Approximate Pareto Set for Fair and Efficient Allocation: Few Agent Types or Few Resource Types

Trung Thanh Nguyen[1]* and **Jörg Rothe**[2]

[1]ORLab, Faculty of Computer Science, Phenikaa University, Hanoi 12116, Vietnam
[2]Institut für Informatik, Heinrich-Heine-Universität Düsseldorf, 40225 Düsseldorf, Germany
thanh.nguyentrung@phenikaa-uni.edu.vn, rothe@hhu.de

Abstract

In fair division of indivisible goods, finding an allocation that satisfies fairness and efficiency simultaneously is highly desired but computationally hard. We solve this problem approximately in polynomial time by modeling it as a bi-criteria optimization problem that can be solved efficiently by determining an approximate Pareto set of bounded size. We focus on two criteria: max-min fairness and utilitarian efficiency, and study this problem for the setting when there are only a few item types or a few agent types. We show in both cases that one can construct an approximate Pareto set in time polynomial in the input size, either by designing a dynamic programming scheme, or a linear-programming algorithm. Our techniques strengthen known methods and can be potentially applied to other notions of fairness and efficiency as well.

1 Introduction

Fair division with its wide range of important applications has a long research history, see the book chapters by Bouveret *et al.* [2016] and Lang and Rothe [2015] for an overview. In this setting, we are given n agents and m indivisible resources (or items or goods), and agents express their preferences over subsets (or bundles) of items by individual value functions. One of the central tasks is to fairly allocate items to agents in an efficient manner, as motivated by real-world applications such as vehicle-request assignment in ridesourcing [Lesmana *et al.*, 2019], bandwidth allocation [Goel *et al.*, 2001], and load balancing [Kleinberg *et al.*, 2001].

Various axiomatically justified notions of fairness have been studied, including *envy-freeness* (no one wants to swap her bundle with others), *proportionality* (every agent receives a value worth at least a fraction of $1/n$ of her value for the whole set of items), and *max-min fairness* (maximizing the value of the worst-off agent, i.e., egalitarian social welfare).

Widely used notions of efficiency include *utilitarian efficiency* (allocations that maximize the sum of the agents' values, i.e., utilitarian social welfare) and *Pareto optimality* (allocations in which no agent can increase her value without decreasing the value of some other agent). Pareto optimality is necessary but not sufficient for utilitarian efficiency. We will focus on max-min fairness and utilitarian efficiency.

It is unfortunate that, due to the conflict of the two criteria, allocations that are both fair and efficient do not always exist. This has motivated further research, such as to study the efficiency loss under fair allocations, which is well captured by the notion of *price of fairness* introduced and studied by Bertsimas *et al.* [2011] and Caragiannis *et al.* [2012].

Another remarkable research direction is to look for allocations that satisfy fairness and efficiency only to some extent [Lesmana *et al.*, 2019; Aziz *et al.*, 2019]. For example, Lesmana *et al.* [2019] study this issue in the context of ridesourcing request assignments where a set of requests need to be matched to a set of available vehicles based on a given weighted bipartite graph. They present an efficient algorithm for finding an allocation with any desired fairness and *bounded* efficiency. For general graphs, however, achieving such a result is well-known to be computationally hard.

This raises the question of whether one can find in polynomial time good allocations with only a small loss on both fairness and efficiency. Our goal is to answer this question by focusing on two special cases: when there are only a few agent types or a few item types. These two parameters have been the subject of intensive investigations (see, e.g., [Brânzei *et al.*, 2016; Bouveret *et al.*, 2017; Jansen and Maack, 2019; Galil and Megiddo, 1979; Brown, 1979; Krysta *et al.*, 2013; Bredereck *et al.*, 2019]).

We model the problem of computing fair and efficient allocations as a general bi-criteria optimization problem and identify the set of Pareto-optimal solutions from the perspective of approximation algorithms—a similar idea was used by Escoffier *et al.* [2013]. That is, we study an *approximation* of this set, called ε-*Pareto set*, which includes approximately non-Pareto-dominated solutions: For every feasible solution π, there is a solution π' in the ε-Pareto set that is better than π within a factor of $1-\varepsilon$ on both criteria. The purpose of constructing an approximate Pareto set is to help the decision maker to easily find an allocation satisfying any specific level of the desired quality of efficiency and fairness.

Our contribution. For any fixed constant $\varepsilon > 0$, we propose polynomial-time algorithms for generating an ε-Pareto set, for both studied cases. When the number of item types is

*Contact Author

constant, we propose a dynamic programming algorithm that runs in time polynomial in the size of the input. Interestingly, our dynamic programming technique can also be applied to (exactly) compute allocations that are Pareto-optimal and fair, w.r.t. other notions of fairness. Notably, the technique can be employed to solve the open problem of finding a min-max allocation, as stated by Jansen and Maack [2019]. For the case when the number of agent types is constant, we give an approximation scheme whose running time is polynomial in the size of the instance. Our source of improvement in the scheme is a novel technique where we significantly extend the technique used by Jansen and Maack [2019]. In particular, by a more involved procedure involving finding a basic solution of nice structure to a mixed integer linear program, we can round it to an integral solution with only a small loss on both fairness and efficiency. We believe that this idea is universally applicable and might be applied to more general classes of objective functions.

Related work. Finding fair and efficient allocations has gained a lot of interest in the last decade—see, e.g., the work of Aumann and Dombb [2015], Bliem *et al.* [2016], Caragiannis *et al.* [2016], Bei *et al.* [2019], Suksompong [2019], and Igarashi and Peters [2019]. Bredereck *et al.* [2019] presented fixed-parameter tractable algorithms for computing envy-free and Pareto-optimal allocations, when parameterized by the number of agents and the maximum value of any item. Their result also holds for various fairness notions including max-min fairness. Barman *et al.* [2018] showed that one can produce an allocation that is envy-free up to one item and Pareto-optimal in pseudo polynomial time. Aziz *et al.* [2019] presented an algorithm for finding an envy-free up to one item allocation whose utilitarian social welfare satisfies a given efficiency goal. Considering max-min fairness and utilitarian efficiency, Lesmana *et al.* [2019] obtained an approximation algorithm for a restricted setting in which every allocation forms a perfect matching on a bipartite graph. Since our focus is on those cases when the number of item types or the number of agent types are constant, we will give a short overview of related work. Firstly, the case of one item type was shown to be tractable for a class of single-objective optimization problems, including for utilitarian social welfare maximization (even for concave value functions) by Galil and Megiddo [1979], and for egalitarian social welfare maximization by Brown [1979]. It still remains open if such results can be extended to the case of few item types. A similar question has been posed by Jansen and Maack [2019] in the context of machine scheduling, where the objective is to minimize the maximum value among all agents. In the same paper, the authors presented a polynomial-time approximation scheme (PTAS) for the problem of maximizing egalitarian social welfare, assuming that the number of different types of agents is fixed. This result significantly improved a previous PTAS designed for the special case when all the agents' valuations are identical [Woeginger, 1997]. Very recently, Kones and Levin [2019] proposed a framework for designing PTASes for more general classes of machine scheduling problems, which generalizes the results of Jansen and Maack [2019]. Nevertheless, their technique fails when applied to our problems.

2 Model and Notation

An *instance of a fair and efficient allocation problem* is often given by a triple $\mathcal{I} = (\mathcal{A}, \mathcal{O}, \mathcal{U})$, where $\mathcal{A} = \{1, 2, \ldots, n\}$ is the set of agents, $\mathcal{O} = \{1, 2, \ldots, m\}$ the set of items, and $\mathcal{U} = \{u_1, u_2, \ldots, u_n\}$ the agents' valuation functions expressing their utilities over the set of items. We restrict our attention to the class of additive valuation functions which have been widely used in the context of resource allocation. A valuation function is *additive* if for any subset (bundle) $S \subseteq \mathcal{O}$, $u_i(S) = \sum_{j \in S} u_{ij}$, where u_{ij} is agent i's nonnegative integer value for item j. We assume that the value of the empty bundle is zero for every agent. Furthermore, by scaling if needed we can assume that if $u_{ij} \neq 0$ then $u_{ij} \geq 1$, for all i, j. For ease of presentation, define $u_{\max} = \max_{i,j} u_{ij}$. An *allocation* of items among agents is a partition of the set $\mathcal{O}$ of items into n disjoint subsets. We denote an allocation by $\pi = (\pi_1, \ldots, \pi_n)$, where π_i is the bundle allocated to agent i.

Agent types and item types. The set of agents is divided into T groups $\mathcal{A}_1, \ldots, \mathcal{A}_T$ such that agents in each group have the same valuation function. Let $[T] = \{1, \ldots, T\}$. Let $n_t = |\mathcal{A}_t|$ be the number of agents of type t, and $u_{t,j}$ be the value of item j for any agent in $\mathcal{A}_t$. Let $u_{t,\mathcal{O}} = \sum_{j \in \mathcal{O}} u_{t,j}$ for $t \in [T]$. We say that two items j, j' are of the same type if they have the same value for every agent i, i.e., $u_{ij} = u_{ij'}$. Hence, we can divide the item set $\mathcal{O}$ into q classes such that items in each class are of same type. We assume that there are m_j (identical) items for each item type j.

Max-min fairness and other fairness notions. The *egalitarian social welfare* (ESW) of an allocation π is defined as $\mathsf{sw}_e(\pi, \mathcal{I}) = \min_{i=1}^{n} u_i(\pi_i)$. If π has maximum egalitarian social welfare among all allocations, we say that π is *max-min fair* (MMF). An allocation is *proportional fair* (PF) if every agent i gets a value of at least her proportional share of $u_i(\mathcal{O})/n$, and is *maximin-share fair* (MSF) if every agent gets a value of at least her share when she can partition $\mathcal{O}$ into n bundles and then has to pick a bundle of smallest value.

Utilitarian efficiency. The *utilitarian social welfare* (USW) of an allocation π is defined as $\mathsf{sw}_u(\pi, \mathcal{I}) = \sum_{i=1}^{n} u_i(\pi_i)$. If π has maximum utilitarian social welfare among all allocations, π is said to be *utilitarian efficient* (UE). Note that if an allocation is utilitarian efficient then it is Pareto-optimal, but not vice versa. We write $\mathsf{sw}_u(\pi)$ and $\mathsf{sw}_e(\pi)$ if $\mathcal{I}$ is clear from the context. We now define Pareto set and its approximation.

Definition 1 (Pareto set and ε-Pareto set)**.** *Given an instance $\mathcal{I}$, we say that an allocation π dominates an allocation π' if $\mathsf{sw}_e(\pi) \geq \mathsf{sw}_e(\pi')$ and $\mathsf{sw}_u(\pi) \geq \mathsf{sw}_u(\pi')$, with at least one inequality being strict. The* Pareto set of $\mathcal{I}$*, denoted by $\mathcal{P}(\mathcal{I})$, is the set of all allocations that are not dominated by any other allocation. For $\varepsilon \in (0, 1]$, the* ε-Pareto set of $\mathcal{I}$*, denoted by $\mathcal{P}_\varepsilon(\mathcal{I})$, is the set of allocations such that for all allocations π', there is always an allocation $\pi \in \mathcal{P}_\varepsilon(\mathcal{I})$ such that $\mathsf{sw}_e(\pi) \geq (1-\varepsilon)\mathsf{sw}_e(\pi')$ and $\mathsf{sw}_u(\pi) \geq (1-\varepsilon)\mathsf{sw}_u(\pi')$.*

We will address the problem of constructing an ε-Pareto set $\mathcal{P}_\varepsilon(\mathcal{I})$ of bounded size for any given instance $\mathcal{I}$, and for any fixed constant $\varepsilon > 0$, when the number of item types or the number of agent types are not part of the input.

3 Few Item Types

We present a dynamic programming scheme that is the key ingredient in the construction of an approximate Pareto set. It can also be used as a subroutine to solve other problems.

Lemma 1. *Given an instance $\mathcal{I}$ and a value β, one can compute in polynomial time an allocation of maximum utilitarian social welfare subject to the constraint that the value of every agent is at least β, if such an allocation exists, assuming that the number of item types is constant.*

Proof. Given $(\mathcal{I},\beta)$, we present a dynamic programming scheme, denoted by $\mathsf{DP}(\mathcal{I},\beta)$, for finding a required allocation in terms of a state-transition diagram. A state is defined as a $(q+1)$-tuple of the form $S=(i,s_1,s_2,\ldots,s_q)$, where $i\in\{0\}\cup[n]$ and $s_j\in\{0\}\cup[m_j]$, for $j\in[q]$. States are grouped into $n+1$ blocks $B_0,B_1,\ldots,B_n$. The first and the last blocks, B_0 and B_n, contain only one state denoted by $S_{start}=(0,m_1,m_2,\ldots,m_q)$ and $S_{end}=(n,0,0,\ldots,0)$, respectively. Block B_i, for $i\in[n-1]$, contains all possible $(q+1)$-tuples S ensuring that the size of B_i is $(m_1+1)\cdots(m_q+1)$. Intuitively, a state $s\in B_i$ gives us partial information about an allocation of items to agents. In particular, it indicates that agents $1,\ldots,i$ have been allocated items (but we do not know which items are assigned to which agents), and $s_1,s_2,\ldots,s_q$ are the number of items of each type that are unallocated yet at the current state and will be assigned to the remaining agents $i+1,\ldots,n$ at the next states. The total number of states in $n+1$ blocks is $(n-1)(m_1+1)\cdots(m_q+1)+2$. We next define transitions of states between two consecutive blocks as follows. For each $i\in[n]$, there is a transition from a state $S=(i-1,s_1,s_2,\ldots,s_q)\in B_{i-1}$ to a state $S'=(i,s'_1,s'_2,\ldots,s'_q)\in B_i$ if and only if $s_j-s'_j\geq 0$ holds for all $j\in[q]$, and we set the value of this transition to $w=\sum_{j=1}^{q}u_{ij}(s_j-s'_j)$ if $w\geq\beta$ and $w=-\infty$, otherwise. By this transition we mean that agent i is allocated a bundle of items encoded by the vector $(s_1-s'_1,\ldots,s_q-s'_q)$.

One can see that a (simple) path of the constructed state-transition diagram (a directed acyclic graph) above corresponds to an allocation for $\mathcal{I}$, and vice versa. Our problem is therefore equivalent to computing a longest path (i.e., the path of maximum total value) in the diagram, which can be done via dynamic programming. If the path is of positive value, $\mathsf{DP}(\mathcal{I},\beta)$ returns a corresponding allocation π satisfying the conditions in the lemma, otherwise it returns "NO," meaning that there is no such allocation. Regarding the running time of DP, note that the number of states is bounded by $O(nm^q)$, and for each state we have at most m^q outgoing transitions. Therefore, the overall complexity of DP is $O(nm^{O(q)})$, which is polynomial in n and m, as q is constant. □

Using Lemma 1 we can now prove the following theorem.

Theorem 1. *Given an instance $\mathcal{I}$, one can compute in polynomial time a set $\mathcal{P}_\varepsilon(\mathcal{I})$ for any fixed constant $\varepsilon\in(0,1]$, when the number of item types is constant.*

Proof. Let $\mathcal{I}$ be an instance and ε be a constant in $(0,1]$. To prove the claim in the theorem, we describe a polynomial-time algorithm as follows. The idea is to enumerate all possible lower bounds *LB* on the egalitarian social welfare of an allocation, that is, $LB\in\{0,1,1+\varepsilon,(1+\varepsilon)^2,\ldots,(1+\varepsilon)^k\}$, where $k=\lfloor\log_{1+\varepsilon}\{\min_i u_i(\mathcal{O})\}\rfloor$. For each value of *LB*, we run the dynamic program $\mathsf{DP}(\mathcal{I},LB)$ from Lemma 1 to return an allocation π of maximum utilitarian social welfare (if there exists one), respecting the constraint on lower bound *LB*.

We include such an allocation π into $\mathcal{P}_\varepsilon(\mathcal{I})$, for every *LB*. One can see that $|\mathcal{P}_\varepsilon(\mathcal{I})|\leq k+2$. This together with the polynomial running time of $\mathsf{DP}(\mathcal{I},LB)$ implies that our algorithm runs in time polynomial in the size of the instance.

We now prove that for every allocation π, there is always an allocation $\pi'\in\mathcal{P}_\varepsilon(\mathcal{I})$ such that $\mathsf{sw}_e(\pi')\geq(1-\varepsilon)\mathsf{sw}_e(\pi)$ and $\mathsf{sw}_u(\pi')\geq(1-\varepsilon)\mathsf{sw}_u(\pi)$. Indeed, the first case $\mathsf{sw}_e(\pi)=0$ is easy to see because the allocation π' returned by $\mathsf{DP}(\mathcal{I},0)$ is UE. Suppose that $\mathsf{sw}_e(\pi)\neq 0$. Let ℓ be a nonnegative integer such that $\mathsf{sw}_e(\pi)\in[(1+\varepsilon)^\ell,(1+\varepsilon)^{\ell+1})$. Hence, π must correspond to some path in the diagram considered by $\mathsf{DP}(\mathcal{I},(1+\varepsilon)^\ell)$. Let π' be the allocation returned by $\mathsf{DP}(\mathcal{I},(1+\varepsilon)^\ell)$. It must then hold that $\mathsf{sw}_u(\pi)\leq\mathsf{sw}_u(\pi')$. Moreover, from $\mathsf{sw}_e(\pi)<(1+\varepsilon)^{\ell+1}$ and $\mathsf{sw}_e(\pi')\geq(1+\varepsilon)^\ell$ and as $\varepsilon\in(0,1]$, we have that $\mathsf{sw}_e(\pi')>(1+\varepsilon)^{-1}\mathsf{sw}_e(\pi)$ or $\mathsf{sw}_e(\pi')>(1-\varepsilon)\mathsf{sw}_e(\pi)$. □

The dynamic programming above can be applied to compute other fair and Pareto-optimal allocations as shown in the theorem below. The proof is omitted due to space limitations and can be found in the full version of the paper.

Theorem 2. *Given that the number of item types is constant, one can compute in polynomial time a fair and Pareto-optimal allocation (if there exists one), w.r.t. either one of the following fairness criteria:* MMF*,* PF*, and* MSF*. Also, computing a min-max allocation can be done in polynomial time.*

4 Few Agent Types

In this section, we present a linear-programming algorithm for constructing an ε-Pareto set for an instance $\mathcal{I}$ where the number of different agents is small. We make use of some of the ideas from the PTAS for finding max-min fair allocations by Jansen and Maack [2019].

Theorem 3. *For any constant $\varepsilon\in(0,1]$, one can construct an ε-Pareto set in time polynomial in the size of the instance, when the number of agent types is constant.*

Proof. Fix $\varepsilon\in(0,1]$ and let $\alpha=\varepsilon/5$ and $\tau=(1-\alpha)/2$, Algorithm 1 computes an ε-Pareto set for $\mathcal{I}$. The lemma below indicates that it suffices to focus on allocations of some special structure. The proof, which is omitted here due to space limitations, is similar to that of Alon *et al.* [1998].

Lemma 2. *Given an allocation π, one can find an allocation π' such that: i) $\mathsf{sw}_u(\pi',\mathcal{I})=\mathsf{sw}_u(\pi,\mathcal{I})$ and $\mathsf{sw}_e(\pi',\mathcal{I})\geq\mathsf{sw}_e(\pi,\mathcal{I})$; ii) for each agent type t, there is a value ξ_t such that: if there is some agent who receives an item of value $\geq\xi_t$ then it is the only one she received. Moreover, every agent who did not get any such item has a value $\in(\frac{1}{2}\xi_t,2\xi_t)$.*

The construction of an ε-Pareto set is described in Algorithm 1. The basic idea is to enumerate a bounded number of possible values of ξ_t and the corresponding egalitarian social welfare $\theta_t\in(\frac{1}{2}\xi_t,2\xi_t)$, for every $t\in[T]$, and try to find (if

Algorithm 1 PARETO-SET

1: Let $\bar{\pi}$ be an allocation maximizing $\mathsf{sw}_u(\bar{\pi},\mathcal{I})$
2: $\mathcal{P}_\varepsilon(\mathcal{I}) \leftarrow \{\bar{\pi}\}$; Define $\mathcal{X}_t$ for $t \in [T]$
3: **for** each $\xi = (\xi_1,\ldots,\xi_T) \in \mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_T$ **do**
4: $\quad W \leftarrow 2(2+\log_{1-\alpha}\alpha^2)\cdot T$; $d \leftarrow \lceil W/\alpha \rceil$
5: $\quad \mathcal{Q} \leftarrow \{\{\mathcal{D}_1,\ldots,\mathcal{D}_T\} |\ \mathcal{D}_t \subseteq \mathcal{O}, |\mathcal{D}_t| \le d, t \in [T]\}$
6: $\quad$ Define Θ_t for $t \in [T]$
7: $\quad$ **for** each valid $\mathcal{D} = \{\mathcal{D}_1,\ldots,\mathcal{D}_T\} \in \mathcal{Q}$ **do**
8: $\quad\quad$ **for** each $\theta \in \Theta = \Theta_1 \times \cdots \times \Theta_T$ **do**
9: $\quad\quad\quad$ Round item values
10: $\quad\quad\quad$ Compute an optimal BFS of $\mathsf{MILP}[\xi,\theta]$, denoted by $(\mathbf{x}^*,\mathbf{y}^*)$ (if there exists one)
11: $\quad\quad\quad$ Round $(\mathbf{x}^*,\mathbf{y}^*)$ to an integer solution $(\bar{\mathbf{x}}^*,\mathbf{y}^*)$ via solving a totally unimodular LP
12: $\quad\quad\quad$ Convert $(\bar{\mathbf{x}}^*,\mathbf{y}^*)$ into π^* using Algorithm 2
13: $\quad\quad\quad$ $\mathcal{P}_\varepsilon(\mathcal{I}) \leftarrow \mathcal{P}_\varepsilon(\mathcal{I}) \cup \{\pi^*\}$
14: Return $\mathcal{P}_\varepsilon(\mathcal{I})$

there exists one) an allocation of maximum utilitarian social welfare, w.r.t. these values.

Enumerating possible values regarding USW. For each $t \in [T]$, consider a set $\mathcal{X}_t = \{1, 1+\alpha, \ldots, (1+\alpha)^{\lfloor \log_{1+\alpha} u_{t,\mathcal{O}} \rfloor}\}$. Each $\xi_t \in \mathcal{X}_t$ represents a possible (approximate) average value of agents in a subset $A \subseteq \mathcal{A}_t$ such that every agent in $\mathcal{A}_t \setminus A$ gets only one item of value at least ξ_t. For each vector $\xi = (\xi_1,\ldots,\xi_T)$, Algorithm 1 performs Steps 4–13, which are described in detail below.

Classifying items into groups. For each agent type t, we classify items in $\mathcal{O}$ into *large items* (having value bigger than $\alpha^2\xi_t$ and less than ξ_t), *small items* (having value at most $\alpha^2\xi_t$), and *huge items* (having value at least ξ_t). Next, we guess at most $d = \lceil W/\alpha \rceil$ huge items of value at least ξ_t that are assigned to agents of this type (W is some constant that is specified in line 5 of Algorithm 1 and Lemma 3). Let $\mathcal{D}_t$ be the set of these guessed items. We call $\mathcal{D} = \{\mathcal{D}_t\}_{t\in[T]}$ a *valid* guess if $\{\mathcal{D}_t\}_{t\in[T]}$ are disjoint subsets. Given a valid guess $\mathcal{D}$, huge items that are not in $\mathcal{D}_t$ and have values bigger than $\min_{j\in\mathcal{D}_t} u_{t,j}$ should not be allocated to agents of type t.

Enumerating possible values regarding ESW. For each $t \in [T]$, given a value ξ_t, the algorithm considers a set Θ_t of possible values of the minimum value over all agents of type t in the domain $(\tau\xi_t, \xi_t/\tau)$, $\Theta_t = \{\tau\xi_t, \frac{\tau\xi_t}{1-\alpha}, \ldots, \frac{\tau\xi_t}{(1-\alpha)^p}\}$, where $p = \lfloor \log_{1-\alpha}\tau^2 \rfloor$. It is argued later that it is not necessary to consider values that do not belong to this domain.

Rounding item values (line 10 of Algorithm 1). The value of item j w.r.t. agents of type $t \in [T]$ is rounded down to

$$\hat{u}_{t,j} = (1-\alpha)^{\lceil \log_{1-\alpha}\left(u_{t,j}/\alpha^2\xi_t\right)\rceil} \cdot \alpha^2\xi_t. \tag{1}$$

Observation 1. *By the rounding step (1) we get a new instance $\hat{\mathcal{I}} = (\mathcal{A},\mathcal{O},\hat{\mathcal{U}})$ with the same agents and items but different valuation functions. Moreover, for every t and j, it holds that $\hat{u}_{t,j} \ge (1-\alpha)\cdot u_{t,j}$.*

For every $t \in [T]$, let $\mathcal{H}_t$, $\mathcal{L}_t$, and $\mathcal{S}_t$, respectively, be the sets of huge, large, and small rounded item values. We bound the size of $\mathcal{L}_t$ as follows. Since every large item has a value less than ξ_t, from (1) it follows that the number of large rounded item values (w.r.t. agent type t) must be at most $\log_{1-\alpha}\alpha^2$.

Configurations. For $t \in [T]$, we encode a bundle of items of values in $\mathcal{L}_t$ as an $|\mathcal{L}_t|$-dimensional vector $\kappa = (\kappa_v)_{v\in\mathcal{L}_t}$, called a *configuration*, where κ_v denotes the number of items of value v. The value of a configuration (or a bundle) κ for an agent of type t is defined as $\hat{u}_{t,\kappa} = \sum_{v\in\mathcal{L}_t} v\cdot\kappa_v$. For a value $\theta_t \in \Theta_t$, define

$$\begin{aligned} \mathcal{C}^t_{\ge\theta_t} &= \{\kappa = (\kappa_v)_{v\in\mathcal{L}_t} |\ \theta_t \le \hat{u}_{t,\kappa} < \xi_t/\tau\} \quad \text{and} \\ \mathcal{C}^t_{<\theta_t} &= \{\kappa = (\kappa_v)_{v\in\mathcal{L}_t} |\ \hat{u}_{t,\kappa} < \theta_t\}. \end{aligned}$$

We call $\mathcal{C}^t_{<\theta_t}$ and $\mathcal{C}^t_{\ge\theta_t}$ the sets of small and big configurations, respectively. Let $\mathcal{C}^t = \mathcal{C}^t_{\ge\theta_t} \cup \mathcal{C}^t_{<\theta_t}$. We show that $|\mathcal{C}^t|$ is bounded by a constant. Indeed, by the fact that the value of every large item is bigger than $\alpha^2\xi_t$ and the value of every configuration is less than ξ_t/τ, it follows that $\alpha^2\xi_t \cdot \sum_{v\in\mathcal{L}_t}\kappa_v \le \sum_{v\in\mathcal{L}_t} v\cdot\kappa_v < \xi_t/\tau$. Therefore, $\sum_{v\in\mathcal{L}_t}\kappa_v \le 1/(\tau\alpha^2)$. This together with the claim that $|\mathcal{L}_t| \le \log_{1-\alpha}\alpha^2$ implies the constant size of $\mathcal{C}^t$, as α and τ are constant.

Mixed integer linear program (line 11 of Algorithm 1). Now, given the vectors $\xi = (\xi_1,\ldots,\xi_T)$, $\theta = (\theta_1,\ldots,\theta_T)$, and $\mathcal{D} = \{\mathcal{D}_t\}_{t\in[T]}$, we formulate a mixed integer linear program, denoted by $\mathsf{MILP}[\xi,\theta]$, as follows. $\mathsf{MILP}[\xi,\theta]$ consists of both fractional and integer variables, and three classes of linear constraints, which are defined next. We use the following variables $\mathbf{x} = (x_{t,j})_{t\in[T],j\in[m]}$ and $\mathbf{y} = (y_{t,\kappa})_{t\in[T],\kappa\in\mathcal{C}^t}$, where

- $x_{t,j} \in [0,1]$: denotes the fraction of item j assigned to agents of type t. Some variables $x_{t,j}$ are set to be either 0 or 1 according to our guessing of $\mathcal{D}_t$,
- $y_{t,\kappa} \in \mathbb{N}$: denotes the number of configurations $\kappa \in \mathcal{C}^t$ that are assigned to agents of type t.

Let $\mathcal{O}_{t,v} = \{j \in \mathcal{O} |\ u_{t,j} = v\}$ and $\mathcal{G}_t = \cup_{v\in\mathcal{H}_t}\mathcal{O}_{t,v}$, for $t \in [T]$. The objective of $\mathsf{MILP}[\xi,\theta]$ is to maximize

$$f(\mathbf{x},\mathbf{y}) = \sum_{t\in[T]}\left(\sum_{j\in\mathcal{G}_t}\hat{u}_{t,j}x_{t,j} + \sum_{\kappa\in\mathcal{C}^t} y_{t,\kappa}\hat{u}_{t,\kappa} + \sum_{v\in\mathcal{S}_t} v\sum_{j\in\mathcal{O}_{t,v}} x_{t,j}\right)$$

subject to the following classes of constraints. In the first class of constraints, we require that, for every type $t \in [T]$, the total number of configurations and huge items assigned to agents of this type is exactly the number of agents:

$$\sum_{\kappa\in\mathcal{C}^t} y_{t,\kappa} + \sum_{j\in\mathcal{G}_t} x_{t,j} = n_t. \tag{2}$$

For the second class of constraints, we first ensure that every item $j \in \mathcal{O}$ is assigned to exactly one agent type:

$$\sum_{t\in[T]} x_{t,j} = 1. \tag{3}$$

For every $t \in [T]$ and for every $v \in \mathcal{L}_t$, we must have that the total number of items of value v that are assigned to the group of agents $\mathcal{A}_t$ is at least the total number of items of the same value used in the chosen configurations:

$$\sum_{j\in\mathcal{O}_{t,v}} x_{t,j} \ge \sum_{\kappa\in\mathcal{C}^t} \kappa_v \cdot y_{t,\kappa}. \tag{4}$$

The third class of constraints guarantees that, for $t \in [T]$, the total value of the small configurations and small items assigned to agents of type t is at least θ_t times the number of agents owning small configurations:

$$\sum_{\kappa \in \mathcal{C}^t_{<\theta_t}} y_{t,\kappa} \hat{u}_{t,\kappa} + \sum_{v \in \mathcal{S}_t} v \sum_{j \in \mathcal{O}_{t,v}} x_{t,j} \geq \left(n_t - \sum_{j \in \mathcal{G}_t} x_{t,j} - \sum_{\kappa \in \mathcal{C}^t_{\geq \theta_t}} y_{t,\kappa} \right) \cdot \theta_t. \quad (5)$$

An optimal basic (fractional) solution (BFS) to $\mathsf{MILP}[\xi, \theta]$, if there exists one, is denoted by $(\mathbf{x}^*, \mathbf{y}^*)$. We now show how to convert it into an allocation with the desired properties.

Lemma 3. *A basic solution $(\mathbf{x}^*, \mathbf{y}^*)$ to $\mathsf{MILP}[\xi, \theta]$ has at most $W = 2(2 + \log_{1-\alpha} \alpha^2) \cdot T$ fractional components.*

Proof. It suffices to prove the $\mathbf{x}^*$ has at most W fractional components. Since the number of nontrivial constraints of $\mathsf{MILP}[\xi, \theta]$ is $h = 2T + \sum_{t \in [T]} |\mathcal{L}_t| + m$, $\mathbf{x}^*$ has at most h positive components. Consider the family of constraints (3). In each of these constraints, there are only two cases: (i) there is exactly one positive component $x^*_{t,j} = 1$, and (ii) there are at least two nonintegral components. Denote by ℓ_1 and ℓ_2, respectively, the number of constraints (3) in each case. Then we have that $\ell_1 + \ell_2 = m$. The total number of positive components of $\mathbf{x}^*$ is at least $\ell_1 + 2\ell_2 = 2m - \ell_1$. By an earlier argument, we must have that $2m - \ell_1 \leq h = 2T + \sum_{t \in [T]} |\mathcal{L}_t| + m$ or, equivalently, $\ell_1 \geq m - 2T - \sum_{t \in [T]} |\mathcal{L}_t|$. Therefore, the number of nonintegral components of $\mathbf{x}^*$ is at most $h - \left(m - 2T - \sum_{t \in [T]} |\mathcal{L}_t|\right) = 2\left(2T + \sum_{t \in [T]} |\mathcal{L}_t|\right) \leq 2\left(2 + \log_{1-\alpha} \alpha^2\right) \cdot T$, as $|\mathcal{L}_t|$ is upper-bounded by $\log_{1-\alpha} \alpha^2$. □

Converting $(\bar{\mathbf{x}}^*, \mathbf{y}^*)$ into an allocation. We first show how to round $(\mathbf{x}^*, \mathbf{y}^*)$ to an integer solution $(\bar{\mathbf{x}}^*, \mathbf{y}^*)$. We need only to round $\mathbf{x}^*$ (or, more precisely, the fractional components of $\mathbf{x}^*$) since $\mathbf{y}^*$ is already integral. We find a nonnegative basic (integer) solution to the following linear program (LP):

$$\begin{aligned}
\sum_{j \in \mathcal{G}_t} z_{t,j} &= n_t - \sum_{\kappa \in \mathcal{C}^t} y^*_{t,\kappa}, && t \in [T], \\
\sum_{t \in [T]} z_{t,j} &= 1, && j \in \mathcal{O}, \\
\sum_{j \in \mathcal{O}_{t,v}} z_{t,j} &\geq \sum_{\kappa \in \mathcal{C}^t} \kappa_v y^*_{t,\kappa}, && t \in [T], v \in \mathcal{L}_t, \\
\sum_{j \in \mathcal{O}_{t,v}} z_{t,j} &\geq \lfloor \sum_{j \in \mathcal{O}_{t,v}} x^*_{t,j} \rfloor, && t \in [T], v \in \mathcal{S}_t,
\end{aligned}$$

in which we set $z_{t,j} = x^*_{t,j}$ for all t, j such that $x^*_{t,j}$ is integral. One can see that LP has the form of $Ax \geq b$, where A has only 0- or 1-entries while b is an integer vector. Furthermore, every column of A contains exactly two nonzero entries. Hence, A is a totally unimodular matrix, making every basic solution of LP integral. It is well-known that a basic solution $\mathbf{z}^*$ of LP can be found in polynomial time. We set $\bar{x}^*_{t,j} = z^*_{t,j}$ if $x^*_{t,j}$ is fractional, and $\bar{x}^*_{t,j} = x^*_{t,j}$ otherwise. One can check that the obtained solution $(\bar{\mathbf{x}}^*, \mathbf{y}^*)$ fulfills the constraints (2), (3), and (4), though it may violate the constraints (5). We will show later that this violation does not affect much the quality of the rounded solution. Finally, we apply Algorithm 2 for each $t \in [T]$ to convert $(\bar{\mathbf{x}}^*, \mathbf{y}^*)$ into an allocation.

Algorithm 2 CONVERT$(\bar{\mathbf{x}}^*, \mathbf{y}^*, \mathcal{A}_t)$

1: Allocate item j to group $\mathcal{A}_t$ if $\bar{x}^*_{t,j} = 1$, for all j, and follow the next steps to assign items to agents in this group.
2: *Huge items*: we allocate each huge item to at most one agent such that nobody gets more than one huge item.
3: *Large items*: agents who have not received any huge item are assigned exactly one configuration each, and we further allocate large items to each of them based on $y^*_{t,\kappa}$.
4: *Small items*: we greedily allocate small items among groups of agents having a configuration of value less than $(1-3\alpha)\theta_t$. Pick an arbitrary agent from this group and assign small items to her (one-by-one) until her value exceeds $(1-3\alpha)\theta_t$. Remove the last item assigned to her and repeat this process with the remaining agents and remaining items. At the end of the process, if there still remain some items, we allocate them arbitrarily to agents.

Correctness of Algorithm 1. One needs to prove that, for a given feasible allocation π, there is an allocation $\pi^* \in \mathcal{P}_\varepsilon(\mathcal{I})$ such that $\mathsf{sw}_e(\pi^*, \mathcal{I}) \geq (1-\varepsilon)\mathsf{sw}_e(\pi, \mathcal{I})$ and $\mathsf{sw}_u(\pi^*, \mathcal{I}) \geq (1-\varepsilon)\mathsf{sw}_u(\pi, \mathcal{I})$. By Lemma 2, one can transform π into some allocation π' of the *desired structure*, without decreasing the egalitarian social welfare of the former one while keeping the utilitarian social welfare unchanged. If some agent gets nothing in π then $\mathsf{sw}_e(\pi, \mathcal{I}) = 0$. Therefore, this allocation must be dominated by $\bar{\pi} \in \mathcal{P}_\varepsilon(\mathcal{I})$. Hence, we now consider the case when every agent gets at least one item in π. From π' one can easily determine the value ξ_t for every $t \in [T]$, and let $\theta_t = \min_{i \in \mathcal{A}_t} u_i(\pi'_i) \in (\frac{1}{2}\xi_t, 2\xi_t)$. Let ξ^*_t be a lower bound of ξ_t such that $\xi^*_t \leq \xi_t < (1+\alpha)\xi^*_t$. This lower bound is considered at Step 3 of Algorithm 1. Also, let $\mathcal{G}_t$ be the set of huge items assigned to agents of type t by allocation π'. Then a set $\mathcal{D}_t$ of the d highest-value huge items in $\mathcal{G}_t$ is considered by the loop in line 7 of Algorithm 1. If $|\mathcal{G}_t| \leq d$, $\mathcal{D}_t$ is exactly $\mathcal{G}_t$. Let $\hat{\theta}_t = \min_{i \in \mathcal{A}_t} \hat{u}_i(\pi'_i)$. From (1) it follows that $(1-\alpha)\theta_t \leq \hat{\theta}_t \leq \theta_t$. Hence, $\hat{\theta}_t \in (\frac{1-\alpha}{2}\xi_t, 2\xi_t)$ or $\hat{\theta}_t \in (\tau\xi^*_t, \xi^*_t/\tau)$, where $\tau = (1-\alpha)/2$. Then there must be a lower bound $\theta^*_t \in (\tau\xi^*_t, \xi^*_t/\tau)$ of $\hat{\theta}_t$, considered by the loop at line 8 of Algorithm 1, such that $\theta^*_t \leq \hat{\theta}_t < \frac{\theta^*_t}{1-\alpha}$. Let $\xi^* = (\xi^*_1, \ldots, \xi^*_T)$ and $\theta^* = (\theta^*_1, \ldots, \theta^*_T)$. Now the Algorithm 1 tries to solve the $\mathsf{MILP}[\xi^*, \theta^*]$ to obtain some allocation with the desired properties as claimed in Lemma 4.

Lemma 4. *There exists an integer solution to $\mathsf{MILP}[\xi^*, \theta^*]$. Also, a rounded solution of an optimal BFS to $\mathsf{MILP}[\xi^*, \theta^*]$ corresponds to an allocation π^* such that $\mathsf{sw}_u(\pi^*, \mathcal{I}) \geq (1-\varepsilon)\mathsf{sw}_u(\pi, \mathcal{I})$ and $\mathsf{sw}_e(\pi^*, \mathcal{I}) \geq (1-\varepsilon)sw_e(\pi, \mathcal{I})$.*

Proof. The construction of an integer solution to $\mathsf{MILP}[\xi^*, \theta^*]$ from π' is not difficult to verify, and we omit it here due to lack of space. We now prove the second part of the lemma. Let $(\mathbf{x}^*, \mathbf{y}^*)$ be an optimal BFS of $\mathsf{MILP}[\xi^*, \theta^*]$ returned by Step 10 of Algorithm 1, and $(\bar{\mathbf{x}}^*, \mathbf{y}^*)$ be its rounded solution. One can see that $(\bar{\mathbf{x}}^*, \mathbf{y}^*)$ satisfies all the constraints because of the LP constraints, except (5).

In the following we will show how much the rounding step (line 11 of Algorithm 1) violates the constraints (5). Due to the last family of constraints of LP, it holds that $\sum_{j\in\mathcal{O}_{t,v}}\bar{x}^*_{t,j} \geq \sum_{j\in\mathcal{O}_{t,v}}x^*_{t,j}-1$. Let $L_t=\sum_{v\in\mathcal{S}_t}v\sum_{j\in\mathcal{O}_{t,v}}x^*_{t,j}$. For every agent type t and for every item value v, we have that:

$$\sum_{v\in\mathcal{S}_t}v\sum_{j\in\mathcal{O}_{t,v}}\bar{x}^*_{t,j} \geq L_t-\sum_{v\in\mathcal{S}_t}v \geq L_t-\alpha^2\xi^*_t\cdot\sum_{k\in\mathbb{N}}(1-\alpha)^k = L_t-\alpha\xi^*_t. \quad (6)$$

The second inequality follows from the definition of $\mathcal{S}_t$.

Consider the constraints (5). Let $K_t=n_t-\sum_{j\in\mathcal{G}_t}x^*_{t,j}-\sum_{\kappa\in\mathcal{C}^t_{\geq\theta_t}}y^*_{t,\kappa}$. Since $(\mathbf{x}^*,\mathbf{y}^*)$ satisfies (5), from (6) we have:

$$\sum_{\kappa\in\mathcal{C}^t_{<\theta_t}}y^*_{t,\kappa}\hat{u}_{t,\kappa}+\sum_{v\in\mathcal{S}_t}v\sum_{j\in\mathcal{O}_{t,v}}\bar{x}^*_{t,j} \geq K_t\theta^*_t-\alpha\xi^*_t \geq K_t\theta^*_t-(\alpha/\tau)\cdot\theta^* \geq K_t\cdot(1-3\alpha)\theta^*_t, \quad (7)$$

since $\xi^*_t<\theta^*_t/\tau$ and $\tau=(1-\alpha)/2>1/3$ for $\alpha<1/3$.

Let $U_t=\sum_{j\in\mathcal{G}_t}\hat{u}_{t,j}\bar{x}^*_{t,j}+\sum_{\kappa\in\mathcal{C}^t}y^*_{t,\kappa}\hat{u}_{t,\kappa}+\sum_{v\in\mathcal{S}_t}v\sum_{j\in\mathcal{O}_{t,v}}\bar{x}^*_{t,j}$. For every t, it suffices to consider the case when $|\mathcal{G}_t|>d$, and thus $\mathcal{D}_t\subset\mathcal{G}_t$. Let $s=\arg\min_{j\in\mathcal{D}_t}u_{t,j}$ and $\mathcal{G}'_t\subseteq\mathcal{G}_t$ be the set of items j for which $x^*_{t,j}$ was fractional but rounded down to 0 by the rounding (line 11 of Algorithm 1). By Lemma 3, we have that $|\mathcal{G}'_t|\leq W$. Hence,

$$\sum_{j\in\mathcal{G}_t}\hat{u}_{t,j}\bar{x}^*_{t,j} \geq \sum_{\mathcal{G}_t}\hat{u}_{t,j}x^*_{t,j}-\sum_{j\in\mathcal{G}'_t}\hat{u}_{t,j}x^*_{t,j} \geq \sum_{j\in\mathcal{G}_t}\hat{u}_{t,j}x^*_{t,j}-W\cdot\hat{u}_{t,s}$$

$$\geq \sum_{j\in\mathcal{G}_t}\hat{u}_{t,j}x^*_{t,j}-\frac{W}{|\mathcal{D}_t|}\cdot\sum_{j\in\mathcal{D}_t}\hat{u}_{t,j} \geq (1-\alpha)\sum_{j\in\mathcal{G}_t}\hat{u}_{t,j}x^*_{t,j}. \quad (8)$$

From (6) and (8) one can show, for every $t\in[T]$, that

$$U_t \geq (1-3\alpha)\left(\sum_{j\in\mathcal{G}_t}\hat{u}_{t,j}x^*_{t,j}+\sum_{\kappa\in\mathcal{C}^t}y^*_{t,\kappa}\hat{u}_{t,\kappa}+\sum_{v\in\mathcal{S}_t}v\sum_{j\in\mathcal{O}_{t,v}}x^*_{t,j}\right).$$

Now the integer solution $(\bar{\mathbf{x}}^*,\mathbf{y}^*)$ is converted into an allocation π^* by line 12 of Algorithm 1. By converting, we have that $\mathsf{sw}_u(\pi^*,\hat{\mathcal{I}})=\sum_{t\in[T]}U_t \geq (1-3\alpha)f(\mathbf{x}^*,\mathbf{y}^*) \geq (1-3\alpha)\mathsf{sw}_u(\pi',\hat{\mathcal{I}}) \geq (1-4\alpha)\mathsf{sw}_u(\pi,\mathcal{I})$. The third inequality is due to the fact that π' can be converted into a feasible solution to $\mathsf{MILP}[\xi^*,\theta^*]$, whose value is equal to the utilitarian social welfare of π' w.r.t. $\hat{\mathcal{I}}$.

We now prove that $\mathsf{sw}_e(\pi^*,\mathcal{I}) \geq (1-5\alpha)\mathsf{sw}_e(\pi,\mathcal{I})$. Notice that the assignment of small items at Step 4 of Algorithm 2 is doable due to (7). Let $\mathcal{A}'_t$ be the set of agents who received configurations of values less than $(1-3\alpha)\theta^*_t$ due to this algorithm, for all $t\in[T]$. Let i be such an agent. By the greedy assignment, the small items are assigned to her until her value exceeds $(1-3\alpha)\theta^*_t$. Suppose that j is the last item assigned to her. Then it must be that the value of the agent before getting the last item is at least $(1-3\alpha)\theta^*_t-\hat{u}_{t,j} \geq (1-3\alpha)\theta^*_t-\alpha^2\xi^*_t>(1-3\alpha)\theta^*_t-\frac{2\alpha^2}{1-\alpha}\theta^*_t \geq (1-4\alpha)\theta^*_t$, for $\alpha<1/3$. The first inequality is because j is a small item and the second inequality follows from the fact that $\frac{1-\alpha}{2}\xi^*_t\leq\theta^*_t$. Therefore, $\mathsf{sw}_e(\pi^*,\hat{\mathcal{I}}) \geq \min_{t\in[T]}\{(1-3\alpha)\theta^*_t\}$. Note that $\theta^*_t \geq (1-\alpha)\hat{\theta}_t \geq (1-\alpha)^2\theta_t \geq (1-2\alpha)\theta_t$. Hence, $\mathsf{sw}_e(\pi^*,\hat{\mathcal{I}}) \geq \min_{t\in[T]}\{(1-3\alpha)(1-2\alpha)\theta_t\} \geq \min_{t\in[T]}\{(1-5\alpha)\theta_t\}=(1-5\alpha)\mathsf{sw}_e(\pi',\mathcal{I}) \geq (1-5\alpha)\mathsf{sw}_e(\pi,\mathcal{I})$.

Note that by rounding down item values, it holds that $u_i(\pi^*_i)\geq\hat{u}_i(\pi^*_i)$ for all i. Hence, $\mathsf{sw}_e(\pi^*,\mathcal{I}) \geq \mathsf{sw}_e(\pi^*,\hat{\mathcal{I}}) \geq (1-5\alpha)\mathsf{sw}_e(\pi,\mathcal{I})$, and $\mathsf{sw}_u(\pi^*,\mathcal{I}) \geq \mathsf{sw}_u(\pi^*,\hat{\mathcal{I}}) \geq (1-4\alpha)\mathsf{sw}_u(\pi,\mathcal{I})$.

Finally, by choosing $\alpha=\frac{\varepsilon}{5}$ we have completed the proof of Lemma 4. □

Complexity analysis of Algorithm 1. It is not difficult to see that the overall running time of Algorithm 1 is upper-bounded by $\mathcal{T}=O(P_1\cdot P_2\cdot P_3\cdot P_4)$ in which P_1, P_2, and P_3, respectively, are the sizes of the sets $\mathcal{X}$, $\mathcal{Q}$, and Θ, while P_4 is the amount of time needed for solving $\mathsf{MILP}[\xi,\theta]$ and for solving the linear program LP. We have that $P_1=O(\log^T(m\cdot u_{\max}))$, P_2 is at most $|\mathcal{H}_1|^d\times\cdots\times|\mathcal{H}_T|^d=O(m^{T(\frac{W}{\alpha}+1)})$ (which is polynomial in m since T, W, and α are constants), and $P_3=O(1)$. To estimate P_4, note that $\mathsf{MILP}[\xi,\theta]$ has many variables but a constant number of integer ones and thus can be solved in time polynomial in the size of input (see Lenstra [1983]). In addition, computing a basic (integer) solution to LP can be also done in polynomial time. Hence, $\mathcal{T}$ is a polynomial in the size of $\mathcal{I}$. This completes the proof of Theorem 3. □

5 Conclusions

We have studied the problem of computing approximately max-min fair and utilitarian efficient allocations for additive valuation functions. In particular, we have modeled the problem as a bi-criteria optimization problem and presented two polynomial-time algorithms for constructing approximate Pareto sets, in two special but nontrivial settings when there are a few agent types or there are a few item types. Our first algorithm is based on a dynamic programming scheme that has been shown to be very useful in solving other fair and efficient allocation problems. In fact, one of our results regarding finding min-max allocations for the constant-item-type case answers an open problem raised by Jansen and Maack [2019] in the context of machine scheduling. Our second algorithm is achieved by applying a linear-programming method, which can be seen as a significant extension of Jansen and Maack's method. We believe that our technique can be potentially applied to a wider class of resource allocation problems with more general objectives, e.g., the ones considered by Alon *et al.* [1998]. This would be an interesting direction for future work. Another idea is to see whether we can obtain similar results to what we have done in this paper with respect to the criterion of envy-freeness.

Acknowledgments

We thank the anonymous reviewers for their very helpful comments and suggestion. Part of this work has been done when the first author was visiting Vietnam Institute for Advanced Study in Mathematics and their support is kindly acknowledged. This work was partially supported by the Stay-Connected@HHU Programme and DFG grants RO 1202/14-2 and RO 1202/21-1.

References

[Alon *et al.*, 1998] Noga Alon, Yossi Azar, Gerhard Woeginger, and Tal Yadid. Approximation schemes for scheduling on parallel machines. *J. Scheduling*, 1(1):55–66, 1998.

[Aumann and Dombb, 2015] Yonatan Aumann and Yair Dombb. The efficiency of fair division with connected pieces. *ACM Trans. Economics and Comput.*, 3(4):23:1–23:16, 2015.

[Aziz *et al.*, 2019] Haris Aziz, Xin Huang, Nicholas Mattei, and Erel Segal-Halevi. The constrained round robin algorithm for fair and efficient allocation. *CoRR*, abs/1908.00161, 2019.

[Barman *et al.*, 2018] Siddharth Barman, Sanath Krishnamurthy, and Rohit Vaish. Finding fair and efficient allocations. In *Proc. 18th ACM Conference on Economics and Computation*, pages 557–574, 2018.

[Bei *et al.*, 2019] Xiaohui Bei, Xinhang Lu, Pasin Manurangsi, and Warut Suksompong. The price of fairness for indivisible goods. In *Proc. 28th International Joint Conference on Artificial Intelligence*, pages 81–87, 2019.

[Bertsimas *et al.*, 2011] Dimitris Bertsimas, Vivek Farias, and Nikolaos Trichakis. The price of fairness. *Operations Research*, 59(1):17–31, 2011.

[Bliem *et al.*, 2016] Bernhard Bliem, Robert Bredereck, and Rolf Niedermeier. Complexity of efficient and envy-free resource allocation: Few agents, resources, or utility levels. In *Proc. 25th International Joint Conference on Artificial Intelligence*, pages 102–108, 2016.

[Bouveret *et al.*, 2016] Sylvain Bouveret, Yann Chevaleyre, and Nicolas Maudet. Fair allocation of indivisible goods. In Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel Procaccia, editors, *Handbook of Computational Social Choice*, chapter 12, pages 284–310. Cambridge University Press, 2016.

[Bouveret *et al.*, 2017] Sylvain Bouveret, Katarína Cechlárová, Edith Elkind, Ayumi Igarashi, and Dominik Peters. Fair division of a graph. In *Proc. 26th International Joint Conference on Artificial Intelligence*, pages 135–141, 2017.

[Brânzei *et al.*, 2016] Simina Brânzei, Yuezhou Lv, and Ruta Mehta. To give or not to give: Fair division for single minded valuations. In *Proc. 25th International Joint Conference on Artificial Intelligence*, pages 123–129, 2016.

[Bredereck *et al.*, 2019] Robert Bredereck, Andrzej Kaczmarczyk, Dusan Knop, and Rolf Niedermeier. High-multiplicity fair allocation: Lenstra empowered by *n*-fold integer programming. In *Proc. 19th ACM Conference on Economics and Computation*, pages 505–523, 2019.

[Brown, 1979] J. Randall Brown. The knapsack sharing problem. *Operations Research*, 27(2):341–355, 1979.

[Caragiannis *et al.*, 2012] Ioannis Caragiannis, Christos Kaklamanis, Panagiotis Kanellopoulos, and Maria Kyropoulou. The efficiency of fair division. *Theory Comput. Syst.*, 50(4):589–610, 2012.

[Caragiannis *et al.*, 2016] Ioannis Caragiannis, David Kurokawa, Hervé Moulin, Ariel Procaccia, Nisarg Shah, and Junxing Wang. The unreasonable fairness of maximum Nash welfare. In *Proc. 16th ACM Conference on Economics and Computation*, pages 305–322, 2016.

[Escoffier *et al.*, 2013] Bruno Escoffier, Laurent Gourvès, and Jérôme Monnot. Fair solutions for some multiagent optimization problems. *Autonomous Agents and Multi-Agent Systems*, 26(2):184–201, 2013.

[Galil and Megiddo, 1979] Zvi Galil and Nimrod Megiddo. A fast selection algorithm and the problem of optimum distribution of effort. *J. ACM*, 26(1):58–64, 1979.

[Goel *et al.*, 2001] Ashish Goel, Adam Meyerson, and Serge Plotkin. Combining fairness with throughput: Online routing with multiple objectives. *Journal of Computer and System Sciences*, 63(1):62–79, 2001.

[Igarashi and Peters, 2019] Ayumi Igarashi and Dominik Peters. Pareto-optimal allocation of indivisible goods with connectivity constraints. In *Proc. 33rd AAAI Conference on Artificial Intelligence*, pages 2045–2052. AAAI Press, 2019.

[Jansen and Maack, 2019] Klaus Jansen and Marten Maack. An EPTAS for scheduling on unrelated machines of few different types. *Algorithmica*, 81(10):4134–4164, 2019.

[Kleinberg *et al.*, 2001] Jon Kleinberg, Yuval Rabani, and Éva Tardos. Fairness in routing and load balancing. *Journal of Computer and System Sciences*, 63(1):2–20, 2001.

[Kones and Levin, 2019] Ishai Kones and Asaf Levin. A unified framework for designing EPTAS for load balancing on parallel machines. *Algorithmica*, 81(7):3025–3046, 2019.

[Krysta *et al.*, 2013] Piotr Krysta, Orestis Telelis, and Carmine Ventre. Mechanisms for multi-unit combinatorial auctions with a few distinct goods. In *Proc. 12th International Conference on Autonomous Agents and Multiagent Systems*, pages 691–698. IFAAMAS, 2013.

[Lang and Rothe, 2015] Jérôme Lang and Jörg Rothe. Fair division of indivisible goods. In Jörg Rothe, editor, *Economics and Computation. An Introduction to Algorithmic Game Theory, Computational Social Choice, and Fair Division*, Springer Texts in Business and Economics, chapter 8, pages 493–550. Springer, 2015.

[Lenstra Jr., 1983] Hendrik Lenstra Jr. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8(4):538–548, 1983.

[Lesmana *et al.*, 2019] Nixie Lesmana, Xuan Zhang, and Xiaohui Bei. Balancing efficiency and fairness in on-demand ridesourcing. In *Proc. 33rd Neural Information Processing Systems (NeurIPS)*, pages 5310–5320, 2019.

[Suksompong, 2019] Warut Suksompong. Fairly allocating contiguous blocks of indivisible items. *Discrete Applied Mathematics*, 260:227–236, 2019.

[Woeginger, 1997] Gerhard Woeginger. A polynomial-time approximation scheme for maximizing the minimum machine completion time. *Operations Research Letters*, 20(4):149–154, 1997.