Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)

Algorithms for Max-Min Share Fair Allocation of Indivisible Chores

Haris Aziz
Data61 and UNSW
Sydney, NSW 2052, Australia
haris.aziz@data61.csiro.au

Gerhard Rauchecker
University of Regensburg
93053 Regensburg, Germany
gerhard.rauchecker@ur.de

Guido Schryen
University of Regensburg
93053 Regensburg, Germany
guido.schryen@ur.de

Toby Walsh
UNSW, Data61 and TU Berlin
Sydney, NSW 2052, Australia
toby.walsh@data61.csiro.au

Abstract

We consider Max-min Share (MmS) fair allocations of indivisible chores (items with negative utilities). We show that allocation of chores and classical allocation of goods (items with positive utilities) have some fundamental connections but also differences which prevent a straightforward application of algorithms for goods in the chores setting and vice-versa. We prove that an MmS allocation does not need to exist for chores and computing an MmS allocation - if it exists - is strongly NP-hard. In view of these non-existence and complexity results, we present a polynomial-time 2-approximation algorithm for MmS fairness for chores. We then introduce a new fairness concept called optimal MmS that represents the best possible allocation in terms of MmS that is guaranteed to exist. We use connections to parallel machine scheduling to give (1) a polynomial-time approximation scheme for computing an optimal MmS allocation when the number of agents is fixed and (2) an effective and efficient heuristic with an ex-post worst-case analysis.

Introduction

Fair allocation of indivisible items is a central problem in economics, computer science, and operations research (Aziz et al. 2015; Brams and Taylor 1996; Bouveret, Chevaleyre, and Maudet 2015; Lipton et al. 2004). We focus on the setting in which we have a set of N agents and a set of items with each agent expressing utilities over the items. The goal is to allocate the items among the agents in a fair manner without allowing transfer of money. If all agents have positive utilities for the items, we can view the items as goods. On the other hand, if all agents have negative utilities for the items, we can view the items as chores. In this paper we focus on fair allocation of indivisible chores. Although multi-agent resource allocation has been extensively studied, mostly the resources considered are goods, i.e., they yield positive utility. On the other hand, several important problems require the items to be allocated to yield negative utility. This is particularly so when the items in question are project tasks, house-hold chores, or climate change actions to be handled by different countries.

In order to identify fair allocations, one needs to formalize what fairness means. A compelling fairness concept called *Max-min Share (MmS)* was recently introduced which holds more often than traditional fairness concepts such as envy-freeness and proportionality (Bouveret and Lemaître 2016; Budish 2011). An agent's MmS is the "most preferred bundle he could guarantee himself as a divider in divide-and-choose against adversarial opponents" (Budish 2011). The main idea is that an agent partitions the items into N sets in a way that maximizes the utility of the least preferred set in the partition. The utility of the least preferred set is called the *MmS guarantee* of the agent. An allocation satisfies *MmS fairness* if each agent gets at least as much utility as her MmS guarantee. We refer to such an allocation as *MmS allocation*.[1]

Although MmS is a highly attractive fairness concept and a natural relaxation of proportionality and envy-freeness (Bouveret and Lemaître 2016), Procaccia and Wang (2014) showed that an MmS allocation of goods does not exist in general. This fact initiated research on approximate MmS allocations of goods in which each agent gets some fraction of her MmS guarantee. On the positive side, MmS allocations of goods often exist (Kurokawa, Procaccia, and Wang 2016) and there also exists a polynomial-time algorithm that returns 2/3-approximate MmS allocations (Procaccia and Wang 2014; Amanatidis et al. 2015). Algorithms for computing approximate MmS allocations of goods are being used in practice for fair division in real-world problems (Goldman and Procaccia 2014).

In this paper, we turn to MmS allocations of chores, a subject which has not been studied previously. Even in the more general domain of fair allocation, there has been less research on chore allocation compared to goods despite there being many settings where we have chores but not goods (Brams and Taylor 1996). However, the problem of allocation of chores cannot in general be simply transformed into a problem of allocation of goods (Caragiannis et al. 2012). The difference in the two settings can also be judged from the fact that for divisible goods, utilities

[1] Bouveret and Lemaître (2016) and Budish (2011) also formalized a fairness concept called min-Max fairness that is stronger than Max-min fairness.

achieved under competitive equilibrium are unique but for divisible chores, utilities achieved under competitive equilibrium are not unique (Bogomolnaia and Moulin 2016).

Contributions We consider MmS allocation of chores for the first time and present fundamental connections between allocation of chores and goods when the negative utilities of the agents in the case of chores are negated to obtain a goods setting and vice-versa. We also show that there are fundamental differences between the two settings with no known simple reductions between them. In particular, reductions such as negating the utility values and applying an algorithm for one setting does not give an algorithm for the other setting. We show that an MmS allocation does not need to exist for chores and that calculating an MmS allocation for chores is NP-hard in the strong sense.

In view of the non-existence result, we introduce a new concept called *optimal MmS* for chores. An allocation is an *optimal MmS allocation* if it represents the best possible approximation of the MmS guarantee. An optimal MmS allocation has two desirable properties: (1) it always exists and (2) it satisfies MmS fairness whenever an MmS allocation exists. Consequently, optimal MmS is a compelling fairness concept and a conceptual contribution of the paper. We present bounds to quantify the gap between optimal MmS fairness and MmS fairness. We present a polynomial-time greedy round-robin algorithm for this purpose that provides a 2-approximation of the MmS guarantee for chores.

In view of the computational hardness result, we use connections to parallel machine scheduling to develop an algorithm that gives an approximation for optimal MmS fairness for an arbitrary ex-ante error tolerance. Using well-established polynomial-time approximation schemes (PTAS) for the involved parallel machine scheduling problems, we show that this algorithm gives a PTAS for optimal MmS when the number of agents is fixed. In addition to this theoretical result, we develop an efficient and effective solution heuristic (called SCHED). The main difference of the PTAS and the SCHED heuristic is that the former solves the underlying scheduling problems within an ex-ante error tolerance while the latter uses fast heuristics (or a time limit) to solve the underlying scheduling problems.

Not allowing for an ex-ante worst-case analysis anymore, we give an ex-post worst-case analysis to benchmark the SCHED heuristic and test its performance in computational experiments. We show that SCHED is capable of deriving near-optimal solutions within a few minutes in our computational experiments for instances with up to 128 agents and 3,200 chores. We also prove that SCHED greatly outperforms the simpler greedy round-robin 2-approximation algorithm.

Related Work

Fair allocation has been extensively studied for allocation of divisible goods such as cake (Brams and Taylor 1996). However when items are divisible, MmS fairness coincides with the classic fairness notion of proportionality in which each agent's utility must be at least $1/n$ of the maximum possible value. Therefore, MmS fairness only becomes meaningful when items are indivisible.

There is a natural connection between MmS allocations and parallel machine scheduling, which we outline later. This connection turns out to be very fruitful for computing (approximate) optimal MmS allocations. In the following, we briefly introduce the concept of parallel machine scheduling in general and its necessary specifications for our work.

We have a set $\mathcal{M}$ of jobs and a set $[N]$ of N machines. Each of the jobs has to be processed exactly once on exactly one machine without preemption. Furthermore, we have a processing time matrix $P = (p_{ij})_{i,j}$ where $p_{ij} \geq 0$ indicates how long machine i requires to finish job j. If there are no further restrictions on the values of P, we deal with unrelated parallel machines. If $p_{ij} = p_{i'j}$ for all $i, i' \in [N]$ and $j \in \mathcal{M}$ then machines are considered identical.

The goal of each machine scheduling problem is to find a schedule (i.e., an allocation) that optimizes a certain objective function. The problem type we are focusing on in this paper minimizes the time where the latest machine finishes (makespan minimization) and is related to MmS allocation of chores. An extensive overview on all important machine scheduling problems is provided by Pinedo (2012).

Graham et al. (1979) established a notation for machine scheduling problems where P stands for identical machines, R for unrelated machines, and C_{max} for minimizing the makespan. According to this notation, we will use the problems P/C_{max} and R/C_{max} in this paper. Both problems are NP-hard in the strong sense but they are well investigated and plenty of research has been conducted on approximation and heuristic algorithms which we will take advantage of (Graham 1969; Haouari, Gharbi, and Jemmali 2006; Hochbaum and Shmoys 1987; Lenstra, Shmoys, and Tardos 1990).

Definitions and Basic Properties of MmS

We introduce the basic notation and definitions for our approach in this section. For a set of items $\mathcal{M}$ and a number $N \in \mathbb{N}$ of agents, let $\Pi_N(\mathcal{M})$ be the set of all N-partitions of $\mathcal{M}$ (i.e., item allocations) and let $\mathcal{P}(\mathcal{M})$ denote the power set of $\mathcal{M}$.

Definition 1. *An **instance** $I = (\mathcal{M}, [N], (v_i)_{i\in[N]})$ is defined as a tuple consisting of a set $\mathcal{M}$ of items, a set $[N]$ of N agents, and a family $(v_i : \mathcal{P}(\mathcal{M}) \to \mathbb{R})_{i\in[N]}$ of additive utility functions. The **corresponding instance to I** is defined as $-I := (\mathcal{M}, [N], (-v_i)_{i\in[N]})$.*

1. *I is a **goods instance** iff $v_i(j) \geq 0$ for all $i \in [N]$ and $j \in \mathcal{M}$ and $v_i \not\equiv 0$ for all $i \in [N]$.*
2. *I is a **chores instance** iff $v_i(j) \leq 0$ for all $i \in [N]$ and $j \in \mathcal{M}$ and $v_i \not\equiv 0$ for all $i \in [N]$.*

The set of all goods instances is denoted by $\mathcal{G}$ and the set of all chores instances is denoted by $\mathcal{C}$.

Definition 2. *Let $I = (\mathcal{M}, [N], (v_i)_{i\in[N]})$ be an instance and $i \in [N]$ be an agent.*

1. *Agent i's **Max-min Share (MmS) guarantee** for I is defined as*

$$MmS^N_{v_i}(\mathcal{M}) := \max_{(S_1,\dots,S_N)\in\Pi_N(\mathcal{M})} \min_{j\in[N]} v_i(S_j).$$

2. *Agent i's **min-Max Share (mMS) guarantee** for I is defined as*

$$mMS^N_{v_i}(\mathcal{M}) := \min_{(S_1,\dots,S_N)\in\Pi_N(\mathcal{M})} \max_{j\in[N]} v_i(S_j).$$

Definition 3. *Let $I = (\mathcal{M}, [N], (v_i)_{i\in[N]})$ be an instance and $S = (S_1, \dots, S_N) \in \Pi_N(\mathcal{M})$ be an allocation.*

1. *S is called an **MmS allocation** for I iff $v_i(S_i) \geq MmS^N_{v_i}(\mathcal{M})$ for all agents $i \in [N]$.*
2. *S is called a **perverse mMS allocation** for I iff $v_i(S_i) \leq mMS^N_{v_i}(\mathcal{M})$ for all agents $i \in [N]$.*

The concept of a perverse mMS allocation seems counter-intuitive but turns out to be helpful to obtain results on MmS allocations for corresponding instances. We can also relax the MmS and perverse mMS allocation concepts as follows.

Definition 4. *Given an instance $I = (\mathcal{M}, [N], (v_i)_{i\in[N]})$ and a constant $\lambda \geq 0$.*

1. *The **λ-max-min problem** for I is about finding an allocation $(S_1, \dots, S_N) \in \Pi_N(\mathcal{M})$ with $v_i(S_i) \geq \lambda \cdot MmS^N_{v_i}(\mathcal{M})$ for all $i \in [N]$.*
2. *The **perverse λ-min-max problem** for I is about finding an allocation $(S_1, \dots, S_N) \in \Pi_N(\mathcal{M})$ with $v_i(S_i) \leq \lambda \cdot mMS^N_{v_i}(\mathcal{M})$ for all $i \in [N]$.*

If we have an instance $I = (\mathcal{M}, [N], (v_i)_{i\in[N]})$ then we have $-MmS^N_{v_i}(\mathcal{M}) = mMS^N_{-v_i}(\mathcal{M})$ for all agents $i \in [N]$, which leads us to the following result.

Proposition 5. *Let $S = (S_1, \dots, S_N) \in \Pi_N(\mathcal{M})$ be an allocation and $\lambda \geq 0$ be arbitrary. Then S is a solution of the λ-max-min problem for I if and only if S is a solution of the perverse λ-min-max problem for the corresponding instance $-I$.*

In particular, there is an MmS allocation for I if and only if there is a perverse mMS allocation for its corresponding instance $-I$. This result shows an interesting connection between MmS and mMS when changing signs in all utility functions - but also a fundamental difference between the allocation of chores and goods since finding MmS allocations and finding perverse mMS allocations involve different objectives. This shows that an MmS algorithm for goods cannot simply be used for the chores setting (after changing signs in the utility functions) and vice-versa.

Non-Existence and Complexity of MmS

In this section, we discuss existence and non-existence examples for MmS allocations as well as complexity results for the computation of MmS allocations. We show another fundamental difference between the goods and chores setting by showing that existence and non-existence examples do not transfer straightforwardly from goods to chores and vice-versa by simply changing signs in the utility functions.

Consider a set $[3] = \{1, 2, 3\}$ of three agents and a set of twelve items (represented by pairs)

$$\mathcal{M} = \{(j, k) | j = 1, 2, 3; k = 1, 2, 3, 4\}.$$

We define matrices

$$B = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}, \quad O = \begin{pmatrix} 17 & 25 & 12 & 1 \\ 2 & 22 & 3 & 28 \\ 11 & 0 & 21 & 23 \end{pmatrix},$$

$$E^1 = \begin{pmatrix} -3 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad E^2 = \begin{pmatrix} -3 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix},$$

$$E^3 = \begin{pmatrix} -3 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

For each agent $i \in [3]$, we define her utility function by

$$u_i : \mathcal{M} \to \mathbb{R}_{\geq 0}, \quad (j, k) \mapsto 10^6 \cdot B_{jk} + 10^3 \cdot O_{jk} + E^i_{jk}.$$

Using the instance $I = (\mathcal{M}, [3], (u_i)_{i\in[3]}) \in \mathcal{G}$, we obtain the following non-existence result for MmS allocations of chores.

Proposition 6. *There is no MmS allocation for $-I$. In particular, an MmS allocation for chores may not exist.*

The instance $-I$ we use is inspired by a clever instance constructed by Procaccia and Wang (2014) to show that an MmS allocation for goods does not necessarily exist. If we denote their instance by $J = (\mathcal{M}, [3], (w_i)_{i\in[3]}) \in \mathcal{G}$, then we get the following interesting result showing that existence and non-existence examples for MmS allocations cannot be simply converted into each other by changing signs in the utility functions.

Remark 7. *There is an MmS allocation for I but no MmS allocation for $-I$. There is no MmS allocation for J but an MmS allocation for $-J$.*

This is another fundamental difference between MmS for goods and chores. Furthermore, not only do MmS allocations not exist in general, but computing an MmS allocation is also strongly NP-hard if it exists. The reduction is straightforward from Integer Partition to an allocation instance in which each agent has the same utility function.[2]

Proposition 8. *Computing an MmS allocation for chores - if it exists - is strongly NP-hard. The problem is weakly NP-hard even for two agents.*

Due to the intractability in general of computing MmS allocations for chores, in the following sections, we will develop different approximation and heuristic algorithms.

2-MmS-Approximation for Chores

The purpose of this section is to present a polynomial-time 2-MmS-approximation algorithm (Algorithm 1) for chores, i.e., where each agent is guaranteed a utility of at least twice her (negative) max-min share guarantee.

We obtain the following theorem, which shows that Algorithm 1 gives us a 2-MmS-approximation algorithm for chores.

[2]The corresponding complexity result for goods has already been proved by Bouveret and Lemaître (2016).

Algorithm 1 Greedy round-robin protocol

1: Given an arbitrary instance, agents pick in round-robin manner and are given the item with the highest utility among all remaining items at each pick.

Theorem 9. *Let $I = (\mathcal{M}, [N], (d_i)_{i\in[N]}) \in \mathcal{C}$ be a chores instance and denote the allocation obtained from Algorithm 1 by $(S_1, \dots, S_N) \in \Pi_N(\mathcal{M})$. Then we have*

$$d_i(S_i) \geq \left(2 - \frac{1}{N}\right) \cdot MmS^N_{d_i}(\mathcal{M})$$

for all $i \in [N]$ and the inequality cannot be improved for general instances.

Proof. Define $d_i^{min} = \min_{j\in\mathcal{M}} d_i(j)$. As the first step, we show

$$d_i(S_i) \geq d_i(S_{i'}) + d_i^{min} \tag{1}$$

for all $i, i' \in [N]$. This is obvious for $i \leq i'$ (picking rule and non-positivity of utilities) and therefore, we can assume $i > i'$ (i.e., i' picks before i in each round). Let K be the particular round where agent i' picks her final item and denote the pick of agent i (i' resp.) in round $k = 1, \dots, K$ by r_i^k ($r_{i'}^k$ resp.). Please note that the last pick r_i^K of agent i may be empty resulting in $d_i(r_i^K) = 0$. We have $d_i(r_i^k) \geq d_i(r_{i'}^{k+1})$ for all $k = 1, \dots, K-1$ and therefore

$$\begin{aligned}
d_i(S_i) - d_i(S_{i'}) &= \sum_{k=1}^{K} d_i(r_i^k) - d_i(r_{i'}^k) \\
&= d_i(r_i^K) - d_i(r_{i'}^K) + d_i(r_i^{K-1}) - d_i(r_{i'}^{K-1}) + \dots \\
&\dots + d_i(r_i^2) - d_i(r_{i'}^2) + d_i(r_i^1) - d_i(r_{i'}^1) \\
&\geq d_i(r_i^K) - d_i(r_{i'}^1) \geq d_i^{min}
\end{aligned}$$

with the last inequality being a consequence of having a chores instance.

By applying $\sum_{i'\in[N]}$ to both sides of equation (1), we obtain[3]

$$d_i(S_i) \geq \frac{1}{N} \cdot d_i(\mathcal{M}) + \left(1 - \frac{1}{N}\right) \cdot d_i^{min}$$

for all $i \in [N]$. This finally transforms to the theorem's inequality because we have $\frac{1}{N} \cdot d_i(\mathcal{M}) \geq MmS^N_{d_i}(\mathcal{M})$ and $d_i^{min} \geq MmS^N_{d_i}(\mathcal{M})$ in every chores setting.

To verify that the bound cannot be improved in general, consider a set $\mathcal{M} = (t_1, t_2, \dots, t_{(N-1)\cdot N+1})$ of $(N-1)\cdot N + 1$ items and let the utility function d be the same for all agents with $d(t_j) = -\frac{1}{N}$ for all $j = 1, \dots, (N-1)\cdot N$ and $d(t_{(N-1)\cdot N+1}) = -1$. □

Amanatidis et al. (2015) used a similar round robin subroutine to obtain a 2-MmS-approximation for goods but only after they allocate the most valuable goods.

[3]The summand d_i^{min} can be omitted for $i' = i$.

Optimal MmS Fairness for Chores

In this section, we introduce the new concept of *optimal MmS fairness* as a natural relaxation of *MmS fairness*.

Definition 10. *For a chores instance $I \in \mathcal{C}$, the **optimal MmS ratio** λ^I is defined as the minimal $\lambda \in [0, \infty)$ for which the λ-max-min-problem for I has a solution.*

Note that the minimum exists in this definition since for a fixed instance I, there is only a finite number of possible allocations. Based on this notation, we define the new optimal Mms fairness concept.

Definition 11. *For a chores instance $I = (\mathcal{M}, [N], (d_i)_{i\in[N]}) \in \mathcal{C}$, an **optimal MmS allocation** is an allocation $(S_1, \dots, S_N) \in \Pi_N(\mathcal{M})$ with $d_i(S_i) \geq \lambda^I \cdot MmS^N_{d_i}(\mathcal{M})$ for all $i \in [N]$.*

Optimal MmS is equivalent to maximizing the egalitarian welfare of agents when the utilities of each agent are normalized by the agent's MmS guarantee. Although optimal MmS can be viewed as combining two natural ideas from egalitarian welfare and MmS fairness, there are two main advantages to the introduced concept. First, for each specific chore instance, we can guarantee the existence of an optimal MmS allocation. Second, an optimal Mms allocation is always an MmS allocation if the latter exists. Both observations follow immediately from the definitions. We will give an introductory example for an optimal MmS allocation.

Example 12. *Define a chores instance $I = (\mathcal{M}, [2], (d_i)_{i\in[2]}) \in \mathcal{C}$ with a set $[2] = \{1, 2\}$ of two agents and a set of two items $\mathcal{M} = \{a, b\}$. We define $d_1(a) = -r$, $d_1(b) = -1$, $d_2(a) = -1$, and $d_2(b) = -r$ for some $r > 1$. Then we have $MmS^2_{d_1}(\mathcal{M}) = MmS^2_{d_2}(\mathcal{M}) = -r$ which means that $S_1 = \{a\}$ and $S_2 = \{b\}$ is an MmS allocation for I where each agent gets a total utility of $-r$. The optimal MmS allocation for I, however, is $S_1 = \{b\}$ and $S_2 = \{a\}$ giving each agent a total utility of -1. In particular, we have $\lambda^I = \frac{1}{r}$.*

This example shows that each agent's ratio of the utility in an optimal MmS allocation to the utility in an arbitrary MmS allocation can be arbitrarily small as $r > 1$ can be any real number. A natural complementary question is the worst-case for the utility in an optimal MmS allocation in comparison to the MmS guarantee. This is addressed by the following definition.

Definition 13. *The **universal MmS ratio** for chores is defined as $\lambda^- := \sup_{I\in\mathcal{C}} \lambda^I$.*

We give bounds for and a connection between the instance-dependent *optimal* and the instance-independent *universal* MmS ratio in the following. Note that we do not claim the upper bounds to be tight.

Lemma 14. *Let $I = (\mathcal{M}, [N], (d_i)_{i\in[N]}) \in \mathcal{C}$ be a chores instance. Then we have $0 \leq \lambda^I \leq 2$, $\lambda^I \leq \lambda^-$, and $1 < \lambda^- \leq 2$.*

Proof. The inequalities $0 \leq \lambda^I$ and $\lambda^I \leq \lambda^-$ hold per definition. $\lambda^I \leq 2$ follows from Theorem 9 and also implies

$\lambda^- \leq 2$ by definition. Finally, $1 < \lambda^-$ follows from Proposition 6. □

However, as we have seen in Proposition 8, the complexity of computing an MmS allocation - if it exists - is strongly NP-hard and hence the same holds true for the computation of an optimal MmS allocation. Therefore, we show in the next sections that there is a PTAS for the computation of such an allocation as long as the number of agents is fixed. Furthermore, we provide an efficient and effective solution heuristic.

Remark 15. *The concepts of this section can be formulated for goods in a similar way. For a goods instance $I \in \mathcal{G}$, the optimal MmS ratio λ^I can be defined as the maximal $\lambda \in [0, \infty]$ for which the λ-max-min-problem for I has a solution. The universal MmS ratio for goods can be defined as $\lambda^+ := \inf_{I \in \mathcal{G}} \lambda^I$ and fulfills $\frac{2}{3} \leq \lambda^+ < 1$. The inequalities follow from approximation and non-existence results for MmS allocation of goods proven by Procaccia and Wang (2014).*

Algorithms for Optimal MmS Fairness

In this section, we present an approximation algorithm for finding an optimal MmS allocation for chores (Algorithm 2) and show that the algorithm gives a PTAS when the number of agents is fixed. Finally, we develop an efficient and effective solution heuristic (Algorithm 3), which we will evaluate in computational experiments in the next section.

For a given *goods* instance $(\mathcal{M}, [N], (u_i)_{i \in [N]}) \in \mathcal{G}$, the computation of $mMS^N_{u_i}(\mathcal{M})$ for an agent $i \in [N]$ is equivalent to the computation of a job partition that minimizes the makespan on N identical parallel machines (P/C_{max}) where the processing time of a job $j \in \mathcal{M}$ is defined as $p_j := u_i(j)$ on any machine. Let now a *chores* instance $I = (\mathcal{M}, [N], (d_i)_{i \in [N]}) \in \mathcal{C}$ and $\varepsilon \geq 0$ be given. With this notation, we can formulate the following algorithm.

Algorithm 2 PTAS for optimal MmS

1: Select $\alpha, \beta \geq 0$ with $(1 + \alpha) \cdot (1 + \beta) \leq 1 + \varepsilon$.
2: Define $u_i := -d_i$ for all $i \in [N]$.
3: Compute c_i with $mMS^N_{u_i}(\mathcal{M}) \leq c_i \leq (1 + \alpha) \cdot mMS^N_{u_i}(\mathcal{M})$ for each $i \in [N]$ via the corresponding P/C_{max} problem.
4: Define new additive utility functions $u'_i : \mathcal{M} \to \mathbb{R}_{\geq 0}$ for all $i \in [N]$ by $u'_i(j) := \frac{1}{c_i} \cdot u_i(j) \quad \forall j \in \mathcal{M}$.
5: Consider the corresponding R/C_{max} problem where the processing times are defined as $p_{ij} := u'_i(j)$ for all $i \in [N]$ and $j \in \mathcal{M}$. Denote the optimal objective function value by λ^*. Compute an approximate solution $S^\varepsilon = (S^\varepsilon_1, \ldots, S^\varepsilon_N) \in \Pi_N(\mathcal{M})$ with $u'_i(S^\varepsilon_i) \leq (1 + \beta) \cdot \lambda^*$ for all $i \in [N]$.

Theorem 16. *If we apply Algorithm 2 to the pair (I, ε), then S^ε is a solution of the $(1 + \varepsilon) \cdot \lambda^I$-max-min problem for I.*

Proof. Since a solution of the λ^I-max-min problem for I exists per definition, we can conclude by Proposition 5 that a solution of the perverse λ^I-min-max problem for $-I$ exists. This implies the existence of $(S_1, \ldots, S_N) \in \Pi_N(\mathcal{M})$ with $u_i(S_i) \leq \lambda^I \cdot mMS^N_{u_i}(\mathcal{M}) \leq \lambda^I \cdot c_i$ for all $i \in [N]$. From this we have $u'_i(S_i) \leq \lambda^I$ for all $i \in [N]$ and we can conclude $\lambda^* \leq \lambda^I$.

This gives us

$$\max_{i \in [N]} \frac{u_i(S^\varepsilon_i)}{c_i} = \max_{i \in [N]} u'_i(S^\varepsilon_i) \leq (1 + \beta) \cdot \lambda^* \leq (1 + \beta) \cdot \lambda^I$$

and since $c_i \leq (1 + \alpha) \cdot mMS^N_{u_i}(\mathcal{M})$, this leads us to

$$\max_{i \in [N]} \frac{u_i(S^\varepsilon_i)}{mMS^N_{u_i}(\mathcal{M})} \leq (1 + \alpha) \cdot (1 + \beta) \cdot \lambda^I \leq (1 + \varepsilon) \cdot \lambda^I$$

which is equivalent to

$$u_i(S^\varepsilon_i) \leq (1 + \varepsilon) \cdot \lambda^I \cdot mMS^N_{u_i}(\mathcal{M})$$

for all $i \in [N]$.

This proves that S^ε is a solution of the perverse $(1 + \varepsilon) \cdot \lambda^I$-min-max problem for $-I$. The result follows now by Proposition 5. □

Remark 17. *Executing Algorithm 2 for $\varepsilon = 0$ leads to an exact algorithm for finding an optimal MmS allocation for chores.*

Hochbaum and Shmoys (1987) present a PTAS for P/C_{max} and Lenstra, Shmoys, and Tardos (1990) present a PTAS for R_N/C_{max} (which means that the number of agents is fixed to N). This implies that we can run Algorithm 2 in polynomial time for each $\varepsilon > 0$ when the number of agents is fixed and therefore gives us (in combination with Theorem 16) immediately the following important corollary.

Corollary 18. *Let the number of agents be fixed to N and let $I \in \mathcal{C}$ be a chores instance with N agents. Then Algorithm 2 provides a PTAS for the computation of an optimal MmS allocation for I.*

This is a strong result since it gives a PTAS for the computation of an optimal MmS allocation of a given chores instance, no matter if an MmS allocation exists or not (for a fixed number of agents).

Since the PTAS is only of theoretical interest due to large hidden constants, we formulate an efficient and effective solution heuristic, which we call *SCHED*, in the following. For a P/C_{max} instance, we can calculate a feasible solution by the simple and well-known *longest processing time first (LPT)* rule - see Graham (1969) for details. Furthermore, an R/C_{max} instance can be solved to optimality using an integer linear program (Lenstra, Shmoys, and Tardos 1990). This allows us to formulate Algorithm 3 for a given chores instance $I = (\mathcal{M}, [N], (d_i)_{i \in [N]}) \in \mathcal{C}$.

This algorithm differs from Algorithm 2 in two ways. First, the calculation of upper bounds in step 2 is fast (below $0.25s$ in our experiments - see the following section - with up to 128 agents and $3,200$ items) but has only a worst-case guarantee of $1 + \alpha = \frac{4}{3}$ for general instances (Graham 1969). Second, the termination criterion in step 4 is a time limit instead of an ex-ante worst-case guarantee β.

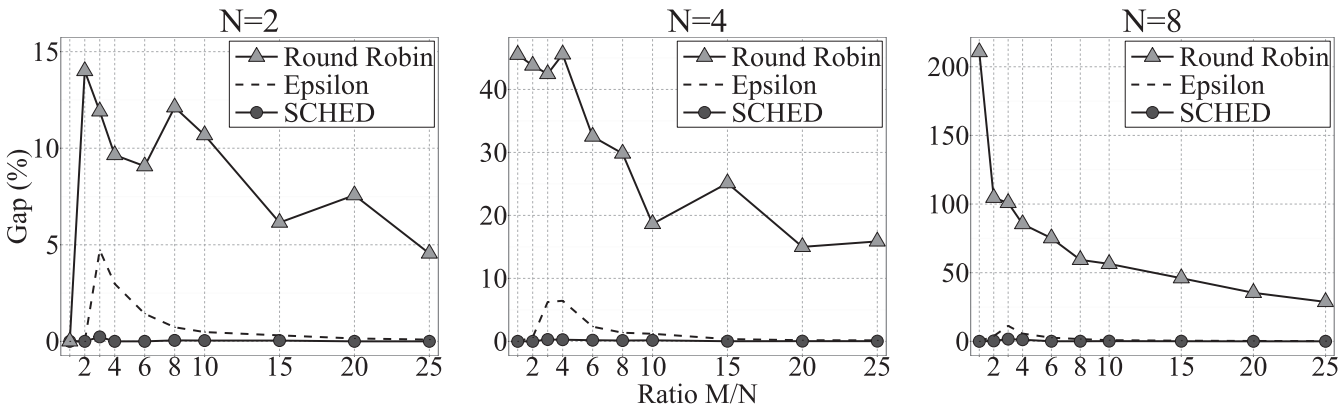

Figure 1: Gaps obtained in computational results

Algorithm 3 SCHED heuristic for optimal MmS

1: Define $u_i := -d_i$ for all $i \in [N]$.
2: Calculate an upper bound c_i on $mMS^N_{u_i}(\mathcal{M})$ for each $i \in [N]$ by applying LPT to the corresponding P/C_{max} instance.
3: Define new additive utility functions $u'_i : \mathcal{M} \to \mathbb{R}_{\geq 0}$ for all $i \in [N]$ by $u'_i(j) := \frac{1}{c_i} \cdot u_i(j) \quad \forall j \in \mathcal{M}$.
4: Consider the corresponding R/C_{max} problem where the processing times are defined as $p_{ij} := u'_i(j)$ for all $i \in [N]$ and $j \in \mathcal{M}$. Solve the instance to optimality with an integer programming solver and abort calculations after a pre-set time limit T_{max}. Denote the best incumbent solution found within the time limit by $\widetilde{S} = (\widetilde{S_1}, \ldots, \widetilde{S_N}) \in \Pi_N(\mathcal{M})$.

Although not having an ex-ante worst-case guarantee for SCHED, we can give an ex-post worst-case analysis in the following way. In addition to an upper bound (step 2), we can calculate a lower bound on $mMS^N_{u_i}(\mathcal{M})$ via the corresponding P/C_{max} instance - see Haouari, Gharbi, and Jemmali (2006) for details. We denote the maximum relative gap between upper and lower bounds among all agents by α and the relative gap between the current best incumbent solution and the current best lower bound of the R/C_{max} problem in step 4 (reported after T_{max} by the integer programming solver) by β. From Theorem 16, we get the following result.

Remark 19. *Apply Algorithm 3 to a chores instance $I \in \mathcal{C}$ and set $\varepsilon = (1+\alpha) \cdot (1+\beta) - 1$. Then $\widetilde{S}$ is a solution of the $(1+\varepsilon) \cdot \lambda^I$-max-min problem for I.*

Computational Experiments

In this section, we report the results of our computational study where we tested the performance of our SCHED heuristic (Algorithm 3). We show that SCHED returns near-optimal solutions and greatly outperforms the simpler greedy round-robin protocol (Algorithm 1).

We coded all algorithms in C++ on a Linux CentOS 7 based 12-core processor with a clock speed of 3.07 GHz and 24 GiB memory. Integer linear programs were solved via the Gurobi 6 C++ API. We tested different instance sizes with a varying number of agents $N \in \{2, 4, 8, 16, 32, 64, 128\}$ and items $M \in \{N, 2N, 3N, 4N, 6N, 8N, 10N, 15N, 20N, 25N\}$ by generating 10 instances per instance size and averaging the results. The time limit in step 4 of SCHED was set to $T_{max} = 300s$. Utilities were drawn from a uniform distribution $u_i(j) \sim U(0, 100)$ (and negated for obtaining chores instances) as it is common in the literature for fair division of goods (Amanatidis et al. 2015; Bouveret and Lemaître 2016; Kurokawa, Procaccia, and Wang 2016).

We were able to calculate optimal MmS ratios for instance sizes with $N \in \{2, 4, 8\}$ using Algorithm 2 (see Remark 17). We find that the optimal MmS ratios are decreasing with both an increasing number of agents and an increasing ratio of items to agents and they are varying between 0.82 and 0.67 for 2 agents, between 0.54 and 0.40 for 4 agents, and between 0.34 and 0.23 for 8 agents. In particular, an MmS allocation exists in all of these instances.

For these instance sizes with $N \in \{2, 4, 8\}$, Figure 1 reports the relative gaps of the achieved MmS ratio from the optimal MmS ratio[4] using the SCHED heuristic and the greedy round-robin protocol. Note that these gaps are only available when optimal MmS ratios are available. We obtain that the SCHED heuristic performs very close to the optimum with a maximum average deviation of 1.64% (observed for 8 agents and 24 items). The observed gaps for the greedy round-robin protocol, however, are much higher.

The worst-case gaps for our SCHED heuristic are represented by the *Epsilon* lines and can be calculated without knowing the optimal MmS ratios (see Remark 19). We find that the observed gaps for SCHED are clearly below the worst-case gaps for SCHED and that these worst-case gaps for SCHED are in turn much smaller than the observed gaps for the greedy round-robin protocol.

We were not able to calculate optimal MmS ratios for instance sizes with $N \geq 16$. Therefore, the only value we

[4]For example, if the optimal MmS ratio is 0.5 and the achieved MmS ratio is 0.6, then the relative gap is $\frac{0.6-0.5}{0.5} = 0.2 = 20\%$.

can benchmark our SCHED heuristic with is the worst-case gap. Our experiments show that the maximum worst-case gap (which always occurs for $M = 3N$) stays almost constant when N is further increased - the value is 11.28% for $N = 8$ and 11.92% for $N = 128$. Therefore, good solutions are also guaranteed for instance sizes with $N \geq 16$.

We can conclude that our SCHED heuristic clearly outperforms the greedy round-robin protocol,[5] returns solutions which are very close to the optimal MmS ratio for instances with a small number of agents, and has a good worst-case performance for all remaining test instances. Based on results for $N \leq 8$, we can also expect the SCHED solutions to clearly beat these worst-case guarantees for higher N.

Conclusions

We initiated work on MmS allocation of chores and presented interesting connections and differences between fair allocation of goods and chores. We showed that an MmS allocation for chores does not need to exist and that computing an MmS allocation is NP-hard in the strong sense if it exists. Consequently, we developed a polynomial-time greedy round-robin 2-MmS-approximation algorithm and a new fairness concept called optimal MmS. For a fixed number of agents, we proposed a PTAS for finding optimal MmS allocations. We developed an efficient and effective solution heuristic (with an ex-post worst-case analysis) which finds near-optimal solutions within short time in our computational experiments for up to 128 agents and 3,200 chores and greatly outperforms the simpler greedy round-robin 2-approximation algorithm.

References

Amanatidis, G.; Markakis, E.; Nikzad, A.; and Saberi, A. 2015. Approximation algorithms for computing maximin share allocations. In *Proceedings of the 35th International Colloquium on Automata, Languages, and Programming (ICALP)*, 39–51. Springer.

Aziz, H.; Gaspers, S.; Mackenzie, S.; and Walsh, T. 2015. Fair assignment of indivisible objects under ordinal preferences. *Artificial Intelligence* 227:71–92.

Bogomolnaia, A., and Moulin, H. 2016. Envy-free division of bads: a difficulty.

Bouveret, S., and Lemaître, M. 2016. Characterizing conflicts in fair division of indivisible goods using a scale of criteria. *Autonomous Agents and Multi-Agent Systems* 30(2):259–290.

Bouveret, S.; Chevaleyre, Y.; and Maudet, N. 2015. Fair allocation of indivisible goods. In Brandt, F.; Conitzer, V.; Endriss, U.; Lang, J.; and Procaccia, A. D., eds., *Handbook of Computational Social Choice*. Cambridge University Press.

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

[5] Almost identical results are obtained for a modification where the picking order is reversed in each round.

Budish, E. 2011. The combinatorial assignment problem: Approximate competitive equilibrium from equal incomes. *Journal of Political Economy* 119(6):1061–1103.

Caragiannis, I.; Kaklamanis, C.; Kanellopoulos, P.; and Kyropoulou, M. 2012. The efficiency of fair division. *Theory of Computing Systems* 50(4):589–610.

Goldman, J., and Procaccia, A. D. 2014. Spliddit: Unleashing fair division algorithms. *ACM SIGecom Exchanges* 13(2):41–46.

Graham, R. L.; Lawler, E. L.; Lenstra, J. K.; and Kan, A. H. G. R. 1979. Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics* 5:287–326.

Graham, R. L. 1969. Bounds on multiprocessing timing anomalies. *SIAM journal on Applied Mathematics* 17(2):416–429.

Haouari, M.; Gharbi, A.; and Jemmali, M. 2006. Tight bounds for the identical parallel machine scheduling problem. *International Transactions in Operational Research* 13(6):529–548.

Hochbaum, D. S., and Shmoys, D. B. 1987. Using dual approximation algorithms for scheduling problems theoretical and practical results. *Journal of the ACM (JACM)* 34(1):144–162.

Kurokawa, D.; Procaccia, A. D.; and Wang, J. 2016. When can the maximin share guarantee be guaranteed? In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI)*. AAAI Press.

Lenstra, J. K.; Shmoys, D. B.; and Tardos, E. 1990. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical programming* 46(1-3):259–271.

Lipton, R. J.; Markakis, E.; Mossel, E.; and Saberi, A. 2004. On approximately fair allocations of indivisible goods. In *Proceedings of the 5th ACM Conference on Electronic Commerce (ACM-EC)*, 125–131. ACM Press.

Pinedo, M. L. 2012. *Scheduling: Theory, Algorithms, and Systems*, volume 4. Springer.

Procaccia, A. D., and Wang, J. 2014. Fair enough: Guaranteeing approximate maximin shares. In *Proceedings of the 15th ACM Conference on Economics and Computation (ACM-EC)*, 675–692. ACM Press.