ exists if and only if $j_{k,i} \in X_d^{t-1}$, i.e. $k = d$.
- A directed edge $(j_{k,i}, j_{k',i'})$, where $k' \neq 0$ exists if and only if $u_{k'}(j_{k,i}) = 1$ and $X_{k'}^{t-1} \cup \{j_{k,i}\}$ is not feasible and $X_{k'}^{t-1} \setminus \{j_{k',i'}\} \cup \{j_{k,i}\}$ is feasible.

Remark 2. Note that in determining the feasibility of a job set, we utilize the definition of the condensed instance [Li *et al.*, 2021] to improve the running time.

Lemma 3. *For any iteration $1 \leq t \leq (2m-1)\cdot n \cdot \ln\frac{4n^2}{m}$, the directed graph $G'(\mathbf{X}^{t-1})$ can be constructed in polynomial time.*

Below, we define a feasible schedule update method for each type of directed edge:

- A directed edge $(j_{k,i}, a_d)$ represents moving job $j_{k,i}$ from bundle X_k^{t-1} to bundle X_d^{t-1}. i.e., $X_k^{t-1} = X_k^{t-1} \setminus \{j_{k,i}\}$ and $X_d^{t-1} = X_d^{t-1} \cup \{j_{k,i}\}$.
- A directed edge $(a_d, j_{k,i})$ represents no change in scheduling. i.e., $X_k^{t-1} = X_k^{t-1}$ and $X_d^{t-1} = X_d^{t-1}$.
- A directed edge $(j_{k,i}, j_{k',i'})$ represents moving job $j_{k,i}$ from bundle X_k^{t-1} to bundle $X_{k'}^{t-1}$. i.e., $X_k^{t-1} = X_k^{t-1} \setminus \{j_{k,i}\}$ and $X_{k'}^{t-1} = X_{k'}^{t-1} \setminus \{j_{k',i'}\} \cup \{j_{k,i}\}$.

For the directed graph $G'(\mathbf{X}^{t-1})$, we can find all feasible pairs (Definition 8) in polynomial time, and denote the set of these pairs as S. For any feasible pair $(a_k, a_d) \in S$, it is not difficult to compute the Nash social welfare after updating the schedule according to the paths between them (Observation 1). Therefore, the algorithm greedily find the feasible pair (a_{k^*}, a_{d^*}) that maximizes the Nash social welfare after updating the schedule. If the Nash social welfare does not increase after the update, we determine that the current schedule is a MaxNSW schedule, and output the schedule $\mathbf{X}^{t-1}$. Otherwise, we update the schedule along any directed path from a_{k^*} to a_{d^*}, obtaining a new schedule $\mathbf{X}^t$, and continue the iteration. It turns out that the algorithm terminates after running at most $(2m-1)\cdot n \cdot \ln\frac{4n^2}{m}$ iterations.

Definition 8. *A pair of agent vertices (a_k, a_d) is called a feasible pair if it is reachable from a_k to a_d, that is, there exists at least one directed path from a_k to a_d.*

Observation 1. *For directed graph $G'(\mathbf{X}^{t-1})$, we have the following results:*

Algorithm 1 Greedy Algorithm

Input: An arbitrary instance $I = (J, A, \mathbf{u}_A)$ of FISP with <Binary, Unit>.
Output: A non-wasteful MaxNSW schedule $\mathbf{X} = (X_1, \ldots, X_m)$ that is both EFX and PO.

Initialize $X_1 = \ldots = X_m = \emptyset$, $X_0 = J$.
Constructing the bipartite graph $G(A \cup J, E)$ and compute a maximum (weighted) matching $M = A_M \cup J_M$.
for each $(a_i, j_k) \in M$ **do**
  $X_i = X_i \cup \{j_k\}$
end for
Let $\mathbf{X}^c = (X_i)_{a_i \in A \setminus A_M}$, $\mathbf{X}^0 = (X_i)_{a_i \in A_M}$.
for $t = 1$ to $(2m-1) \cdot n \cdot \ln \frac{4n^2}{m}$ **do**
  Constructing the directed graph $G'(\mathbf{X}^{t-1})$ corresponding to the current schedule $\mathbf{X}^{t-1}$.
  Let $S = \{(a_k, a_d) \in A_M \cup \{a_0\} \times A_M \cup \{a_0\} : (a_k, a_d)$ is a feasible pair$\}$.
  for each $(a_k, a_d) \in S$ **do**
    $\text{NSW}(\mathbf{X}^{t-1}((a_k, a_d))) \leftarrow$ Calculate the updated NSW according to (a_k, a_d).
  end for
  if $\max_{(a_k,a_d)\in S} \text{NSW}(\mathbf{X}^{t-1}(a_k, a_d)) > \text{NSW}(\mathbf{X}^{t-1})$ **then**
    Let $(a_{k^*}, a_{d^*}) \in \arg\max_{(a_k,a_d)\in S} \text{NSW}(\mathbf{X}^{t-1}(a_k, a_d))$
    Find a path $P = \{a_{k^*}, \ldots a_{d^*}\}$.
    Update $\mathbf{X}^t = \mathbf{X}^{t-1}(P)$, where $\mathbf{X}^{t-1}(P)$ is the schedule update according to path P.
  else
    return $\mathbf{X} = (\mathbf{X}^c, \mathbf{X}^{t-1})$
  end if
end for

(1) If a directed path P ends at some agent vertex, then the schedule after updating along P is feasible.

(2) If (a_k, a_d) is a feasible pair and there exist multiple paths from a_k to a_d, updating the schedule along different paths results in different schedules with the same Nash social welfare.

(3) If C is a directed cycle, the NSW of the schedule remains unchanged after updating along it.

We prove below that in each iteration before reaching MaxNSW, there must exist a feasible pair such that updating along any path between them guarantees a lower bound on the increase of NSW.

Lemma 4. *For any iteration $1 \le t \le (2m-1) \cdot n \cdot \ln \frac{4n^2}{m}$, if NSW$(\mathbf{X}^{t-1}) <$ MaxNSW(I'), then there must exist a feasible pair (a_k, a_d) in the directed graph $G'(\mathbf{X}^{t-1})$, and adjusting along any directed path P from a_k to a_d results in a schedule $\mathbf{X}^{t-1}(P)$ that satisfies*

$$\ln(\mathit{MaxNSW}(I')) - \ln(\mathit{NSW}(\mathbf{X}^{t-1}(P))) \le (1 - \frac{1}{n})(\ln(\mathit{MaxNSW}(I')) - \ln(\mathit{NSW}(\mathbf{X}^{t-1}))).$$

Theorem 5. *Given an arbitrary instance I of FISP with <Binary, Unit>, a non-wasteful MaxNSW schedule which is both EFX and PO can be found in polynomial time.*

3.3 General Valuation

We consider the general instances of FISP and find that the EFX approximation guarantee achieved by the non-wasteful MaxNSW schedules is related to the non-zero range parameter γ.

Theorem 6. *Given an arbitrary instance I of FISP, when $\gamma \ge \sqrt{6}$, if MaxNSW$(I) > 0$, then any non-wasteful MaxNSW schedule is $\frac{1}{\gamma^2}$-EFX and PO; if MaxNSW$(I) = 0$, then there exists a non-wasteful MaxNSW schedule that is $\frac{1}{\gamma^2}$-EFX and PO.*

The results show that the EFX approximation guarantee achievable by the non-wasteful MaxNSW schedules is inversely related to the non-zero range parameter γ. This implies that the larger the disparity in agents' valuations of items, the worse the fairness guarantee provided by the MaxNSW schedules.

Theorem 7. *Given an arbitrary instance I of FISP, when $\gamma < \sqrt{6}$, if MaxNSW$(I) > 0$, then any non-wasteful MaxNSW schedule is $\frac{1}{6}$-EFX and PO; if MaxNSW$(I) = 0$, then there exists a non-wasteful MaxNSW schedule that is a $\frac{1}{6}$-EFX and PO.*

Theorem 8. *There exists an instance I of FISP with <Identical, Unit> such that no MaxNSW schedule of I is $(\frac{1}{\gamma} + \varepsilon)$-EFX, where $\varepsilon > 0$.*

If we restrict the job types to unit jobs, we can obtain a nearly tight EFX approximation guarantee.

Theorem 9. *Given an arbitrary instance I of FISP with <General, Unit>, if MaxNSW$(I) > 0$, then any non-wasteful MaxNSW schedule is $\frac{1}{\gamma+1}$-EFX and PO; if MaxNSW$(I) = 0$, then there exists a non-wasteful MaxNSW schedule that is $\frac{1}{\gamma+1}$-EFX and PO.*

4 Approximately EFX and WIO Scheduling

[Li *et al.*, 2021] proved the incompatibility between IO and EF1 even for FISP with <Identical, Rigid>. This implies that the stronger fairness concept EFX is also incompatible with IO. Therefore, we consider a relaxed concept WIO. [Barman *et al.*, 2023] proved the existence of EFX with bounded charity under generalized assignment constraints. Utilizing their idea framework, we first provide an algorithm demonstrating compatibility between WIO and EFX for all instances of FISP. The high-level idea of the algorithm is as follows: if there exist agents who envy the charity, we identify "the most envious" agent among them and find a bundle in the charity that this agent envies but no other agent EFX envies. Then "the most envious" agent selects the most valuable feasible subset from this bundle, and the remaining jobs, along with the bundle previously owned by the agent, are returned to the charity. This process is repeated until no agent envies the charity.

Algorithm 2 Envy-Bundle Elimination

Input: An arbitrary FISP instance $I = (J, A, \mathbf{u}_A)$.
Output: An EFX and WIO schedule.

Initialize: Schedule $\boldsymbol{X} = (X_1, \cdots, X_m) = (\emptyset, \cdots, \emptyset)$ and charity $X_0 = J$.
while there is an agent $a_i \in A$ with $u_i(X_i) < u_i(X_0)$ **do**
  Set $B = X_0$ and $s = i$.
  while there exists an agent $a_k \in A$ and a job $j_t \in B$ such that $u_k(X_k) < u_k(B \backslash \{j_t\})$ **do**
    Set $B = B \backslash \{j_t\}$ and $s = k$.
  end while
  Let $C \subseteq B$ be a feasible subset such that $\sum_{j_l \in C} u_s(j_l) = u_s(B)$.
  Set $X_s = C$ and $X_0 = J \backslash \cup_{i \in [m]} X_i$.
end while
return $\mathbf{X} = (X_1, \cdots, X_m)$ and $X_0 = J \backslash \cup_{i \in [m]} X_i$

Then, we show that computing a WIO schedule is NP-hard. Finally, we present a polynomial-time algorithm that approximates both EFX and WIO.

4.1 Compatibility of EFX and WIO

In the following, we present the specific algorithm: the initialization of the algorithm begins by assigning an empty bundle $\boldsymbol{X} = (\emptyset, \cdots, \emptyset)$ to each agent, while the charity holds all the jobs, i.e., $X_0 = J$. Whenever there exists an agent who envies the charity, we perform the following operations on the charity's bundle X_0: continuously remove jobs j_t from X_0 that still leave some agent envious after their removal, until we find a bundle B such that only one agent a_s envies B, and no other agents EFX envies B. Then a_s gets a feasible subset $C \subseteq B$ with $\sum_{j_l \in C} u_s(j_l) = u_s(B)$, and the bundle previously owned by a_s, and the jobs removed from the charity and $B \backslash C$ are returned to the charity. The algorithm continues until no agents envy the charity, as illustrated by Algorithm 2.

Note that we update the schedule only after each iteration of the outer while loop. We denote by $\boldsymbol{X}^l = (X_1^l, \ldots, X_m^l)$ and X_0^l the schedule and charity updated after the l-th iteration of the outer while loop in Algorithm 2, respectively. Let $\boldsymbol{X}^0 = \{\emptyset, \ldots, \emptyset\}$ and $X_0^0 = J$.

Lemma 5. *After any l-th ($l \geq 0$) iteration of the outer while loop in Algorithm 2, $\boldsymbol{X}^l = (X_1^l, \ldots, X_m^l)$ is feasible and EFX.*

Theorem 10. *EFX + WIO are compatible for FISP, i.e., there exists an algorithm that can return a feasible schedule that is simultaneously EFX and WIO for all FISP instances.*

4.2 Computational Hardness of the Problem

We now provide a proof showing that computing WIO schedule alone is NP-hard.

Theorem 11. *Given an arbitrary instance I of FISP, computing a WIO schedule is NP-hard.*

Algorithm 3 Efficient Implementation

Input: An arbitrary FISP instance $I = (J, A, \mathbf{u}_A)$; β-approximation polynomial-time algorithm for IS functions, a parameter $\varepsilon \in (0, 1)$.
Output: An $\beta(1-\varepsilon)$-EFX and β-WIO schedule.

Initialize: Schedule $\boldsymbol{X} = (X_1, \cdots, X_m) = (\emptyset, \cdots, \emptyset)$ and charity $X_0 = J$.
while there is an agent $a_i \in A$ with $u_i(X_i) < u_i'(X_0)$ **do**
  Set $B = X_0$ and $s = i$.
  while there exists an agent $a_k \in A$ and a job $j_t \in B$ such that $u_k(X_k) < (1-\varepsilon)u_k'(B \backslash \{j_t\})$ **do**
    Set $B = B \backslash \{j_t\}$ and $s = k$.
  end while
  Let $C \subseteq B$ be a feasible subset such that $\sum_{j_l \in C} u_s(j_l) = u_s'(B)$.
  Set $X_s = C$ and $X_0 = J \backslash \cup_{i \in [m]} X_i$.
end while
return $\mathbf{X} = (X_1, \cdots, X_m)$ and $X_0 = J \backslash \cup_{i \in [m]} X_i$

4.3 Polynomial-time Implementation

Note that Algorithm 2 is inefficient, because if $P \neq NP$, the exact value of the IS function cannot be computed in polynomial time. For special case of rigid or unit jobs, the IS function can be computed in polynomial time. Therefore, in this subsection, we present a polynomial-time algorithm for computing approximate EFX and WIO schedule. For the IS function, we can directly use a β-approximation algorithm, with the most well-known approximation ratio being 0.644 [Im *et al.*, 2020]. For each $a_i \in A$, we use $u_i' : 2^J \to \mathbb{R}^+$ to denote the approximate valuation, and thus $u_i'(S) \geq \beta \cdot u_i(S)$ for any $S \subseteq J$. Thus, we directly obtain Algorithm 3 and the following theorem.

Theorem 12. *For any $0 < \varepsilon < 1$, Algorithm 3 returns a $\beta(1-\varepsilon)$-EFX and β-WIO schedule for arbitrary FISP instance with a β-approximation algorithm for IS functions. The running time is polynomial with $|J|$, $|A|$ and $\frac{1}{\varepsilon}$.*

Corollary 1. *For any $0 < \varepsilon < 1$, given a instance I of FISP, an $0.644(1-\varepsilon)$-EFX and 0.644-WIO schedule can be found in polynomial time; when jobs are either rigid or unit, an $(1-\varepsilon)$-EFX and WIO schedule can be found in polynomial time.*

5 Conclusion and Future Work

This paper studied the fair scheduling problem with time-dependent resources, considering the compatibility between the concepts of fairness, mainly EFX, and efficiency, mainly MaxNSW, WIO and PO. Moreover, we designed polynomial-time algorithms that satisfy various fairness and efficiency concepts. Despite some progress, several important issues remain unresolved. A direct question is whether the relationship between MaxNSW and approximate EFX can be made tight. An interesting direction is to consider the relationship between approximate MaxNSW and approximate EFX. Can we design polynomial-time algorithms that provide strong approximation guarantees for both EF and MaxNSW?

Ethical Statement

There are no ethical issues.

Acknowledgments

This research was supported in part by the National Natural Science Foundation of China (12201590, 12171444) and Natural Science Foundation of Shandong Province (ZR2024MA031).

References

[Ajtai *et al.*, 1998] 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.

[Aleksandrov *et al.*, 2015] Martin Aleksandrov, Haris Aziz, Serge Gaspers, and Toby Walsh. Online fair division: analysing a food bank problem. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, page 2540–2546, Buenos Aires, Argentina, 2015. AAAI Press.

[Amanatidis *et al.*, 2021] Georgios Amanatidis, Georgios Birmpas, Aris Filos-Ratsikas, Alexandros Hollender, and Alexandros A Voudouris. Maximum nash welfare and other stories about efx. *Theoretical Computer Science*, 863:69–85, 2021.

[Aziz and Mackenzie, 2016] Haris Aziz and Simon Mackenzie. A discrete and bounded envy-free cake cutting protocol for any number of agents. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 416–427, Los Alamitos, CA, USA, 2016. IEEE Computer Society.

[Babaioff *et al.*, 2021] Moshe Babaioff, Tomer Ezra, and Uriel Feige. Fair and truthful mechanisms for dichotomous valuations. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(6):5119–5126, 2021.

[Barman and Krishnamurthy, 2019] Siddharth Barman and Sanath Kumar Krishnamurthy. On the proximity of markets with integral equilibria. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):1748–1755, 2019.

[Barman *et al.*, 2018a] Siddharth Barman, Sanath Kumar Krishnamurthy, and Rohit Vaish. Finding fair and efficient allocations. In *Proceedings of the 2018 ACM Conference on Economics and Computation*, page 557–574, Ithaca, NY, USA, 2018. Association for Computing Machinery.

[Barman *et al.*, 2018b] Siddharth Barman, Sanath Kumar Krishnamurthy, and Rohit Vaish. Greedy algorithms for maximizing nash social welfare. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, page 7–13, Stockholm, Sweden, 2018. International Foundation for Autonomous Agents and Multiagent Systems.

[Barman *et al.*, 2023] Siddharth Barman, Arindam Khan, Sudarshan Shyam, and K. V. N. Sreenivas. Guaranteeing envy-freeness under generalized assignment constraints. In *Proceedings of the 24th ACM Conference on Economics and Computation*, page 242–269, , London, United Kingdom,, 2023. Association for Computing Machinery.

[Benabbou *et al.*, 2020] Nawal Benabbou, Mithun Chakraborty, Xuan-Vinh Ho, Jakub Sliwinski, and Yair Zick. The price of quota-based diversity in assignment problems. *ACM Trans. Econ. Comput.*, 8(3), 2020.

[Benabbou *et al.*, 2021] Nawal Benabbou, Mithun Chakraborty, Ayumi Igarashi, and Yair Zick. Finding fair and efficient allocations for matroid rank valuations. *ACM Trans. Econ. Comput.*, 9(4), 2021.

[Berman and DasGupta, 2000] Piotr Berman and Bhaskar DasGupta. Multi-phase algorithms for throughput maximization for real-time scheduling. *Journal of Combinatorial Optimization*, 4:307–323, 2000.

[Bilò *et al.*, 2016] 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.

[Brams and Taylor, 1996] Steven J Brams and Alan D Taylor. *Fair Division: From cake-cutting to dispute resolution*. Cambridge University Press, Cambridge, UK, 1996.

[Brandt *et al.*, 2016] Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel D Procaccia. *Handbook of computational social choice*. Cambridge University Press, Cambridge, UK, 2016.

[Budish *et al.*, 2017] 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.

[Caragiannis *et al.*, 2019a] Ioannis Caragiannis, Nick Gravin, and Xin Huang. Envy-freeness up to any item with high nash welfare: The virtue of donating items. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, page 527–545, Phoenix, AZ, USA, 2019. Association for Computing Machinery.

[Caragiannis *et al.*, 2019b] Ioannis Caragiannis, David Kurokawa, Hervé Moulin, Ariel D. Procaccia, Nisarg Shah, and Junxing Wang. The unreasonable fairness of maximum nash welfare. *ACM Trans. Econ. Comput.*, 7(3), 2019.

[Chuzhoy *et al.*, 2006] Julia Chuzhoy, Rafail Ostrovsky, and Yuval Rabani. Approximation algorithms for the job interval selection problem and related scheduling problems. *Mathematics of Operations Research*, 31(4):730–738, 2006.

[Dai *et al.*, 2024] Sijia Dai, Guichen Gao, Shengxin Liu, Boon Han Lim, Li Ning, Yicheng Xu, and Yong Zhang. Maximum nash social welfare under budget-feasible efx. *IEEE Transactions on Network Science and Engineering*, 11(2):1810–1820, 2024.

[Deng *et al.*, 2012] Xiaotie Deng, Qi Qi, and Amin Saberi. Algorithmic solutions for envy-free cake cutting. *Operations Research*, 60(6):1461–1476, 2012.

[Feldman *et al.*, 2024] Michal Feldman, Simon Mauras, and Tomasz Ponitka. On optimal tradeoffs between efx and nash welfare. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(9):9688–9695, 2024.

[Garg and Murhekar, 2023] Jugal Garg and Aniket Murhekar. Computing fair and efficient allocations with few utility values. *Theoretical Computer Science*, 962:113932, 2023.

[Garg *et al.*, 2023] Jugal Garg, Edin Husić, Wenzheng Li, László A. Végh, and Jan Vondrák. Approximating nash social welfare by matching and local search. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, page 1298–1310, Orlando, FL, USA, 2023. Association for Computing Machinery.

[Goldman and Procaccia, 2015] Jonathan Goldman and Ariel D. Procaccia. Spliddit: unleashing fair division algorithms. *SIGecom Exch.*, 13(2):41–46, 2015.

[Im *et al.*, 2020] Sungjin Im, Shi Li, and Benjamin Moseley. Breaking 1 - 1/e barrier for nonpreemptive throughput maximization. *SIAM Journal on Discrete Mathematics*, 34(3):1649–1669, 2020.

[Johnson and Garey, 1979] David S Johnson and Michael R Garey. *Computers and intractability: A guide to the theory of NP-completeness*. WH Freeman, New York, NY, USA, 1979.

[Kaneko and Nakamura, 1979] Mamoru Kaneko and Kenjiro Nakamura. The nash social welfare function. *Econometrica*, 47(2):423–435, 1979.

[Kumar *et al.*, 2024] Yatharth Kumar, Sarfaraz Equbal, Rohit Gurjar, Swaprava Nath, and Rohit Vaish. Fair scheduling of indivisible chores. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, page 2345–2347, , Auckland, New Zealand, 2024. International Foundation for Autonomous Agents and Multiagent Systems.

[Li *et al.*, 2021] Bo Li, Minming Li, and Ruilong Zhang. Fair scheduling for time-dependent resources. In *Advances in Neural Information Processing Systems*, volume 34, pages 21744–21756, , NY, USA, 2021. Curran Associates, Inc.

[Moulin, 2019] Hervé Moulin. Fair division in the internet age. *Annual Review of Economics*, 11:407–441, 2019.

[Ramezani and Endriss, 2010] Sara Ramezani and Ulle Endriss. Nash social welfare in multiagent resource allocation. In *Agent-Mediated Electronic Commerce. Designing Trading Strategies and Mechanisms for Electronic Markets*, pages 117–131, Berlin, Heidelberg, Germany, 2010. Springer Berlin Heidelberg.

[Richard *et al.*, 2004] Lipton Richard, Markakis Evangelos, Mossel Elchanan, and Saberi Amin. On approximately fair allocations of indivisible goods. In *Proceedings of the 5th ACM Conference on Electronic Commerce*, page 125–131, New York, NY, USA, 2004. Association for Computing Machinery.

[Schrijver, 1998] Alexander Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, Hoboken, NJ, USA, 1998.

[Varian, 1974] Hal R. Varian. Equity, envy, and efficiency. *Journal of Economic Theory*, 9(1):63–91, 1974.