Proportionally Fair Makespan Approximation

Michal Feldman[1,2], **Jugal Garg**[3], **Vishnu V. Narayan**[1], **Tomasz Ponitka**[1]

[1]Tel Aviv University
[2]Microsoft ILDC
[3]University of Illinois at Urbana Champaign, USA

Abstract

We study fair mechanisms for the classic job scheduling problem on unrelated machines with the objective of minimizing the makespan. This problem is equivalent to minimizing the egalitarian social cost in the fair division of chores. The two prevalent fairness notions in the fair division literature are envy-freeness and proportionality. Prior work has established that no envy-free mechanism can provide better than an $\Omega(\log m/\log\log m)$-approximation to the optimal makespan, where m is the number of machines, even when payments to the machines are allowed. In strong contrast to this impossibility, our main result demonstrates that there exists a proportional mechanism (with payments) that achieves a $3/2$-approximation to the optimal makespan, and this ratio is tight. To prove this result, we provide a full characterization of allocation functions that can be made proportional with payments. Furthermore, we show that for instances with normalized costs, there exists a proportional mechanism that achieves the optimal makespan. We conclude with important directions for future research concerning other fairness notions, including relaxations of envy-freeness. Notably, we show that the technique leading to the impossibility result for envy-freeness does not extend to its relaxations.

Extended version — https://arxiv.org/abs/2412.08572

1 Introduction

We consider the problem of fairly and efficiently scheduling a set $[n]$ of indivisible jobs on a collection $[m]$ of machines. Each machine-job pair (i, j) has an associated processing time $c_{i,j}$. A schedule is an assignment of jobs to machines, where the processing time of machine i is the sum of $c_{i,j}$ over all jobs j assigned to machine i. A major objective in job scheduling is to minimize the *makespan*, which is the processing time of the machine with the largest processing time (i.e., the time needed to complete all the jobs).

The makespan minimization problem has been of great interest to the algorithms community. The seminal work of Lenstra, Shmoys, and Tardos (1990) provided a polynomial-time 2-approximation algorithm for this problem and showed that no poly-time algorithm can guarantee a factor less than $3/2$ unless $\mathsf{P} = \mathsf{NP}$. However, beyond computational efficiency, there are additional properties that may be desired. A natural question arises: how do these other desiderata affect the makespan approximation?

A paradigmatic example, introduced by Nisan and Ronen (1999), considers scenarios where the processing times of jobs on each machine are private information, and one seeks a *truthful* mechanism—namely a schedule combined with payments to machines—that incentivizes machines to report their processing times truthfully. This problem has been foundational in the initiation of the field of algorithmic mechanism design. Nisan and Ronen (1999) conjectured that any truthful deterministic mechanism incurs a factor-m loss in the makespan compared to the optimal solution (even in the absence of any computational constraints), sparking a significant body of research which culminated in the recent proof by Christodoulou, Koutsoupias, and Kovács (2023) confirming their conjecture.

Another major desideratum, which is the focus of this paper, is *fairness*. There is a large body of literature on the *fair division* problem, which asks how to divide a collection of undesirable items (chores) among a set of agents in a manner that is fair to every agent (see the recent surveys of (Amanatidis et al. 2023; Liu et al. 2024)). The job scheduling problem can be interpreted as a fair division problem by viewing jobs as chores and machines as agents. In this setting, the makespan objective aligns with the egalitarian social cost.

The fair division problem has a rich history, beginning with the seminal work of Steinhaus (1948), who proposed *proportionality* as a fairness objective. This notion ensures that each agent's cost does not exceed their fair share (a $1/m$ fraction of the total cost). Proportionality is one of the key fairness notions in the extensive research on the fair division of chores. Another prominent fairness notion is *envy-freeness*, introduced by Foley (1967), where every agent weakly prefers their own allocation over that of any other agent. It is known that for job scheduling, both envy-freeness and proportionality can be achieved through a mechanism with payments (Aragones 1995; Hartline et al. 2008; Cohen et al. 2010). We note that payments are necessary to achieve either notion; this can be demonstrated via a simple instance with one indivisible job and two agents. In this work, the main question we ask is the following:

Does there exist an assignment of jobs to machines (with payments) such that the resulting allocation is *fair* and also has a *small makespan*?

In other words, what is the *price of fairness* for minimizing the makespan in the job scheduling problem? The price of fairness is an important notion that has been well-studied in various fair division settings.

For the notion of envy-freeness with payments, Cohen et al. (2010) demonstrated that no assignment can guarantee a makespan less than $\Omega(\log m/\log\log m)$ times the optimal makespan[1], improving an earlier bound of $(2-1/m)$ by Hartline et al. (2008). This shows that requiring envy-freeness necessarily incurs a significant increase in the best-possible makespan (even without any computational constraints). In strong contrast to this impossibility, we present surprising positive results for the proportionality notion.

1.1 Our Results

Our primary focus in this work is on proportional mechanisms for the job scheduling problem. In Section 3, we begin by characterizing all *proportionable* job allocations, i.e., allocations that can be made proportional through payments.

We analyze proportional mechanisms in two key settings: a general setting with arbitrary cost functions and a normalized setting where the total cost for each machine equals a common value[2]. Our main results on proportionality are summarized and compared with known results on envy-freeness in Table 1.

For the general setting, we introduce the Anti-Diagonal Mechanism (Algorithm 1), enabling us to demonstrate the following result.

Theorem 1. *There is a proportional mechanism for the job scheduling problem over general instances $\mathcal{C}$ that gives a $3/2$-approximation to the optimal makespan.*

We also prove that the result above is tight.

Theorem 2. *For every case where $n \geq m$, there is no proportional mechanism for the job scheduling problem over general instances $\mathcal{C}$ that gives a $(3/2-\epsilon)$-approximation to the optimal makespan for any $\epsilon > 0$.*

The Anti-Diagonal Mechanism used in the proof of Theorem 1 takes as input an arbitrary allocation, and outputs an allocation that is proportionable and has a makespan that is at most $3/2$ times the makespan of the input allocation. Together with the known polynomial-time 2-approximation algorithm for the optimal makespan (Lenstra, Shmoys, and Tardos 1990), this implies the existence of a polynomial-time algorithm that finds a proportionable allocation with a constant-factor approximation of the optimal makespan.

For normalized instances, we show that the above result can be improved to achieve the optimal makespan.

Theorem 3. *There is a proportional mechanism for the job scheduling problem over normalized instances $\mathcal{N}$ that attains the optimal makespan.*

[1]We note that the improved bound of $\Omega(\log m)$ in (Fiat and Levavi 2012) has a bug that, to the best of our knowledge, remains unresolved.

[2]The main objective we study, the makespan, is not scale-free, since scaling the cost functions can alter both the makespan-minimizing allocation and the resulting makespan. We therefore distinguish between these cases and provide an analysis for both.

Additionally, we establish that the logarithmic lower bound for envy-freeness holds even for normalized instances, which further widens the gap between the best proportional mechanism and any envy-free mechanism compared to general instances (see Lemma 4.2).

Furthermore, we extend our analysis of the job scheduling problem to the dual problem of the fair division of goods with the objective of maximizing the egalitarian welfare, formally introduced in Section 5. For general valuations, we demonstrate that, unlike the job scheduling problem, the goods allocation problem admits no proportional approximation to the optimal makespan.

Theorem 4. *For every case where $n \geq m$, there is no proportional mechanism for the goods allocation problem over general instances $\mathcal{C}$ that gives a β-approximation to the optimal egalitarian welfare for any $\beta > 0$.*

We also show that for normalized instances the situation for goods allocation mirrors that of job scheduling.

Theorem 5. *There is a proportional mechanism for the goods allocation problem over normalized instances $\mathcal{N}$ that attains the optimal egalitarian welfare.*

Although this work primarily focuses on proportional mechanisms, we explore another important direction by considering relaxations of envy-freeness. We provide a characterization of allocation rules that can be made *approximately* envy-free with payments. While we leave open the question of whether a constant-factor approximation to the makespan can be achieved with an α-envy-free mechanism for some constant $\alpha < 1$, we make some progress by showing that the technique used in (Cohen et al. 2010) to establish the impossibility result for envy-freeness does not extend to α-envy-freeness for any $\alpha < 1$.

1.2 Our Techniques

The first technical contribution of this work is a characterization of all allocation rules that can be paired with a payment function to form a proportional mechanism. Our characterization is based on a new concept of mean efficiency, which requires that the total cost $\sum_{i\in[m]} c_i(A_i)$ of an allocation A does not exceed the average cost in the cost matrix multiplied by n, i.e., $\sum_{i\in[m]} c_i(A_i) \leq (1/m)\cdot \sum_{i\in[m]}\sum_{j\in[n]} c_{i,j}$. This condition naturally introduces a new efficiency criterion for job scheduling. Notably, our characterization reveals a perhaps surprising property that the fairness notion of proportionality for the job scheduling problem is equivalent to a notion of efficiency.

We also provide a simple formula to compute payments that ensure any mean-efficient allocation rule A is proportional. Specifically, each machine is compensated for the cost it incurs minus its proportional share of the total cost of all jobs. Formally, we set $p_i = c_i(A_i) - (1/m)\cdot c_i([n])$.

Our main technical contribution is the Anti-Diagonal Mechanism (Algorithm 1) which is proportional and provides a $3/2$-approximation to the optimal makespan for any general instance. Here, we briefly outline the intuition behind this mechanism. We begin with an initial allocation

	Property	Upper Bound	Lower Bound
General instances	Proportionality	$3/2$ (Theorem 1)	$3/2$ (Theorem 2)
	Envy-freeness	$O(\log m)$ (Cohen et al. 2010)	$\Omega(\log m/\log\log m)$ (Cohen et al. 2010)
Normalized instances	Proportionality	1 (Theorem 3)	1
	Envy-freeness	$O(\log m)$ (Cohen et al. 2010)	$\Omega(\log m/\log\log m)$ (Lemma 4.2)

Table 1: Bounds for makespan approximation in job scheduling under two properties—proportionality and envy-freeness—examined across two settings: general costs and normalized costs.

B that minimizes the makespan across all possible job-to-machine assignments, given as input to the mechanism. The key challenge is that this allocation may not be proportional. The mechanism's objective is to adjust the allocation to ensure proportionality without increasing the makespan by more than a factor of $3/2$. For the sake of simplifying the high-level argument, let us assume that the number of jobs equals the number of machines and that each machine i is assigned job i. While these assumptions are *not* explicitly used in our proof, they can be made without loss of generality. This is achieved by merging all jobs in each bundle A_i into a single meta-job, a process that does not create any new proportionable allocations, and then relabeling the jobs accordingly.

In the first step, the algorithm evaluates all m anti-diagonals in the m-by-m cost matrix c and selects the one that minimizes the total cost. Given that the average cost of all anti-diagonals equals the average cost of the entire matrix c, the allocation corresponding to the cost-minimizing anti-diagonal must be mean-efficient. The remaining challenge is that this selected anti-diagonal allocation might result in a high makespan. Our goal is to reduce the makespan while preserving mean-efficiency. We demonstrate that this can be achieved through a series of carefully executed merge or swap operations on pairs of antipodal machines along the anti-diagonal.

We employ a different mechanism for normalized instances. Specifically, we always select the makespan-minimizing allocation A with the minimum cost. We prove that this allocation is mean-efficient. The core idea of the proof is to construct a graph where each node represents a machine. We add an edge from machine i to machine j if j is more efficient at processing the jobs allocated to i, i.e., $c_j(A_i) < c_i(A_i)$. This leads to a key observation: there exists a machine j that is weakly less efficient than all others for their respective bundles of jobs, meaning $c_i(A_i) \leq c_j(A_i)$ for every machine i. The result then naturally follows from this observation.

1.3 Additional Related Works

There is a large body of related literature on the job scheduling problem, the fair division problem with payments, the fair division problem for chores, and the price of fairness. Due to length restrictions, we include a detailed discussion of the literature on each of these topics, and the full proofs of our theorems, in the extended version of this paper.

2 Preliminaries

Throughout this work, we denote $[n] = \{1, 2, \ldots, n\}$.

2.1 Job Scheduling

We denote the set of machines by $[m]$ and the set of jobs by $[n]$. We assume there are at least two machines and at least two jobs, i.e., $m \geq 2$ and $n \geq 2$. An instance of the job scheduling problem is represented by an m-by-n matrix c: For each machine $i \in [m]$, there is an associated cost $c_{i,j} \in \mathbb{R}_{\geq 0}$ for every job $j \in [n]$. The costs are assumed to be additive: If a set of jobs $S \subseteq [n]$ is allocated to machine i, then machine i incurs a total cost equal to $c_i(S) = \sum_{j \in S} c_{i,j}$.

We differentiate between instances with normalized cost functions, where the total cost for each machine to handle all jobs is scaled to a common value, and instances with general cost functions, where assigning jobs to machines involves arbitrary non-negative costs.

Definition 2.1 (General and normalized instances)**.** *We consider the following cases:*

- *The set of* general *instances is denoted by* $\mathcal{C} = \mathbb{R}_{\geq 0}^{m \times n}$.
- *The set of* normalized *instances is denoted by* $\mathcal{N} = \{c \in \mathcal{C} : c_i([n]) = C \text{ for all } i \in [m] \text{ and some } C\}$*, where* C *is the normalization factor.*

We state the definition of normalized instances in terms of the normalization factor C rather than fixing it to 1, as this allows some of our analysis to extend to cases where the costs and the normalization factor are negative, as explained in Section 5.

The goal of the job scheduling problem is to assign jobs to machines and determine appropriate payments for them. This is captured by the following notion of a mechanism.

Definition 2.2 (Mechanisms)**.** *A mechanism* $M = (A, p)$ *defined over a set of instances* $\mathcal{I}$ *consists of an allocation function* A *and a payment function* p*. For each instance* $c \in \mathcal{I}$*, the allocation function determines the allocation* $A(c) = (A(c)_1, \ldots, A(c)_m)$*, where* $A(c)_i$ *is the set of jobs allocated to machine* i *and each job must be allocated to exactly one machine. The payment function determines the payments* $p(c) = (p(c)_1, \ldots, p(c)_m)$*, where* $p(c)_i \in \mathbb{R}$ *is the payment for machine* i*.*

For simplicity, we use A_i and p_i to denote $A(c)_i$ and $p(c)_i$, respectively, when c is clear from the context.

The primary objective in the job scheduling problem is to minimize the makespan, defined as the maximum cost of any machine. In this work, we aim to develop mechanisms that achieve approximately optimal makespan, as defined below.

Definition 2.3 (Makespan approximation)**.** *An allocation rule* A *defined over a set of instances* $\mathcal{I}$ *provides an* β*-approximation to the optimal makespan for some* $\beta \in [1, \infty)$

if, for every instance $c \in \mathcal{I}$, it holds that $\max_{i\in[m]} c_i(A_i) \leq \beta \cdot \mathrm{OPT}(c)$*, where* $\mathrm{OPT}(c)$ *is defined as* $\mathrm{OPT}(c) = \min_X \max_{i\in[m]} c_i(X_i)$ *and* X *ranges over all possible allocations of jobs to machines.*

2.2 Fairness Notions

In this section, we define two relevant fairness notions for mechanisms in the job scheduling problem: proportionality and envy-freeness. We assume that machines have *quasi-linear disutilities*, meaning the disutility for machine i in instance c under mechanism (A, p) is given by $c_i(A(c)_i) - p(c)_i$. First, we define proportionality.

Definition 2.4 (Proportional mechanism)**.** *A mechanism (A, p) over a set of instances $\mathcal{I}$ is* proportional *if, for every instance $c \in \mathcal{I}$ and every machine $i \in [m]$, we have $c_i(A_i) - p_i \leq (1/m) \cdot \sum_{j\in[m]}(c_i(A_j) - p_j)$.*

To simplify our analysis, we focus on allocation functions for which appropriate payments can be defined, initially ignoring the value of the payments. To capture this idea, we introduce the following definition.

Definition 2.5 (Proportionable allocation)**.** *An allocation function A is* proportionable *if it can be made proportional with some payments, i.e., there exists a payment function p such that the mechanism (A, p) is proportional.*

Next, we introduce the second fairness notion relevant to this work, namely, envy-freeness.

Definition 2.6 (Envy-free mechanism)**.** *A mechanism (A, p) over a set of instances $\mathcal{I}$ is* envy-free *if for every instance $c \in \mathcal{I}$ and for every pair $i, j \in [m]$ of machines, we have $c_i(A_i) - p_i \leq c_i(A_j) - p_j$.*

Similarly to the notion of proportionable allocations, we have the following notion of envy-freeable allocations.

Definition 2.7 (Envy-freeable allocation)**.** *An allocation function A is* envy-freeable *if it can be made envy-free with some payments, i.e., there exists payment function p such that the mechanism (A, p) is envy-free.*

3 Characterizations of Fair Allocations

An important contribution of this work is a new characterization for proportionable allocation functions through the concept of mean efficiency, which we introduce here.

Definition 3.1 (Mean efficiency)**.** *An allocation function A defined over a set of instances $\mathcal{I}$ is* mean-efficient *if for every instance $c \in \mathcal{I}$ and every machine $i \in [m]$, it holds that $\sum_{i\in[m]} c_i(A_i) \leq (1/m) \cdot \sum_{i\in[m]} c_i([n])$.*

The following theorem provides a formal characterization of proportionable allocation functions.

Theorem 6. *An allocation function A for the job scheduling problem over any set of instances $\mathcal{I}$ is proportionable if and only if it is mean-efficient.*

Proof. Fix an instance c. Let A be any proportionable allocation function and let p be a payment function so that (A, p) is proportional. Since the mechanism is proportional, for every agent i, we have $c_i(A_i) - p_i \leq (1/m) \cdot \sum_{j\in[m]} (c_i(A_j) - p_j)$. Summing these inequalities over all $i \in [m]$, we get

$$\begin{aligned}\sum_{i\in[m]} c_i(A_i) - \sum_{i\in[m]} p_i &\leq (1/m) \cdot \sum_{i\in[m]} \sum_{j\in[m]} (c_i(A_j) - p_j) \\ &= (1/m) \cdot \sum_{i\in[m]} c_i([n]) - \sum_{i\in[m]} p_i\end{aligned}$$

which implies $\sum_{i\in[m]} c_i(A_i) \leq (1/m) \cdot \sum_{i\in[m]} c_i([n])$. Thus, any proportionable allocation must be mean-efficient.

For the other direction, let A be any mean-efficient allocation function. Consider the payment function p given by $p_i = c_i(A_i) - (1/m) \cdot c_i([n])$ for every machine $i \in [m]$. Observe that by the definition of mean efficiency,

$$\begin{aligned}\sum_{i\in[m]} p_i &= \sum_{i\in[m]} (c_i(A_i) - (1/m) \cdot c_i([n])) \\ &= \sum_{i\in[m]} c_i(A_i) - (1/m) \cdot \sum_{i\in[m]} c_i([n]) \leq 0.\end{aligned}$$

It follows that for every agent i, we have $c_i(A_i) - p_i = (1/m) \cdot c_i([n]) = (1/m) \cdot \sum_{j\in[m]} c_i(A_j) \leq (1/m) \cdot \sum_{j\in[m]} (c_i(A_j) - p_j)$. Since this condition holds for any instance c, we get that the mechanism (A, p) is proportional. Hence, any mean-efficient allocation function is proportionable, which concludes the proof. □

A similar characterization for the set of allocations that can be made envy-free was previously known in the literature (Aragones (1995); Hartline et al. (2008); Cohen et al. (2010)), and we present the earlier characterization here for the sake of comparison. This characterization involves the concept of local efficiency, defined below.

Definition 3.2 (Local efficiency)**.** *An allocation function A defined over a set of instances $\mathcal{I}$ is* locally efficient *if for every instance $c \in \mathcal{I}$, there is no permutation of the (fixed) bundles of jobs among the agents that decreases the social cost, i.e., for every permutation $\pi : [m] \to [m]$, it holds that $\sum_{i\in[m]} c_i(A_i) \leq \sum_{i\in[m]} c_i(A_{\pi(i)})$.*

We restate the characterization of Hartline et al. (2008) below, which demonstrates that an allocation function is envy-freeable if and only if it is locally efficient.

Theorem 7 (Hartline et al. (2008))**.** *An allocation function A for the job scheduling problem over any set of instances $\mathcal{I}$ is envy-freeable if and only if it is locally efficient.*

In the extended version, we also consider approximate envy-freeness. In particular, we extend the above definition to the notion of *approximate* local efficiency, and present an extended characterization for approximately envy-free allocations using this notion.

4 Proportional Mechanisms

In this section, we provide our main results for proportional mechanisms.

4.1 General Instances

We first consider the case of general instances (see Definition 2.1). Our first result is the following upper bound.

Theorem 1. *There is a proportional mechanism for the job scheduling problem over general instances $\mathcal{C}$ that gives a $3/2$-approximation to the optimal makespan.*

Proof. We analyze the mechanism described in Algorithm 1. Figure 1 shows an illustration of the operations performed by the algorithm.

We begin by showing that the allocation A is proportionable, regardless of the initial allocation B provided as input to Algorithm 1. Fix any general instance $c \in \mathcal{C}$ and let k be defined as in Line 2. After initializing A in Line 3, we have:

$$\begin{aligned}\sum_{i\in[m]} c_i(A_i) &= \sum_{i\in[m]} c_i(B_{m-i+k}) \\ &\leq (1/m)\cdot \sum_{i\in[m]}\sum_{\ell\in[m]} c_i(B_{m-i+\ell}) \\ &= (1/m)\cdot \sum_{i\in[m]} c_i([n]).\end{aligned}$$

This shows that allocation A is mean-efficient immediately after Line 3. Additionally, note that the sum $\sum_{i\in[m]} c_i(A_i)$ decreases whenever the swap operation in Line 7 or the merge operation in Line 10 is executed. This is because when the swap operation is executed, the inequality $c_i(A_j) + c_j(A_i) \leq c_i(A_i) + c_j(A_j)$ holds, and when the merge operaetion is executed, the condition $c_i(A_j) \leq c_j(A_j)$ is satisfied. Therefore, allocation A remains mean-efficient throughout the execution of the algorithm, and so the mechanism is proportional since payments in Line 13 are exactly the payments given in the proof of Theorem 6.

Next, we prove that for any general instance $c \in \mathcal{C}$, the returned allocation satisfies $\max_{\ell\in[m]} c_\ell(A_\ell) \leq (3/2)\cdot M$, where $M = \max_{\ell\in[m]} c_\ell(B_\ell)$. Consider any $\ell \in [m]$ and let $h = m - \ell + k$. Note that A_ℓ can only be one of $\{\emptyset, B_\ell, B_h, B_\ell \cup B_h\}$. This is because the only operations that modify A_ℓ in Line 7 or Line 10 occur in the for-loop iteration with $i = \ell$ or $j = \ell$. The only for-loop iteration where $j = \ell$ must be when $i = h$, since $m - h + k = m - (m - \ell + k) + k = \ell$.

Clearly, we have $c_\ell(\emptyset) = 0 \leq (3/2)\cdot M$. By the definition of M, it follows that $c_\ell(B_\ell) \leq \max_{w\in[m]} c_w(B_w) \leq (3/2)\cdot M$. The only way for A_ℓ to be set to $B_\ell \cup B_h$ is in the merge operation in Line 10 during the iteration with $i = \ell$. In this case, the if-condition ensures that $c_\ell(B_\ell \cup B_h) = c_i(A_i) + c_i(A_j) \leq (3/2)\cdot M$.

Finally, consider the scenario where $A_\ell = B_h$ and $A_h = B_\ell$ with the assumption that $c_\ell(A_\ell) > (3/2)\cdot M$. Initially, A_ℓ is set to B_h. If $A_\ell = B_h$ at the end of the algorithm, then A_ℓ and A_h were neither swapped by Line 7 nor merged by Line 10 in any for-loop iteration with $i = \ell$ or $i = h$. Since Line 7 was not executed in the iteration with $i = \ell$, the if-condition implies $c_\ell(B_h) + c_h(B_\ell) \leq c_\ell(B_\ell) + c_h(B_h)$. Given $c_\ell(B_h) > (3/2)\cdot M$ by assumption, along with $c_\ell(B_\ell) \leq M$ and $c_h(B_h) \leq M$, we get

$$\begin{aligned}c_h(B_\ell) &\leq c_\ell(B_\ell) + c_h(B_h) - c_\ell(B_h) \\ &\leq 2\cdot M - (3/2)\cdot M = (1/2)\cdot M.\end{aligned}$$

Moreover, since $c_\ell(B_h) > (3/2)\cdot M > c_h(B_h)$,

$$c_h(B_\ell) \leq c_\ell(B_\ell) + c_h(B_h) - c_\ell(B_h) \leq c_\ell(B_\ell).$$

However, this leads to a contradiction, as it implies the if-condition for the merge operation in Line 10 was met during the iteration with $i = h$, contradicting the assumption that $A_\ell = B_h$. Therefore, we conclude that $c_\ell(A_\ell) \leq (3/2)\cdot M$ for all $\ell \in [m]$ at the end of the algorithm's execution.

The theorem follows by selecting for B, the input to Algorithm 1, an optimal (makespan-minimizing) allocation. We then have $M = \mathrm{OPT}(c)$, which establishes the result. □

Remark 4.1. *The running time of Algorithm 1 is polynomial in n and m. That is, given any allocation B as input, the algorithm returns a proportional allocation that gives a $3/2$-approximation with respect to the makespan of B, in polynomial time. By taking B to be the optimal (i.e., makespan-minimizing) allocation, this implies the existence of a $3/2$-approximation stated in Theorem 1. However, it is well-known that finding an optimal allocation is NP-hard (Lenstra, Shmoys, and Tardos 1990). Despite this computational challenge, the black-box nature of the analysis of Algorithm 1 implies that, combined with the polynomial-time 2-approximation algorithm by Lenstra, Shmoys, and Tardos (1990), one can obtain a proportional mechanism that gives a 3-approximation to the optimal makespan in polytime.*

We also provide a matching lower bound, showing that the result of Theorem 1 is tight.

Theorem 2. *For every case where $n \geq m$, there is no proportional mechanism for the job scheduling problem over general instances $\mathcal{C}$ that gives a $(3/2-\epsilon)$-approximation to the optimal makespan for any $\epsilon > 0$.*

Proof. Suppose, for the sake of contradiction, that there exists a mechanism (A, p) that is proportional and provides a $(3/2-\epsilon)$-approximation to the optimal makespan. Consider an instance c defined by the following conditions: $c_{i,i} = 1$ for all $1 \leq i \leq m$, $c_{i,j} = 1/2$ for all $1 \leq j < i \leq m$, $c_{i,j} = 3/2 - \epsilon/2$ for all $1 \leq i < j \leq m$, and $c_{i,j} = 0$ for all $1 \leq i \leq m$ and $m < j \leq n$. For the special case of $m = n = 2$, we get the following cost matrix:

$$\begin{pmatrix} 1 & 3/2-\epsilon/2 \\ 1/2 & 1 \end{pmatrix}$$

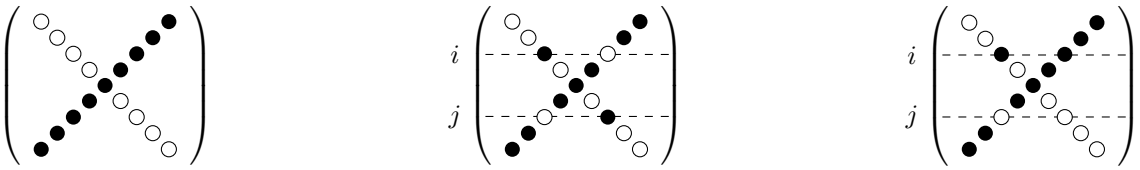
Note that the optimal makespan is $\mathrm{OPT}(c) = 1$ because each job $j \in [m]$ can be allocated to machine j, and jobs $j > m$ can be assigned to any machine without affecting the makespan. Therefore, since (A, p) achieves a $(3/2-\epsilon)$-approximation, $A(c)$ must satisfy $c_i(A_i) \leq 3/2 - \epsilon$ for all machines $i \in [m]$.

We will show by induction on k that every job $j \in \{k, \ldots, m\}$ must be allocated to machine j. For the base case, consider job m. We have $c_{m,m} = 1$ and $c_{i,m} =$

Algorithm 1: Anti-Diagonal Mechanism

Input: c is an m-by-n cost matrix and B is any initial allocation.
Output: (A, p) is a proportional mechanism with $\max_{\ell \in [m]} c_\ell(A_\ell) \leq (3/2) \cdot \max_{\ell \in [m]} c_\ell(B_\ell)$.

1 **Procedure** `AntiDiagonalMechanism()`**:**
  `// Indices are taken modulo` m `so that` $B_{m+1} = B_1$ `and so on.`
2 $k \leftarrow$ any $k \in [m]$ that minimizes $\sum_{i \in [m]} c_i(B_{m-i+k})$
3 initialize allocation $(A_1, \ldots, A_m)$ so that $A_i \leftarrow B_{m-i+k}$ for all $i \in [m]$
4 **for** $i \in [m]$ **do**
5 $j \leftarrow m - i + k$
6 **if** $c_i(A_j) + c_j(A_i) < c_i(A_i) + c_j(A_j)$ **then**
7 swap bundles A_i and A_j
8 **end**
9 **if** $c_i(A_j) < c_j(A_j)$ and $c_i(A_i) + c_i(A_j) \leq (3/2) \cdot \max_{\ell \in [m]} c_\ell(B_\ell)$ **then**
10 merge bundles A_i and A_j by letting $A_i \leftarrow A_i \cup A_j$ and $A_j \leftarrow \emptyset$
11 **end**
12 **end**
13 set payments $(p_1, \ldots, p_m)$ so that $p_i \leftarrow c_i(A_i) - (1/m) \cdot c_i([n])$ for all $i \in [m]$
14 **return** (A, p)



(a) Anti-diagonal allocation (Line 3) (b) Swap operation (Line 7) (c) Merge operation (Line 10)

Figure 1: Illustration of the Anti-Diagonal Mechanism for the case where $m = n$, with the initial allocation $B_i = \{i\}$ and the parameter k set to 1. Rows represent machines and columns represent jobs. Empty dots indicate the initial allocation B. The subfigures show the allocation matrix after the swap and merge operations, where full dots represent the resulting allocation.

$3/2 - \epsilon/2 > 3/2 - \epsilon$ for all $i < m$. Therefore, job m must be allocated to machine m, since any other assignment results in a makespan greater than $3/2 - \epsilon$. Now, suppose that for some $1 \leq k < m$, every job $j \in \{k+1, \ldots, m\}$ is allocated to machine j. We will show that job k must also be allocated to machine k. Let i be the machine that job k is allocated to. First, observe that we cannot have $1 \leq i < k$ because $c_{i,k} = 3/2 - \epsilon/2 > 3/2 - \epsilon$ for all $i < k$, and such an assignment would result in a makespan exceeding $3/2 - \epsilon$. Second, we cannot have $k < i \leq m$, since for these machines, $c_{i,k} = 1/2$, and by the inductive assumption, each such machine i already has job i allocated to it, so assigning job k to any such machine results in a total cost for machine i of $c_i(A_i) \geq 1 + 1/2 = 3/2 > 3/2 - \epsilon$. Consequently, job k must be allocated to machine k. By induction, this argument holds for every job $j \in \{1, \ldots, m\}$. Therefore we have $\sum_{i \in [m]} c_i(A_i) = m$.

Since the mechanism is proportional, its allocation function must be proportionable. According to Theorem 6, this implies that for our instance, $\sum_{i \in [m]} c_i(A_i) \leq (1/m) \cdot \sum_{i \in [m]} \sum_{j \in [n]} c_{i,j}$. Let us calculate the sum $\sum_{i \in [m]} \sum_{j \in [n]} c_{i,j}$ as follows: The number of terms with value 1 is m, the number of terms with value $1/2$ is $m(m-1)/2$, and the number of terms with value $3/2 - \epsilon/2$ is $m(m-1)/2$. These terms sum up to $m^2 - (m(m-1)/2) \cdot (\epsilon/2)$. It follows that $\sum_{i \in [m]} c_i(A_i) \leq m - ((m-1)/2) \cdot (\epsilon/2) < m$, which is in contradiction with our previous observation that $\sum_{i \in [m]} c_i(A_i) = m$. □

4.2 Normalized Instances

Next, we consider the case of normalized instances (see Definition 2.1). Our first result is the following upper bound, the proof of which is given in the extended version.

Theorem 3. *There is a proportional mechanism for the job scheduling problem over normalized instances $\mathcal{N}$ that attains the optimal makespan.*

The above result indicates that for proportional mechanisms, makespan approximation ratio can be improved from $3/2$ (Theorem 2) to 1 (Theorem 3) when the cost functions are normalized. However, this improvement does not extend to envy-free mechanisms. We demonstrate that the lower bound for envy-free mechanisms, $\Omega(\log m / \log \log m)$, as established by Cohen et al. (2010), remains valid even for normalized instances. This conclusion stems directly from the following reduction, presented using the more general concept of α-envy-freeness, which corresponds exactly to envy-freeness when $\alpha = 1$.

Lemma 4.2. *Fix $\alpha \in (0, 1]$ and $\beta \in [1, \infty)$. Suppose that there is an α-envy-free mechanism for the job scheduling problem over normalized instances $\mathcal{N}$ with $m + 1$ machines and $n + 1$ jobs that gives a β-approximation to the optimal makespan. Then, there is an α-envy-free mechanism for the job scheduling problem over general instances $\mathcal{C}$ with m machines and n jobs that gives a β-approximation to the optimal makespan.*

5 The Goods Allocation Problem

In this section, we consider the dual problem, where instead of allocating costly jobs, we allocate valuable goods. As is standard in the fair division literature, throughout this section, we refer to machines as agents.

An instance of the goods allocation problem is represented by an m-by-n matrix v, which specifies the value $v_{i,j}$ that agent $i \in [m]$ derives from being allocated a good $j \in [n]$. We will consider general instances, $v \in \mathcal{C}$, and normalized instances, $v \in \mathcal{N}$ (see Definition 2.1). Notably, an instance of the goods allocation problem v corresponds to an instance of the job scheduling problem, where $c = -v$.

The goal of the goods allocation problem is to allocate goods to agents and determine the transfers they must make to the mechanism. We capture this solution using a mechanism (A, q), consisting of an allocation function A and a transfer function q, as defined in Definition 2.2. The utility of agent i for instance v is given by $v_i(A_i) - q_i = -(c_i(A_i) - p_i)$, where $c = -v$ and $p = -q$. Using this correspondence between (negative) instances of the job scheduling problem and instances of the goods allocation problem, the definition of proportionality in Definition 2.4 becomes the following definition.

Definition 5.1 (Proportionality)**.** *A mechanism (A, q) defined over a set of instances $\mathcal{I}$ is* proportional *if, for every instance $v \in \mathcal{I}$ and every agent $i \in [m]$, we have $v_i(A_i) - q_i \geq (1/m) \cdot \sum_{j \in [m]} (v_i(A_j) - q_j)$.*

Similarly, the definition of mean efficiency (Definition 3.1) is adapted as follows.

Definition 5.2 (Mean efficiency)**.** *An allocation function A defined over a set of instances $\mathcal{I}$ is* mean-efficient *if for every instance $v \in \mathcal{I}$ and every agent $i \in [m]$, it holds that $\sum_{i \in [m]} v_i(A_i) \geq (1/m) \cdot \sum_{i \in [m]} v_i([n])$.*

The equivalence between proportionability (Definition 2.5) and mean efficiency for the goods allocation problem can be established through the same reasoning as used in the job scheduling problem. The proof of Theorem 6 (Section 3) remains valid even when the costs are negative. Consequently, due to the correspondence between instances of the goods allocation problem and the job scheduling problem discussed earlier, we get the following corollary.

Corollary 5.3. *An allocation function A for the goods allocation problem over any set of instances $\mathcal{I}$ is proportionable if and only if it is mean-efficient.*

The corresponding objective in the goods allocation problem is to maximize the egalitarian welfare, defined as the minimum value derived by any agent. In this work, we aim to develop mechanisms that achieve approximately optimal egalitarian welfare, as defined below.

Definition 5.4 (Egalitarian welfare approximation)**.** *We say that a mechanism $M = (A, p)$ defined over a set of instances $\mathcal{I}$ provides an α-approximation to the optimal egalitarian welfare for some $\alpha \in [1, \infty)$ if, for every instance $v \in \mathcal{I}$, it holds that $\alpha \cdot \min_{i \in [m]} v_i(A_i) \geq \mathrm{OPT}(v)$, where $\mathrm{OPT}(v)$ is defined as $\mathrm{OPT}(v) = \max_X \min_{i \in [m]} v_i(X_i)$ and X ranges over all possible allocations of goods to agents.*

First, we consider the case of general instances.

Theorem 4. *For every case where $n \geq m$, there is no proportional mechanism for the goods allocation problem over general instances $\mathcal{C}$ that gives a β-approximation to the optimal egalitarian welfare for any $\beta > 0$.*

Next, we address the case of normalized instances. Notably, the proof of Theorem 3 is applicable to any normalized instance of the job scheduling problem, regardless of whether the costs or the normalization factor are negative. Consequently, we directly obtain the following theorem for normalized instances of the goods allocation problem.

Theorem 5. *There is a proportional mechanism for the goods allocation problem over normalized instances $\mathcal{N}$ that attains the optimal egalitarian welfare.*

6 Conclusion

Our results significantly advance our understanding of the interplay between the job scheduling problem and fair division. Our main result demonstrates a job scheduling mechanism that achieves a tight approximation factor of $3/2$ for the optimal makespan, while guaranteeing an allocation that is proportionally fair. We also present several results for normalized instances and for fair division of goods. Our work also raises important questions about the price of fairness for the makespan objective for other fairness notions. For example, does there exist a mechanism that achieves approximate envy-freeness while guaranteeing a constant factor approximation to the optimal makespan? We provide some insights into this question in the extended version. Similarly, does there exist a mechanism that satisfies other common relaxations of envy-freeness, such as EF1 (Budish 2011) or EFX (Caragiannis et al. 2019), and achieves a constant factor approximation to the optimal makespan?

Acknowledgments

This project has been partially funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No. 866132), by an Amazon Research Award, by the NSF-BSF (grant number 2020788), by the Israel Science Foundation Breakthrough Program (grant No.2600/24), and by a grant from TAU Center for AI and Data Science (TAD). Jugal Garg was supported by NSF Grants CCF-1942321 and CCF-2334461.

We also thank Amos Fiat, Aleksander Łukasiewicz, Franciszek Malinka, Simon Mauras, and Divyarthi Mohan for invaluable discussions.

References

Amanatidis, G.; Aziz, H.; Birmpas, G.; Filos-Ratsikas, A.; Li, B.; Moulin, H.; Voudouris, A. A.; and Wu, X. 2023. Fair division of indivisible goods: Recent progress and open questions. *Artificial Intelligence*, 322: 103965.

Aragones, E. 1995. A derivation of the money Rawlsian solution. *Social Choice and Welfare*, 12(3): 267–276.

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

Caragiannis, I.; Kurokawa, D.; Moulin, H.; Procaccia, A. D.; Shah, N.; and Wang, J. 2019. The unreasonable fairness of maximum Nash welfare. *ACM Transactions on Economics and Computation (TEAC)*, 7(3): 1–32.

Christodoulou, G.; Koutsoupias, E.; and Kovács, A. 2023. A proof of the Nisan-Ronen conjecture. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, 672–685.

Cohen, E.; Feldman, M.; Fiat, A.; Kaplan, H.; and Olonetsky, S. 2010. Envy-free makespan approximation. In *Proceedings of the 11th ACM conference on Electronic commerce (EC)*, 159–166.

Fiat, A.; and Levavi, A. 2012. Tight Lower Bounds on Envy-Free Makespan Approximation. In *WINE*, volume 7695 of *Lecture Notes in Computer Science*, 553–558. Springer.

Foley, D. 1967. Resource allocation and the public sector. *Yale Econ Essays*, 7(1): 45–98.

Hartline, J.; Ieong, S.; Mualem, A.; Schapira, M.; and Zohar, A. 2008. Multi-dimensional Envy-free Scheduling Mechanisms. Technical Report 1144, The Hebrew University.

Lenstra, J. K.; Shmoys, D. B.; and Tardos, É. 1990. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical programming*, 46: 259–271.

Liu, S.; Lu, X.; Suzuki, M.; and Walsh, T. 2024. Mixed fair division: A survey. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 22641–22649.

Nisan, N.; and Ronen, A. 1999. Algorithmic mechanism design. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, 129–140.

Steinhaus, H. 1948. The problem of fair division. *Econometrica*, 16: 101–104.