Envy-Free Makespan Approximation

Edith Cohen[*] Michal Feldman[†] Amos Fiat[‡] Haim Kaplan [§] Svetlana Olonetsky[¶]

Abstract

We study envy-free mechanisms for assigning tasks to agents, where every task may take a different amount of time to perform by each agent, and the goal is to get *all* the tasks done as soon as possible (i.e., minimize the makespan). For indivisible tasks, we put forward an envy-free polynomial mechanism that approximates the minimal makespan to within a factor of $O(\log m)$, where m is the number of machines. This bound is almost tight, as we also show that no envy-free mechanism can achieve a better bound than $\Omega(\log m/\log\log m)$. This improves the recent result of Mu'alem [24] who introduced the model and gave an upper bound of $(m+1)/2$, and a lower bound of $2-1/m$. For divisible tasks, we show that there always exists an envy-free poly-time mechanism with optimal makespan. Finally, we demonstrate how our mechanism for envy free makespan minimization can be interpreted as a market clearing problem.

[*]AT&T Labs-Research, 180 Park Avenue, Florham Park, NJ.
[†]School of Business Administration and Center for the Study of Rationality, The Hebrew University of Jerusalem.
[‡]The Blavatnik School of Computer Science, Tel Aviv University.
[§]The Blavatnik School of Computer Science, Tel Aviv University.
[¶]The Blavatnik School of Computer Science, Tel Aviv University.

1 Introduction

Imagine a set of n household chores, and m kids in the family. Every chore may take a different amount of time to perform by each child. A single chore cannot be performed by more than one child, but multiple chores can be assigned to a single child. The parents want to allocate chores *fairly*, and may offer inducements to the children so as to ensure fairness. The parents have an additional goal which is to get all the chores out of the way as soon as possible. This problem is our main focus. In job scheduling terminology, we study mechanisms for the fair allocation of tasks to machines (which correspond to the agents), each of which may take a different length of time to complete every task, subject to the added goal of minimizing the makespan; *i.e.*, getting all tasks done as soon as possible. This problem has been first introduced by Mu'alem [24].

The problem of fair division, often modeled as that of partitioning a cake fairly, goes back at least to 1947 and is attributed to Jerzy Neyman, Hugo Steinhaus, Stefan Banach and Bronislaw Knaster ([29, 30]). There are several books on fair division, and hundreds of references, both mathematical and philosophical ([31, 4, 23, 17, 27]). Gardner (1978, [12]) is credited with asking about fair division of household chores. Experiments suggest that human subjects prefer degraded performance over discrimination; for example, they prefer longer waiting times in queues over being treated unfairly [28]. Thus, fairness is a crucial property of any mechanism.

In order to devise a fair division, one should first define the precise notion of fairness desired. One common notion of fairness is that of "envy-freeness", an allocation where no one seeks to switch her outcome with that of another (Dubins and Spanier, 1961, [10], Foley, 1967, [11]). The envy-freeness notion is a natural extension of a Walrasian equilibrium. Whereas Walrasian equilibrium seeks posted *item* pricing that clears the market, envy-free mechanisms use market-clearing pricing, but where prices are assigned to *bundles* of items rather than to individual items.

In a task assignment (or job scheduling) setting, tasks may be divisible or indivisible. With divisible tasks, it is always possible to divide every task equally between all agents. This is obviously envy-free, but infinite task times (e.g., a task too demanding for a four year old) may make this assignment impossible or ill-defined. For indivisible tasks, it is less obvious that one can achieve envy-freeness. However, we can always achieve envy-freeness if we allow the *mechanism* to determine both an allocation and payments (to or from the agents, to the mechanism or between themselves).

When payments are used, we assume that an agent's utility is quasi-linear, *i.e.*, it is equal to the payment that the agent gets from the mechanism minus the sum of the costs of tasks assigned to it by the mechanism. In particular, one can easily check that if we assign each task j to the machine that has the smallest cost for running j (i.e. we maximize the social welfare), and use VCG payments then we get an assignment which is envy-free.

In fact, when payments are allowed, even a Walrasian equilibrium is known to exists. A Walrasian equilibrium is a set of payments, one for each item, such that each agent receives a bundle that maximizes her utility (which is the difference between her payment and the sum of the costs of the bundle's items). By definition every Walrasian equilibrium corresponds to an envy-free mechanism. It is well known that every setting with *gross substitutes* valuations[1] admits a Walrasian equilibrium [14].

Within the range of possible envy-free allocations, one may want to compute an envy-free allocation that achieves additional goals, such as economic efficiency, revenue maximization and incentive compatibil-

[1]Let U be the set of items. For an item-payment vector p, let $D(p) = argmax_{S\subseteq U} \sum_{j\in S} p_j - c(S)$. A cost function $c : 2^U \to R$ satisfies the gross substitutes condition if for any two item-payment vectors p and q such that $q \geq p$, and any $A \in D(q)$, there exists $B \in D(p)$ such that $\{a \in A \mid p_a = q_a\} \subset B$. In paricular if $c(S)$ is the sum of the costs of the items in S then it satisfies the gross substitutes condition.

ity.[2] Mu'alem [24] studied the additional goal of makespan minimization. In particular, they seek envy-free mechanisms for scheduling indivisible tasks on unrelated machines (i.e., where every task may take a different amount of time to perform by each machine) that approximately minimize the makespan. Consider an instance of indivisible task scheduling for m agents (without envy-free requirements), and without loss of generality assume that the assignment of minimal makespan has makespan 1. Mu'alem show that there is no envy-free mechanism that guarantees a makespan less than $2 - 1/m$. They also give an algorithm that always produces a schedule with makespan at most $(m + 1)/2$. More recently, Kempe *et al.* [18] studied the problem of envy-free allocations for bidders with budget constraints.

Nisan and Ronen [26] considered the above setting of job scheduling on unrelated machines in a game theoretic context, where agents are machines that want to minimize their utility. Nisan and Ronen were not concerned with the fairness of the allocation, rather they looked for an incentive compatible mechanism that approximates the minimal makespan. The problem posed by Nisan and Ronen has led to a great deal of work [21, 8, 19, 1], yet the main question is still open. For the general case, all that is known is a lower bound of 2.61 and an upper bound of m (similar to the gap obtained by Mu'alem for envy-free mechanisms)[3]. For divisible tasks, Christodoulou *et al.* [7] demonstrated an upper bound of $1 + (m - 1)/2$ and a lower bound of $2 - 1/m$ (while for the class of "task independent" algorithms, the bound of $1 + (m - 1)/2$ holds as a lower bound as well).

Our contributions: In Section 3 we give an envy-free mechanism for scheduling indivisible tasks on m unrelated machines, that approximates the minimal makespan to within a factor of $O(\log m)$, improving the $(m + 1)/2$ bound of Mu'alem [24]. Our mechanism runs in polynomial time.

In Section 4 we give a lower bound of $\Omega(\log m/ \log\log m)$ on the makespan approximation ratio of any envy-free scheduling mechanism for indivisible tasks (polynomial time, or not). This improves on the previous $2 - 1/m$ of Mu'alem [24].

In Section 5 we consider the problem of machine scheduling with *divisible* tasks. We show that, in contrast to the case of indivisible tasks, there always exists an envy-free mechanism with optimal makespan (which runs in polynomial time).

Finally, in Section 6 we demonstrate how our mechanism for envy-free makespan minimization can be interpreted as a market clearing problem.

2 Preliminaries

In the scheduling notation of [13], the input to the problem of makespan minimization on unrelated machines is defined as follows: there are m machines, n tasks, and a matrix $(c_{ij})_{i\in[m],j\in[n]}$ such that c_{ij} is the time (load or cost) of running task j on machine i.

Machine scheduling can have either divisible or indivisible tasks. An assignment of tasks to machines is specified by an $m \times n$ matrix $a = (a_{ij})$, where a_{ij} is the fraction of job j assigned to machine i. A valid assignment must have $\sum_{i\in[m]} a_{ij} = 1$ for all jobs $j \in [n]$. If tasks are divisible then a_{ij} can take any value in $[0, 1]$. If tasks are indivisible, then $a_{ij} \in \{0, 1\}$.

[2] Several papers consider envy-free item pricing (in various scenarios) with the goal of maximizing revenue [15, 6, 5, 2], hardness results for revenue maximization envy-free item pricing appear in [9].

[3] The bounds given above hold for deterministic mechanisms, but randomization can reduce the approximation ratio. In particular, Mualem and Schapira [25] advocated a randomized truthful mechanism with an upper bound of $7m/8$ and showed a lower bound of $2 - 1/m$ for randomized mechanisms.

Let $c_i = (c_{i1}, \ldots, c_{in})$ be the i'th row of the cost matrix $c = (c_{ij})$ and let a_i be the i'th row of the assignment matrix $a = (a_{ij})$. The load (cost) of machine i under assignment (a_{ij}) is the inner product $c_i \cdot a_i = \sum_{j=1}^{n} c_{ij} a_{ij}$. The makespan of the assignment matrix a is $\max_{i \in [m]} c_i \cdot a_i$.

The problem of finding an assignment with a minimum makespan can be formulated as an integer program (IP) for indivisible jobs ($a_{ij} \in \{0, 1\}$) and as a linear program (LP) for divisible jobs ($a_{ij} \in [0, 1]$).

Following Nisan and Ronen [26] and Mu'alem [24], we consider the setting where the m machines are selfish agents. An allocation function a maps the cost matrix $c = (c_{ij})$ into an $m \times n$ assignment matrix $a(c)$. We denote by $a_i(c)$ the i'th row of the matrix $a(c)$. A payment function p is a mapping from the $m \times n$ cost matrix c to a real payment vector $p(c) = (p_1, p_2, \ldots, p_m)$, $p_i \in \Re$. Let $p_i(c)$ be the i'th coordinate of $p(c)$. We will just write a and p instead of $a(c)$ and $p(c)$ when c would be clear from the context.

A mechanism is a pair of functions, $M =< a, p >$, where a is an allocation function, and p is a payment function. For a mechanism $< a, p >$ with cost matrix $c = (c_{ij})$, the utility to agent i is $p_i(c) - c_i \cdot a_i(c)$. Such a utility function is known as quasi-linear.

A mechanism $< a, p >$ is *envy-free* if no agent prefers to switch her allocation and payment with another. *I.e.*, if for all agents $i, j \in [m]$ and all cost matrices c:

$$p_i(c) - c_i \cdot a_i(c) \geq p_j(c) - c_i \cdot a_j(c).$$

Based on Mu'alem [24], we say that an allocation function a is *envy-free implementable* (EF-implementable) if there exists a payment function p such that the mechanism $M =< a, p >$ is envy-free.

Consider the following definition of a locally efficient allocation.

Definition 2.1. An allocation function a is said to be *locally efficient* if for all cost matrices c and all permutations π of $1, \ldots, m$,

$$\sum_{i=1}^{m} c_i \cdot a_i(c) \leq \sum_{i=1}^{m} c_i \cdot a_{\pi(i)}(c).$$

Haake *et. al.* [16] provided a characterization of allocation functions that are EF-implementable, which is cast in the following theorem.[4]

Theorem 2.2. *[16] An allocation function a is EF-implementable if and only if a is locally efficient.*

3 An Upper Bound for Indivisible Jobs

In this section we present a polynomial time algorithm that produces a locally efficient, and hence, envy-free allocation of indivisible jobs whose makespan is at most $O(\log m)$ times the optimal makespan without envy-freeness constraints. In particular, our algorithm approximates the minimum makespan with envy-freeness constraints to within a factor of $O(\log m)$.

To simplify the description we assume that the algorithm starts with an allocation OPT that minimizes the makespan. If we were to start with an α-approximation to the minimal makespan, for some constant α, the final approximation would be $\alpha \cdot O(\log m) = O(\log m)$. Lenstra, Shmoys and Tardos [22] give a polynomial time 2-approximation algorithm for scheduling indivisible tasks.

The allocation a, which we start with, fixes a partition of the jobs into bundles[5] $B = \{b_1, \ldots, b_m\}$ where b_i is the set of jobs running on machine i. That is $j \in b_i$ if and only if $a_{ij} = 1$. Given a set of bundles

[4] An explicit proof of this theorem appears in [24].

[5] In this paper, bundles consist of jobs or other objects, and do not include payments which are dealt with separately.

$D = \{d_j\}_{j=1}^k$, $k \leq m$, we denote by $LE(D)$ a locally efficient assignment of D. This is an assignment of bundles to machines, no more than one bundle per machine, such that the sum of the loads is minimized. A locally efficient assignment of D can be computed using a weighted matching algorithm in polynomial time, for example, using the Hungarian algorithm [20].

We slightly abuse notation and use OPT to denote both the allocation and its makespan when no confusion can arise. We will sometimes use the bundle b also as the corresponding characteristic vector a, where $a_j = 1$ if $j \in b$ and $a_j = 0$ otherwise; no confusion will arise.

The algorithm works in phases. We start each phase with a subset of the bundles that have not been assigned to machines yet. We compute a locally efficient assignment of these bundles. Then if this locally efficient assignment contains a machine with load larger than $e \cdot$OPT we discard all bundles assigned to such machines (these bundles will be considered again only in the next iteration), and repeat the process with the remaining bundles until the makespan of the locally efficient allocation is at most $e \cdot$OPT. Thus, each phase produces an assignment of some subset of the bundles. The final assignment is the union of the assignments obtained in the different phases. Specifically, we assign to each machine the union of the bundles assigned to it in the different phases. The full description of the algorithm is given in the procedure FIND-APPROX.

Algorithm 1 Compute Envy-Free ($O(\log m)$)-Approximation

```
1:  procedure FIND-APPROX($B, c$)                    ▷ $B$ – set of OPT bundles, $c$ – cost matrix
2:      $q \leftarrow 0$
3:      $B_{out} \leftarrow \emptyset$
4:      $B_{active} \leftarrow B$
5:      while $B_{active} \neq \emptyset$ do
6:          $q \leftarrow q + 1$
7:          $a \leftarrow LE(B_{active})$
8:          while $makespan(a) > e \cdot$ OPT do
9:              for all $i$ do
10:                 if $c_i \cdot a_i > e \cdot$ OPT then
11:                     $B_{out} \leftarrow B_{out} \cup a_i$
12:                     $B_{active} \leftarrow B_{active} \setminus a_i$
13:                 end if
14:             end for
15:             $a \leftarrow LE(B_{active})$
16:         end while
17:         $a^q \leftarrow a$
18:         $B_{active} \leftarrow B_{out}$
19:         $B_{out} \leftarrow \emptyset$
20:     end while
21:     $a_i = \cup_{j=1}^{q} a_i^j$
22:     return $a$
23: end procedure
```

The following is the main result of this section.

Theorem 3.1. *The allocation computed in Algorithm* FIND-APPROX *is locally efficient and its makespan is at most* $e \ln m \cdot OPT = O(\log m)OPT$.

The proof of this theorem follows directly from the following Lemmata.

The first lemma shows that in each phase the number of bundles which we discard is at most a fraction $1/e$ of the number of bundles we start out with.

Lemma 3.2. *During a phase of Algorithm* FIND-APPROX *(lines 5-20) that starts with k bundles, no more than $\frac{k}{e} - 1$ bundles are discarded.*

Proof. Consider the set of bundles $B_{active} = \{b_{i_1}, \cdots, b_{i_k}\}$, $k = |B_{active}|$, at the beginning of a phase. Let a_i be the bundle assigned to machine i by the locally efficient assignment $LE(B_{active})$, and let $a_i = 0$ if no bundle is assigned to machine i by $LE(B_{active})$. By the definition of a locally efficient assignment it follows that the sum of loads in $LE(B_{active})$ is smaller than the sum of loads of the same bundles when placed by OPT, which is smaller than $k \cdot \text{OPT}$; i.e.,

$$\sum_{i=1}^{m} c_i \cdot a_i \leq \sum_{j=1}^{k} c_{i_j} \cdot b_{i_j} \leq k \cdot \text{OPT} .$$

Every time we throw out bundles in the inner loop (lines 8-16 of Algorithm FIND-APPROX) and recompute the locally efficient assignment of the remaining bundles, a bundle of size greater than $e \cdot \text{OPT}$ is removed; thus $\sum_i c_i \cdot a_i$ decreases by more than $e \cdot \text{OPT}$. Clearly, the sum $\sum_i c_i \cdot a_i$ never increases during a phase, since moving to a locally efficient assignment can only decrease the sum of loads. Together, it follows that the inner loop repeats less than $\frac{k \cdot \text{OPT}}{e \cdot \text{OPT}} = \frac{k}{e}$ times, implying that at most $\frac{k}{e} - 1$ bundles can join the set B_{out}. □

The following lemma follows directly from Lemma 3.2.

Lemma 3.3. *When Algorithm* FIND-APPROX *terminates $q \leq \ln m$.*

It follows from the definition of the algorithm that the makespan of the assignment a^j produced by phase j is at most $e \cdot \text{OPT}$. The final assignment assigns to each machine the union of the bundles assigned to it by the different phases. Since we have at most $\ln m$ phases we obtain that the makespan of the final assignment is at most $e \ln m \cdot \text{OPT}$. To complete the proof of Theorem 3.1 we have to show that the assignment which we produce is locally efficient. This follows from a more general observation that any union of locally efficient assignments is locally efficient, as established by the following lemma.

Lemma 3.4. *Let c be a cost matrix, and let b and b' be two assignments of different sets of jobs to the same set of machines. Let a be the assignment such that for every i, $a_i = b_i \cup b'_i$. If b and b' are locally efficient then so is a.*

Proof. Assume that a is not locally efficient then there is a permutation π of $1, 2, \ldots, m$ such that $\sum c_i \cdot a_{\pi(i)} < \sum c_i \cdot a_i$. By the definition of a, this implies that $\sum(c_i \cdot b_{\pi(i)} + c_i \cdot b'_{\pi(i)}) < \sum(c_i \cdot b_i + c_i \cdot b'_i)$ and, therefore, either $\sum c_i \cdot b_{\pi(i)} < \sum c_i \cdot b_i$ or $\sum c_i \cdot b'_{\pi(i)} < \sum c_i \cdot b'_i$, which contradicts the assumption that b is locally efficient or contradicts the assumption that b' is locally efficient, respectively. □

4 A Lower Bound for Indivisible Jobs

We give a lower bound of $\Omega(\log m / \log \log m)$ on the makespan approximation achievable by any locally efficient allocation of indivisible jobs. This shows that the upper bounded given in the previous section is almost tight.

Table 1: Cost matrix of the lower bound proof

$$c = \left(\begin{array}{cccccc}
1 & \infty & \infty & \infty \;\ldots & \infty & \infty \\
1-\frac{1}{2(n-1)} & 1 & \infty & \infty \;\ldots & \infty & \infty \\
1-\frac{2}{2(n-1)} & 1-\frac{1}{2(n-2)} & 1 & \infty \;\ldots & \infty & \infty \\
1-\frac{3}{2(n-1)} & 1-\frac{2}{2(n-2)} & 1-\frac{1}{2(n-3)} & 1 \;\ldots & \infty & \infty \\
\vdots & \vdots & \vdots & & & \vdots \\
1/2+\frac{1}{2(n-1)} & 1/2+\frac{1}{2(n-2)} & 1/2+\frac{1}{2(n-3)} & 1/2+\frac{1}{2(n-4)} \;\ldots & 1 & \infty \\
1/2 & 1/2 & 1/2 & 1/2 \;\ldots & 1/2 & 1 \\
\hline
2 & 2 & 2 & 2 \;\ldots & 2 & 2 \\
4 & 4 & 4 & 4 \;\ldots & 4 & 4 \\
\vdots & \vdots & \vdots & & & \vdots \\
2^\ell & 2^\ell & 2^\ell & 2^\ell \;\ldots & 2^\ell & 2^\ell
\end{array}\right)$$

Let n be the number of jobs. For every n we define the cost matrix $c = (c_{ij})$ with $m = n + \ell$ machines where $2^\ell = \log n/(4 \log\log n)$ as follows.

Row i, $1 \le i \le n + \ell$, of the cost matrix c corresponds to the ith machine and c_{ij} is the cost of running job j on machine i. The horizontal line, which lies between machine n and $n+1$, is drawn merely to emphasize the the two different parts of the matrix. For $1 \le i \le n$, machine i has cost 1 for job i, costs $1 - (i-j)/(2(n-j))$ for jobs $j < i$, and cost of ∞ for the rest of jobs ($j > i$). For $n+1 \le i \le n+\ell$, all costs of machine i are equal to 2^i. Observe that $c_{ij} - c_{(i+1)j} = 1/(2(n-j))$ for $1 \le i < n$ and $j \le i$.

The optimal makespan of all these matrices is 1. We can achieve makespan 1 by running job i on machine i for every $1 \le i \le n$, and we cannot do better since job n has load ≥ 1 on all machines.

We establish a lower bound of $2^\ell = \log n/(4 \log\log n)$ on the makespan of any envy-free allocation for this instance. This shows that we cannot have an algorithm that always produces an envy-free allocation whose makespan approximates the optimal makespan to within a factor smaller than $\log n/(4 \log\log n)$.

Specifically, we show that for *any* partition of the jobs into $\le n + \ell$ bundles, any locally efficient assignment of these bundles to the machines has makespan at least 2^ℓ. Our first lemma makes few easy observations regarding allocations with makespan $< 2^\ell$.

Lemma 4.1. *For the cost matrix (c_{ij}) above, any allocation with makespan $< 2^\ell$ satisfies the following.*

1. *There are fewer than $2^{\ell+1}$ jobs on each machine.*
2. *There are fewer than $2^\ell/2^{(i-n)}$ jobs on each machine $n < i \le n+\ell$.*
3. *There are fewer than a total of 2^ℓ jobs running on the set of machines $n+1, \ldots, n+\ell$.*

Proof. (1) follows since $c_{ij} \ge 1/2$ for all i and j. (2) follows since for $n < i \le n+\ell$, $c_{ij} \ge 2^{i-n}$. (3) follows by summing the upper bound on the number of jobs on each of these machines, this sum can be at most $\sum_{i=n+1}^{(n+\ell)} (2^\ell/2^{(i-n)} - 1) = 2^\ell - \ell < 2^\ell$. $\square$

We can now conclude with the proof of the lower bound:

Theorem 4.2. *For any partition of jobs into bundles, the makespan of any locally efficient assignment of the bundles is at least* $2^\ell = \log n/(4 \log\log n)$.

Proof. Fix an arbitrary partition into bundles and suppose that there is an envy-free assignment of the bundles with makespan $< 2^\ell$. We will derive a contradiction by showing that the assignment is not locally efficient.

Since the makespan is $< 2^\ell$ no bundle is assigned to machine $n + \ell$. So we derive the contradiction by showing that if we move the bundle assigned to machine i to machine $i + 1$ for $1 \leq i < n + \ell$ we decrease the total load.

By Lemma 4.1(1), there are $\leq 2^{\ell+1} - 1$ jobs in the bundle assigned to machine n. So moving this bundle to machine $n + 1$ increases the load of this bundle by at most $3/2 \cdot 2^{\ell+1}$.

Since $c_{(i+1)j} = 2c_{ij}$ for $n + 1 \leq i < n + \ell$ and all j, moving a bundle from machine i to machine $i + 1$ for $n + 1 \leq i < n + \ell$ increases the load of this bundle exactly by a factor of 2. Since the load on each of these machines is $< 2^\ell$, the total increase in load caused by moving each of these bundles one machine down is $< \ell \cdot 2^\ell$.

Summing up we obtain that the increase in the load due to moving bundles on machines $n, \ldots, n + \ell$ is at most $(3/2)2^{\ell+1} + \ell 2^\ell = (\ell + 3)2^\ell$. Substituting $2^\ell = \log n/(4 \log\log n)$ we obtain that this increase for large n is smaller than $\log n/4$ which is smaller than, say, $\ln n/2.5$.

By Lemma 4.1(3), at most 2^ℓ jobs are in bundles assigned to machine $n + 1, \ldots, n + \ell$ and, by Lemma 4.1(1), at most $2^{\ell+1}$ jobs are in the bundle assigned to machine n. Therefore, at least $n - 3 \cdot 2^\ell$ jobs are in bundles assigned to machines $1, \ldots, n - 1$. If job j is in one of these bundles then after we move these bundles the contribution of job j to the load decreases by $1/(2(n - j))$. So the total decrease in load due to moving bundles assigned to machines $1, \ldots, n-1$ is at least $(1/2)(H_n - H_{3 \cdot 2^\ell}) \approx (1/2)(\ln n - \ln(3 \cdot 2^\ell)) \geq (1/2 - \epsilon) \ln n$ for large enough n.

Since the decrease in the load caused by moving bundles on machines $1, \ldots, n - 1$ is larger than the increase in the load caused by moving bundles on machines $n, \ldots, n + \ell$ we obtain a contradiction. □

Since $m = n + \ell = O(n)$, we get that it is $\Omega(\log m/ \log\log m)$ approximation, as promised.

5 Unrelated Machine Scheduling with Divisible Jobs

The existence of an envy-free assignment for divisible tasks is trivial, even without payments. For example, simply assigning each machine a $1/m$ fraction of every job is trivially envy-free. However, it is certainly not optimal with respect to makespan minimization. The question that is of interest in this section is how close to the optimal makespan can we get if the tasks are divisible.

Recall that for indivisible tasks, the previous section establishes a lower bound of $\Omega(\log m/ \log\log m)$. Here we prove that when tasks are divisible, there always exists an optimal envy-free allocation; i.e., one that achieves the minimum makespan. In order to find such an allocation, solve the linear program that minimizes the makespan subject to two sets of constraints: one which ensures the validity of the assignment and one which ensures that the assignment is envy-free. The main challenge is to prove that this LP formulation has a solution. This is established in the following theorem.

Theorem 5.1. *For any instance of machine scheduling with divisible jobs, there is a locally efficient assignment with minimum makespan.*

Consider an instance of the machine scheduling problem, specified by the cost matrix $c = (c_{ij})$. As we deal with divisible assignments, bundles are sets of fractions of tasks. An assignment itself is represented as a real valued matrix, (a_{ij}), where a_{ij} is the fraction of task j assigned to machine i. As before we use a_i to denote the i'th row of the assignment matrix (a_{ij}), and also to denote the the set of fractional tasks assigned to agent i.

Before dwelling into the proof of the general case, we begin with a warm up, which establishes the statement of the theorem for the simple case of two machines with finite valuations.

Warm up: Consider an instance with two machines $i \in \{1, 2\}$ such that all entries in c are finite. We show that *any* assignment with minimum makespan must be locally efficient. Let o be an optimal assignment and assume by contradiction that o is not locally efficient. Without loss of generality, we can assume that the makespan of o is 1 and that both machines have the same load $c_1 \cdot o_1 = c_2 \cdot o_2 = 1$, where o_i is the bundle assigned to machine i ($i \in \{1, 2\}$). (Otherwise, we can transfer (fractional) jobs from the machine with load 1 to the other machine and get an assignment a with a strictly lower makespan than o, which contradicts the optimality of o.)

Our assumption that o is not locally efficient implies that we stricktly decrease the total load to be smaller than 2 if we assign the bundle of machine 2 to machine 1 and vice versa. Without loss of generality, we assume that $c_1 \cdot o_2 = 1 - y$ and $c_2 \cdot o_1 = 1 + x$ for some $y > x \geq 0$.

We now construct a new assignment a, whose makespan is smaller than 1. This will contradict the optimality of o. Let $\epsilon = (y - x)/2$ and let $a_1 = o_2 + (y - \epsilon)o_1$ and $a_2 = (1 - y + \epsilon)o_1$. It is easy to see that a is well defined for any $0 < \epsilon < y$. The makespan of assignment a is $\max\{c_1 \cdot a_1, c_2 \cdot a_2\}$. The load of a on machine 1 is $c_1 \cdot a_1 = (1 - y) + (y - \epsilon) = 1 - \epsilon < 1$; the load of a on machine 2 is $c_2 \cdot a_2 = (1 - y + \epsilon)(1 + x) = 1 - y + x - yx + \epsilon + \epsilon x \leq 1 - (y - x) + \epsilon(1 + x) < 1 - (y - x) + 2\epsilon = 1$. It follows that $\max\{c_1 \cdot a_1, c_2 \cdot a_2\} < 1$, which contradicts the optimality of o.

We next establish the statement in its general form.

General instance: Consider a cost matrix $c = (c_{ij})$ of the machine scheduling problem with $m \geq 2$ machines which may include $+\infty$ entries. We first define a lexicographic order on assignments.

Definition 5.2. A vector $(l_1, \ldots, l_m)$ is smaller than $(l'_1, \ldots, l'_m)$ lexicographically if for some i, $l_i < l'_i$ and $l_k = l'_k$ for all $k < i$. An assignment a is smaller than a' lexicographically if the vector of machine loads $l(a) = (l_1(a), \ldots, l_m(a))$, sorted in non-increasing order, is lexicographically smaller than $l(a')$, sorted in non increasing order.

Clearly, every lexicographically minimal assignment has minimum makespan. When all entries are finite, any assignment with minimum makespan has equal loads on all machines and therefore minimum makespan implies a lexicographically minimal assignment.[6] In either case (with all entries finite or not), there exists some lexicographically minimal schedule with minimal makespan. In order to prove Theorem 5.1, it is therefore sufficient to prove that a lexicographically minimal schedule is also locally efficient.

Lemma 5.3. *Every lexicographically minimal assignment is locally efficient.*

Proof. Assume toward contradiction that o is a lexicographically minimal assignment that is not locally efficient and let ℓ be a locally efficient assignment of the bundles of o (i.e. $\ell = LE(o)$). Consider a directed graph G, where the nodes correspond to machines and the arcs correspond to a reassignment of bundles between o and ℓ. We also include empty bundles and hence this reassignment is a permutation. Each machine has either no incoming and outgoing arcs or exactly one incoming and exactly one outgoing arc. The graph G is therefore a collection of singletons and cycles.

[6]To see this, suppose toward contradiction that this is not the case. Consider an assignment with minimum makespan. Let $\mathcal{M}' \subset [m]$ be machines with load strictly lower than the makespan. We can transfer (fractional) jobs from $[m] \setminus \mathcal{M}'$ machines to $\mathcal{M}'$ machines and obtain an assignment with strictly lower makespan, which contradicts optimality.

Since $\ell \neq o$, and there are no paths, G must contain a cycle. Moreover, because ℓ is locally efficient and o is not, G must contain a cycle X such that

$$\sum_{i \in X} c_i \cdot o_i > \sum_{i \in X} c_i \cdot \ell_i \ . \tag{1}$$

Consider such a cycle X with $|X| = k \geq 2$ nodes (machines). We (re-)number machines such that machines along the cycle are indexed $[0, \ldots, k-1]$ in cyclic order. We also renumber the bundles accordingly such that the bundles of machine i are o_i and ℓ_i. By definition, $\ell_{i+1} = o_i$ (all addition operations through this section are modulo k).

We claim that all machines in the cycle X must be equally loaded under o, that is, $c_i \cdot o_i = c_j \cdot o_j$ for every $0 \leq i, j \leq k-1$.[7] Assume for a contradiction that there are two consecutive machines on X, i and $i+1$, such that $c_i \cdot o_i > c_{i+1} \cdot o_{i+1}$. We construct an assignment a from o by shifting a fraction f of the bundle o_i from machine i to machine $i+1$ such that both machines have equal loads. That is, f is chosen such that $(1-f)c_i \cdot o_i = c_{i+1} \cdot (o_{i+1} + f o_i)$ or explicitly $f = \frac{c_i \cdot o_i - c_{i+1} \cdot o_{i+1}}{(c_i + c_{i+1}) \cdot o_i}$. Since $\ell_{i+1} = o_i$ and $c_{i+1} \cdot \ell_{i+1} < \infty$, it follows that $c_{i+1} \cdot o_i < \infty$ and therefore $f > 0$. The assignment a is strictly lexicographically smaller than o, which is a contradiction.

By scaling, we can assume that $c_i \cdot o_i = 1$ for all $0 \leq i \leq k-1$. Let $c_{i+1} \cdot \ell_{i+1} = 1 + \Delta_i$ be the load of machine $i+1$ under ℓ ($\Delta_i \geq -1$.) From (1) it holds that

$$\sum_{i=0}^{k-1} \Delta_i < 0 \ . \tag{2}$$

We conclude the proof by constructing an assignment a that is identical to o outside the cycle, has $\sum_{i=0}^{k-1} a_i = \sum_{i=0}^{k-1} o_i$, that is, same total allocation as o on cycle machines, has load $c_i \cdot a_i = 1$ on machines $i = 1, \ldots, k-1$ and load $c_0 \cdot a_0 < 1$ on machine 0. The assignment a is strictly lexicographically smaller than o, which contradicts the assumption that o is a lexicographically minimum assignment.

The assignment a is such that for $i = 0, \ldots, k-1$, an $0 \leq \alpha_i \leq 1$ fraction of o_i is assigned to machine i and the remaining $(1 - \alpha_i)$ is assigned to machine $i+1$. Define

$$\mu = \max\{1, \max_{i=0 \ldots k-1} \prod_{j=0}^{i} (1 + \Delta_i)\} \ , \tag{3}$$

$\alpha_0 = 1 - \frac{1}{2\mu}$, and for $i = 1, \ldots, k-1$:

$$(1 - \alpha_i) = (1 - \alpha_{i-1})(1 + \Delta_{i-1}) \ . \tag{4}$$

It follows that for $i = 1, \ldots, k-1$,

$$(1 - \alpha_i) = (1 - \alpha_0) \prod_{j=0}^{i-1} (1 + \Delta_j) \ . \tag{5}$$

We show that all α_i are well defined (i.e., $\alpha_i \in [0,1]$): Since $1 \leq \mu < \infty$, $\alpha_0 \in [1/2, 1)$. For $i = 1, \ldots, k-1$, using (5) and (3) we obtain

$$(1 - \alpha_i) = \frac{1}{2\mu} \prod_{j=0}^{i-1} (1 + \Delta_j) \leq \frac{1}{2} < 1 \ . \tag{6}$$

[7] This is immediate if there are no $+\infty$ entries in v.

As a product of positive quantities, $(1-\alpha_i) \geq 0$.
The load a_i places on machine i ($i = 1, \ldots, k-1$) is (using (4)):

$$c_i \cdot a_i = \alpha_i + (1+\Delta_{i-1})(1-\alpha_{i-1}) = \alpha_i + (1-\alpha_i) = 1 \ . \tag{7}$$

The load a_0 places on machine 0 is (using (5)):

$$\begin{aligned} c_0 \cdot a_0 &= \alpha_0 + (1+\Delta_{k-1})(1-\alpha_{k-1}) \\ &= \alpha_0 + (1-\alpha_0)\prod_{j=0}^{k-1}(1+\Delta_j) \\ &< \alpha_0 + (1-\alpha_0) = 1 \ , \end{aligned}$$

where the strict inequality follows from $(1-\alpha_0) = 1/(2\mu) > 0$ and from $\prod_{j=0}^{k-1}(1+\Delta_j) < 1$ (which follows from (2)). $\square$

6 Market clearing prices for makespan minimization

In our model, a mechanism receives the agents' valuations as an input and computes an envy-free outcome (i.e., allocation and payments). One can reinterpret our mechanism for envy-free makespan minimization as a market clearing problem.

We first argue that any envy-free mechanism can be modified so that agents assigned the empty bundle receive payment zero (and thus utility zero) and that all utilities are non-negative. We refer to such payments as normalized payments. Given an envy-free mechanism $< a, p >$, fix a cost matrix c and consider the allocation $a(c)$ and the payments $p(c)$. We can add an arbitrary constant to the payments of every agent up to the point where all utilities are non-negative and the mechanism remains envy-free. If there exists an agent who receives the empty bundle, then her payment must be zero, otherwise the agent with utility zero envies that agent.

Let the minimal payment to any agent be d. We replace the payment function p with p' where $p'_i(c) = p_i(c) - d$ for all agents i. If the mechanism $< a, p >$ was envy-free then so is $< a, p' >$ but the minimal payment to any agent under $< a, p' >$ is zero. In particular, any agent i that is allocated the empty bundle must receive the minimal payment under p (otherwise any agent receiving less than $p_i(c)$ will envy agent i). Thus, agents allocated the empty bundle receive zero payment under p'.

With this we are ready to reinterpret our mechanism as a market clearing problem. A central authority has a large set of tasks that must be performed in parallel by a set of agents, each of which has different capabilities. The agents report their costs for the tasks, and the authority computes non-negative payment offers for all bundles (where the payment for the empty bundle is zero). This notion is closely related to the notion of a Walrasian equilibrium. However, unlike a Walrasian equilibrium, which has supporting *item* prices, in our case the authority associates a payment to every bundle, where a bundle price may not necessarily be equal to the sum of its items' prices.

As we shall show next, our results imply that it is possible to compute payments for all the bundles such that each agent will choose a bundle that maximizes her utility (i.e., payment - costs) in a way that all the tasks will be performed, no task will be assigned to more than one agent, and the makespan will be within a logarithmic factor of the minimal makespan.

Fix a cost matrix and let $< a, p >$ be an envy-free mechanism with normalized payments. Consider the bundles assigned to agents $1, \ldots, n$, namely $a_1(c), a_2(c), \ldots, a_n(c)$, and their associated payments

$p_1(c), p_2(c), \ldots, p_n(c)$. The authority implicitly offers payments for all possible bundles of tasks as follows; for every bundle B let S_B be the set of agents i such that $a_i(c) \subseteq B$. The payment offered by the authority for bundle B is

$$p(B) = \begin{cases} \max_{j \in S_B} p_j(c) & S_B \neq \emptyset \\ 0 & S_B = \emptyset \end{cases} \tag{8}$$

Observation 6.1. Let $< a, p >$ be an envy-free mechanism with normalized payments, and consider the bundle payments given in Equation (8). Then,

$$a_i \in argmax_{B \subseteq 2^M}(p(B) - \sum_{j \in B} c_{ij});$$

(i.e., a_i is an optimal bundle for agent i).

Proof. Suppose toward contradiction that under the payments specified in Equation (8) there exists a bundle a' such that

$$p(a') - \sum_{j \in a'} c_{ij} > p(a_i) - \sum_{j \in a_i} c_{ij}.$$

Let $S_{a'}$ be the set of agents ℓ such that $a_\ell(c) \subseteq a'$. We distinguish between two cases. If $S_{a'} \neq \emptyset$, then let $i' = argmax_{i \in S_{a'}} p_i(c)$. Since $a_{i'}(c) \subseteq a'$, it holds that $\sum_{j \in a_{i'}(c)} c_{ij} \leq \sum_{j \in a'} c_{ij}$ and from the definition of the payment and the definition of i' it follows that $p(a_{i'}(c)) = p(a')$. It follows that $p(a') - \sum_{j \in a'} c_{ij} \leq p(a_{i'}(c)) - \sum_{j \in a_{i'}(c)} c_{ij}$. Observing that $p(a_{i'}(c)) - \sum_{j \in a_{i'}(c)} c_{ij} \leq p(a_i(c)) - \sum_{j \in a_i(c)} c_{ij}$ (which follows from the envy freeness of the mechanism $< a, p >$) arrives at a contradiction to the envy of agent i. Otherwise, $S_{a'} = \emptyset$. In this case $p(a') = 0$, and it follows that $p(a') - \sum_{j \in a'} c_{ij} \leq 0$. But the original mechanism is normalized so that the utility of every agent is non-negative, in contradiction to the envy of agent i. □

A direct corollary of the last observation is that the bundle payments given in Equation (8) clear the market, i.e., if each agent chooses a bundle that maximizes her utility, then all the tasks will be performed and no task will be assigned to more than one agent. Moreover, the resulting makespan is no worse than $O(\log m)$ times the minimal makespan.

7 Summary and Open Problems

Table 2 summarizes upper and lower bounds on the ratio of the optimal makespan of machine scheduling with envy-freeness constraints and the optimal makespan without envy-freeness constraints. The upper bounds correspond to polynomial time algorithms. An obvious challenge is to close the gap between the upper and lower bounds for indivisible tasks.

	Lower bound	**Upper bound**
(Divisible+EF)/Divisible	1	1 (Thm. 5.1)
(Indivisible+EF)/Indivisible	$\Omega(\frac{\log m}{\log \log m})$ (Thm. 4.2)	$O(\log m)$ (Thm. 3.1)

Table 2: Summary of our results on the cost of envy-freeness. The rows correspond to divisible or indivisible tasks. The columns correspond to upper bounds on the ratio and lower bounds on the *worst-case* ratio. The number of machines is m.

An intriguing issue is to understand the interaction of envy-freeness and incentive compatibility. What can we say about the makespan approximation for mechanisms that are *both* envy-free and incentive compatible? Clearly, any $o(m)$ approximation that is both incentive compatible and envy-free would be a major breakthrough. Recently, Ashlagi, Dobzinski, and Lavi [1] gave a lower bound of $\Omega(m)$ on makespan approximation for incentive compatible and *anonymous* mechanisms. What happens if we discard the *anonymous* assumption but require that the mechanism also be envy-free?

Minimum makespan machine scheduling is classically formulated as a linear program (for divisible jobs) or an integer program (for indivisible jobs), both with the same set of linear constraints. The requirement of envy-freeness can be captured by adding payment variables (that are not required to be integral) as additional linear constraints. Accordingly, for a cost matrix (c_{ij}), we denote the optimal makespan with or without integrality or envy-freeness by $T_{\mathtt{LP}}(c_{ij})$, $T_{\mathtt{IP}}(c_{ij})$, $T_{\mathtt{LP+EF}}(c_{ij})$, and $T_{\mathtt{IP+EF}}(c_{ij})$, respectively. Using this notation, Table 2 lists bounds on the ratios $T_{\mathtt{IP+EF}}(c_{ij})/T_{\mathtt{IP}}(c_{ij})$ (indivisible) and $T_{\mathtt{LP+EF}}(c_{ij})/T_{\mathtt{LP}}(c_{ij})$ (divisible).

Starting with divisible tasks without envy-freeness constraints ($T_{\mathtt{LP}}(c_{ij})$) we consider the impact on the optimal makespan of integrality and envy-freeness. Envy-freeness requirement alone does not result in an increase of the optimal makespan (Thm. 5.1). There are instances (the instances in our lower bound construction in Thm. 4.2) where the integrality requirement (indivisible tasks) results in at most a factor 2 increase while, curiously, the combination of *both* requirements results in $\Omega(\log m/\log\log m)$ factor increase.

Considering the approximability of the optimal makespan under the different types of constraints, $T_{\mathtt{LP}}$ and $T_{\mathtt{LP+EF}}$ are linear programs and hence solvable in polynomial time and $T_{\mathtt{IP}}$ has a 2 approximation algorithm and an inapproximability result of 1.5 [22].

As for $T_{\mathtt{IP+EF}}$, we provided an $O(\log m)$ approximation algorithm and we know the problem is NP-hard because integral machine scheduling with identical machines is known to be NP-hard (by a reduction to partition) and any assignment on identical machines is trivially locally efficient and hence EF. This leaves a wide gap as for the (in)approximability of $T_{\mathtt{IP+EF}}$. Closing this gap seems challenging:

- A 2-approximation algorithm for $T_{\mathtt{IP}}(c_{ij})$ was constructed using the relation to $T_{\mathtt{LP}}(c_{ij})$ [22]. This approximation algorithm is based on taking a fractional schedule a and rounding it to an integral one with a makespan larger by at most an additive term of $\max_{i,j|a_{ij}>0} c_{ij}$ over that of a, where c_{ij} is the time required by machine i to run job j. This approach does not immediately carry over, when starting from a fractional envy-free schedule, because the EF constraints might be violated when rounding.

- The inapproximability result of 1.5 for $T_{\mathtt{IP}}(c_{ij})$ [22] was for makespan minimization. However, the instance used is in fact envy-free. Thus, [22] further implies that one cannot approximate the *minimal makespan and envy-free assignment* to within a factor of 1.5 in polynomial time.

- Lastly, our lower bound on the ratio $T_{\mathtt{IP+EF}}(c_{ij})/T_{\mathtt{IP}}(c_{ij})$ precludes obtaining a tighter approximation using a better rearrangement of the bundles of a solution to $T_{\mathtt{IP}}(C_{ij})$ to achieve envy-freeness.

8 Acknowledgments

Michal Feldman is partially supported by the Israel Science Foundation (grant number 1219/09) and by the Leon Recanati Fund of the Jerusalem school of business administration. Amos Fiat, Haim Kaplan, and Svetlana Olonetsky are partially supported by ISF grant no. 975-06.

References

[1] I. Ashlagi, S. Dobzinski, and R. Lavi. An optimal lower bound for anonymous scheduling mechanisms. In *Proceedings of the ACM Conference on Electronic Commerce*, 2009.

[2] N. Bansal, N. Chen, N. Cherniavsky, A. Rudra, B. Schieber, and M. Sviridenko. Dynamic pricing for impatient bidders. In Bansal et al. [3], pages 726–735.

[3] N. Bansal, K. Pruhs, and C. Stein, editors. *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007*. SIAM, 2007.

[4] S. J. Brams and A. D. Taylor. *Fair Division: From Cake-Cutting to Dispute Resolution*. Cambridge University Press, 1996.

[5] P. Briest. Uniform budgets and the envy-free pricing problem. In L. Aceto, I. Damgård, L. A. Goldberg, M. M. Halldórsson, A. Ingólfsdóttir, and I. Walukiewicz, editors, *ICALP (1)*, volume 5125 of *Lecture Notes in Computer Science*, pages 808–819. Springer, 2008.

[6] N. Chen, A. Ghosh, and S. Vassilvitskii. Optimal envy-free pricing with metric substitutability. In L. Fortnow, J. Riedl, and T. Sandholm, editors, *ACM Conference on Electronic Commerce*, pages 60–69. ACM, 2008.

[7] G. Christodoulou, E. Koutsoupias, and A. Kovács. Mechanism design for fractional scheduling on unrelated machines. In L. Arge, C. Cachin, T. Jurdzinski, and A. Tarlecki, editors, *ICALP*, volume 4596 of *Lecture Notes in Computer Science*, pages 40–52. Springer, 2007.

[8] G. Christodoulou, E. Koutsoupias, and A. Vidali. A characterization of 2-player mechanisms for scheduling. In *Proceedings of the 16th annual European symposium on Algorithms*, pages 297 – 307, 2008. Comment: 20 pages, 4 figures, ESA'08.

[9] E. D. Demaine, M. T. Hajiaghayi, U. Feige, and M. R. Salavatipour. Combination can be hard: approximability of the unique coverage problem. In *SODA*, pages 162–171. ACM Press, 2006.

[10] L. Dubins and E. Spanier. How to cut a cake fairly. *American Mathematical Monthly*, 68:1–17, 1961.

[11] D. Foley. Resource allocation and the public sector. *Yale Economic Essays*, 7:45–98, 1967.

[12] M. Gardner. *aha! Insight*. W.H. Freeman & Company, 1978.

[13] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. R. Kan. Optimization and approximation in deterministic sequencing and scheduling: A survey. *Ann. Discrete Math.*, 5:287–326, 1979.

[14] F. Gul and E. Stacchetti. Walrasian equilibrium with gross substitutes. *Journal of Economic Theory*, 87:95–124, 1999.

[15] V. Guruswami, J. D. Hartline, A. R. Karlin, D. Kempe, C. Kenyon, and F. McSherry. On profit-maximizing envy-free pricing. In *SODA*, pages 1164–1173. SIAM, 2005.

[16] C.-J. Haake, M. G. Raith, and F. E. Su. Bidding for envyfreeness: A procedural approach to n-player fair-division problems. *Social Choice and Welfare*, 19:723–749, 2002.

[17] A. D. T. Julius B. Barbanel. *The Geometry of Efficient Fair Division*. Cambridge University Press, 2005.

[18] D. Kempe, A. Mu'alem, and M. Salek. Envy-free allocations for budgeted bidders. In *Workshop on Internet and Network Economics*.

[19] E. Koutsoupias and A. Vidali. A lower bound of $1+\phi$ for truthful scheduling mechanisms. In L. Kucera and A. Kucera, editors, *MFCS*, volume 4708 of *Lecture Notes in Computer Science*, pages 454–464. Springer, 2007.

[20] H. W. Kuhn. The Hungarian Method for the assignment problem. Naval Research Logistics Quarterly (2), 1955.

[21] R. Lavi and C. Swamy. Truthful mechanism design for multi-dimensional scheduling via cycle monotonicity. In J. K. MacKie-Mason, D. C. Parkes, and P. Resnick, editors, *Proceedings 8th ACM Conference on Electronic Commerce (EC-2007), San Diego, California, USA, June 11-15, 2007*, pages 252–261. ACM, 2007.

[22] J. K. Lenstra, D. B. Shmoys, and É. Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Math. Program.*, 46:259–271, 1990.

[23] H. Moulin. *Fair Division and Collective Welfare*. MIT Press, 2004.

[24] A. Mu'alem. On multi-dimensional envy-free mechanisms. In *First International Conference on Algorithmic Decision Theory*.

[25] A. Mu'alem and M. Schapira. Setting lower bounds on truthfulness: extended abstract. In Bansal et al. [3], pages 1143–1152.

[26] N. Nisan and A. Ronen. Algorithmic mechanism design (extended abstract). In *STOC*, pages 129–140, 1999.

[27] J. Rawls. *A Theory of Justice*. Harvard University Press, 2005.

[28] D. Raz and H. Levy and B. Avi-Itzhak. *A resource-allocation queueing fairness measure*. SIGMETRICS Perform. Eval. Rev. 32:1, 130–141, 2004.

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

[30] H. Steinhaus. *Mathematical Snapshots*. Oxford University Press, 1951.

[31] H. P. Young. *Equity: In Theory and Practice*. Princeton University Press, 1995.