Proportional Allocation of Indivisible Resources under Ordinal and Uncertain Preferences (Supplementary material)

Zihao Li[1] **Xiaohui Bei**[1] **Zhenzhen Yan**[1]

[1]School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore

A MISSING PROOFS IN SECTION 3

Theorem 3.2. EXISTSPOSSIBLYFAIR with regard to weak SD proportionality can be solved in polynomial time.

Proof. Theorem 7 in Aziz et al. [2015] showed that when agents have strict and deterministic preferences, a weak SD proportional allocation exists if and only if (1) $m = n$ and no item is least preferred by all agents, or (2) $m > n$. Furthermore, when such allocation exists, it can also be found in polynomial time.

This result allows us to derive a simple algorithm for EXISTSPOSSIBLYFAIR. First, when $m > n$, we can simply find an arbitrary set of linear extensions and compute a corresponding weak SD proportional allocation. When $m = n$, we need to find a set of linear extensions in which no item is least preferred by all agents, then compute a weak SD proportional allocation with regard to this preference profile. To find such a preference profile, we can construct a bipartite graph between the agents and the items, such that an edge exists if and only if the item is not least preferred by the agent in some preference profiles. If there exists a perfect matching in this bipartite graph, this perfect matching corresponds to an allocation that is weak SD proportional with non-zero probability. If the graph does not have a perfect matching, because each agent has at least $n - 1$ edges, by Hall's marriage theorem [Hall, 2009], there must exist one item which is least preferred by all agents in all preference profiles. Thus, there is no allocation with non-zero fair probability. □

Theorem 3.4. EXISTSCERTAINLYFAIR with regard to weak SD proportionality can be solved in polynomial time when all agents have identical preferences.

Proof. Firstly, we simplify some notions when agents have identical preferences: we use K to represent the number of equivalent classes and use S_j to represent the j-th equivalent class. Then we re-number all items with indices 1 to n such that $S_1 = \{1, \ldots, |S_1|\}$ and $S_j = \{\sum_{k=1}^{j-1} |S_k| + 1, \ldots, \sum_{k=1}^{j} |S_k|\}$ for each $1 < j \leq K$. This means for all agents, smaller-indexed items are always more preferred.

We consider the following greedy algorithm. For each agent from 1 to n, we consider the remaining items in increasing order of indices, and allocate them to this agent one by one, until the current allocation met the fairness condition for this agent with probability 1. We repeat this process until all items are fully allocated or all agents have received their desired bundles of items.

In the following, we show that if a certainly fair allocation M exists in a problem instance, the above greedy algorithm can always return such a certainly fair allocation.

Next, we perform a three-step transformation on the given certainly fair allocation M:

- For each agent i, if removing the item with largest index in her bundle can still keep her fairness condition met, we will remove that item from i's bundle. Repeat this process until no items can be removed anymore. After this step, we denote sta_i as the equivalent class the worst item in i's bundle belongs to. Then, for any agent i, she could not have her fairness condition met after throwing the present worst item and all owned items are always least preferred in their respective equivalent classes in the worst case, so there must exist $\sum_{j=1}^{\mathsf{sta}_i - 1} |S_j| < k_i \leq \sum_{j=1}^{\mathsf{sta}_i} |S_j|$ such that agent i is allocated at least $\lfloor \frac{k_i}{n} + 1 \rfloor$ of her top k_i items in the worst case mentioned above.
- Next, we find any agent who has item o that has a larger index than some thrown away item o', and we replace o with o'. Again repeat this process until no such changes can be made. We know M is still certainly fair. After this step, the present allocation must contain exactly all the items in $[m']$ for some $m' \leq m$.
- Finally, if there exists agent i and j such that the worst item in i's bundle has a larger index than the worst items in j's bundle, and agent i also has an item o_i

Accepted for the 38th Conference on Uncertainty in Artificial Intelligence (UAI 2022).

with a smaller index than some item o_j belonging to agent j, we will be able to swap o_i and o_j while still keeping the fairness conditions for both agents. This obviously holds for agent j. For agent i, this claim is true because o_j is still not worse than the worst item in i's bundle and it still holds that agent i is allocated at least $\lfloor \frac{k_i}{n} + 1 \rfloor$ of her top k_i items in the worst case for that k_i. Repeat this until no more exchanges can be made. After this step, we know the allocation must still be certainly fair and it is a consecutive partition of $[m']$ for some $m' \leq m$.

Finally, it is easy to see if there exists a consecutive partition that is certainly fair, the allocation computed by our greedy algorithm must be also certainly fair. □

Theorem 3.5. EXISTSCERTAINLYFAIR and HIGHESTPROB with regard to weak SD proportionality can be solved in polynomial time when the number of agents is constant.

Proof. Let $m = qn + r$, where q, r are integers and $0 \leq r < n$.

First, we observe that any agent who gets $q + 1$ items is always weak SD proportional with probability one. Thus we can assume that no agent gets more than $q + 1$ items in the allocation with the highest fair probability. This also implies that every agent gets at least $q_m = q - (n - 1 - r)$ items, because otherwise the total number of allocated items will be less than $q-(n-1-r)+(n-1)(q+1) = qn+r = m$.

If there exists some agent i such that $|S_{i,1}| < n \cdot q_m$, we can choose $\min\{|S_{i,1}|, q_m\}$ items from $S_{i,1}$ to be allocated to agent i, which guarantees the fair condition for agent i to be met with probability one. Next, we assign all other agents $q + 1$ arbitrary items each and give the remaining items back to i. It is easy to check that this allocation is weak SD proportional with probability one.

Now we consider the case where $|S_{i,1}| \geq n \cdot q_m$ holds for every agent $i \in N$. This means that the total number of items not in the first equivalent class for any agent is no more than $n(n - 1 - r) + r$.

For each agent i, if there exists an integer $1 \leq j < |k_i|$, such that $1 + \sum_{t=1}^{j-1} |S_{i,t}| \geq (q' - 1)n$ and $\sum_{t=1}^{j+1} |S_{i,t}| < q'n$ for some $q' \leq q + 1$, then the items in the j-th and $(j + 1)$-th equivalent classes can be merged into one equivalent class without affecting the fair probability of any allocation, because all items in these two equivalent classes will always have the ranking in the range between $(q' - 1)n$ and $q'n - 1$. We don't really care about the specific ranking in this range because we can simplify the definition of weak SD proportionality to only consider whether there exists $k \in \{n-1, 2n-1, \ldots, qn-1, m\}$ such that agent i is allocated at least $\lfloor \frac{k}{n} + 1 \rfloor$ of her top k items. We repeatedly merge such adjacent equivalent classes until none can be found. By the end of this procedure, the total number of equivalent classes in each agent's preference list is no more than $1 + 2 \cdot (n - 1 - r + 1) = 2(n - r) + 1$.

Next, we enumerate the number of items assigned to each agent i from each equivalent class. More specifically, for each agent i, we first enumerate the total number of items in her bundle and then the number $C_{i,j}$ of items in each $S_{i,j}$ assigned to agent i. For each set of numbers $\{C_{i,j}\}$, we use the following perfect matching algorithm to find candidate allocations.

- We construct the bipartite graph $G = (A \cup B, E)$ where A has $C_{i,j}$ duplicate vertices representing the positions for items in $S_{i,j}$ for each agent $i \in [n]$ and each equivalent class $j \in [k_i]$, and B contains m vertices each representing an item. For each duplicate vertex for $S_{i,j}$ in A and each vertex in B corresponding to the item in $S_{i,j}$, there is an edge between them in E;
- If there exists a perfect matching between A and B (which also implies $|A| = |B|$), we pick the allocation where each item whose corresponding vertex is matched with one duplicate vertex for one $S_{i,j}$ is assigned to agent i and it meets our requirements.

If such allocation exists, its fair probability can be computed using the algorithm designed for FAIRPROB. Finally, we return the allocation with the highest fair probability.

Finally we analyze the time complexity of the above algorithm. The first step takes $O(nm)$ to enumerate each agent and each item. Merging equivalent classes also takes $O(nm)$ time. Next, for the enumeration step, because each number $C_{i,j} \leq n(n - 1 - r) + r$ when $j > 1$ and the total number of items in any bundle is between $q - (n - 1 - r)$ and $q + 1$, the total number of possibilities is $O((n(n - 1 - r) + r + 1)^{2(n-r)\cdot n} \cdot (n - 1 - r + 1 + 1)^n) = O(n^{4n^2+n+1})$. Finally, deciding the existence of an allocation and calculating its fair probability takes $O(m^3)$ time. Thus, the overall time complexity of this algorithm is $O(n^{4n^2+n+1} \cdot m^3)$. □

B MISSING PROOFS IN SECTION 4

Theorem 4.1. FAIRPROB with regard to SD proportionality can be solved in polynomial time.

Proof. Similar to the case of FAIRPROB with regard to weak SD proportionality, we can compute the fair probability of an allocation by calculating the probability that each agent meets the fair condition independently. We again use dynamic programming to calculate the probability of each $i \in N$ satisfying the fair condition.

Fix one specific agent $i \in N$, we know there are $K := k_i$ equivalent classes and the j-th equivalent class has $s_j := |S_{i,j}|$ items. We first calculate the number of items owned

by i in the j-th equivalent class of i and we denote it by $r_j \leq s_j$.

When calculating the fair probability of one agent, we care only about the number of items owned by this agent i among her top s items for each integer $s \in [m]$, but not the specific ranking of any single item. Thus, we use $P_{\mathsf{num},\mathsf{tot},\mathsf{dis}}$ to represent the probability that there are tot items owned by agent i arranged in the top num items with a fair verification parameter dis defined as $(\mathsf{tot} - \lceil \frac{\mathsf{num}}{n} \rceil)$. The purpose of defining dis as such is to evaluate how much agent i is ahead of her fair condition. Note that dis should never be negative because otherwise fair condition won't be met.

Algorithm 1 FAIRPROB Algorithm

Require:
m: number of items
n: number of agents
K: number of equivalent classes of agent i
$s_{1,\dots,K}$: number of items in each equivalent classes
$r_{1,\dots,K}$: number of items owned by agent i in each equivalent classes

1: $P_{\mathsf{num},\mathsf{tot},\mathsf{dis}} \leftarrow 0 \quad \forall 0 \leq \mathsf{num}, \mathsf{tot}, \mathsf{dis} \leq m$
2: $P_{0,0,0} \leftarrow 1$
3: **for** $\mathsf{num} \leftarrow 0$ to $m-1$ **do**
4: **for** $\mathsf{tot} \leftarrow 0$ to num **do**
5: **for** $\mathsf{dis} \leftarrow 0$ to tot **do**
6: **if** $P_{\mathsf{num},\mathsf{tot},\mathsf{dis}} \neq 0$ **then**
7: Let sta be the minimum integer such that $\sum_{j\in[\mathsf{sta}]} s_j \geq \mathsf{num}+1$.
8: $\mathsf{PrevP} \leftarrow P_{\mathsf{num},\mathsf{tot},\mathsf{dis}}$
9: $p \leftarrow \frac{\sum_{j\in[sta]} r_j - \mathsf{tot}}{\sum_{j\in[sta]} s_j - \mathsf{num}}$
10: Let id be 1 if num is a multiple of n and 0 otherwise.
11: $P_{\mathsf{num}+1,\mathsf{tot}+1,\mathsf{dis}+1-\mathsf{id}}$ += $\mathsf{PrevP} \cdot p$
12: **if** $\mathsf{dis} - \mathsf{id} \geq 0$ **then**
13: $P_{\mathsf{num}+1,\mathsf{tot},\mathsf{dis}-\mathsf{id}}$ += $\mathsf{PrevP} \cdot (1-p)$
14: **end if**
15: **end if**
16: **end for**
17: **end for**
18: **end for**
19: **return** $\sum_{\mathsf{dis}=0}^{\sum_{j\in[K]} r_j} P_{m,\sum_{j\in[K]} r_j,\mathsf{dis}}$

The complete algorithm is shown in Algorithm 1. The key steps are the state transfers in Steps 3-18. We define sta as the equivalent class that the item with ranking $(\mathsf{num}+1)$ belongs to and p corresponding to the probability of the item with ranking $(\mathsf{num}+1)$ owned by i under the present state in Steps 7-9. We also use id to record the change of the term $\lceil \frac{\mathsf{num}}{n} \rceil$ in the above definition of dis when considering the next item in Step 10.

We consider our transfer based on the situation of whether the next item is owned by agent i. Specifically, if the next item is owned by agent i (corresponding to Step 11), there are $\mathsf{tot}+1$ items owned by i arranged in the top $\mathsf{num}+1$ items and the dis' is equal to $\mathsf{dis}+1-\mathsf{id}$. On the other hand, if the next item is not owned by i (corresponding to Steps 12-14), only tot items in i's bundle are arranged in the top $\mathsf{num}+1$ items, and we need to keep dis nonnegative, so we only transfer the state to $P_{\mathsf{num}+1,\mathsf{tot},\mathsf{dis}-\mathsf{id}}$ when $\mathsf{dis}-\mathsf{id} \geq 0$.

Finally, considering all possible amounts ahead of the fair condition, Step 19 returns the probability that agent i meets her fairness condition with all of her owned items. □

Theorem 4.2. EXISTSPOSSIBLYFAIR with regard to SD proportionality can be solved in polynomial time.

Proof. Aziz et al. [2015] showed how to find a fair allocation when agents have strict and deterministic preferences with ties. We can solve this problem using a similar method. In particular, we can use the result in Aziz et al. [2015] to show that if $m \bmod n \neq 0$, there does not exist a possibly fair allocation in our setting. Therefore, we assume $m = qn$ for some integer q in the following. We will reduce EXISTSPOSSIBLYFAIR to a max-flow problem in a flow network G. The construction is as follows.

Assume a flow network $G = (V, E)$ with a capacity function $c : E \mapsto \mathbb{R}^+$. V consists of a source vertex s, a sink vertex t, a vertex set A of $\sum_{i\in[n]} k_i$ vertices representing each equivalent class of each agent denoted by $A_{i\in[n],j\in[k_i]}$, and a vertex set B of m vertices representing each item denoted by $B_{i\in[m]}$. For each $A_{i,j} \in A$, we create an edge $(s, A_{i,j})$ with capacity $c = \left\lceil \frac{\sum_{k=1}^{j} |S_{i,j}|}{n} \right\rceil - \left\lceil \frac{\sum_{k=1}^{j-1} |S_{i,j}|}{n} \right\rceil$, which represents the quantity required to satisfy the fair condition that: if for agent i and $\sum_{k=1}^{j-1} |S_{i,j}| < k \leq \sum_{k=1}^{j} |S_{i,j}|$, agent i is allocated at least $\lceil \frac{k}{n} \rceil$ of her top k items. For each vertex $B_j \in B$, we create an edge (B_j, t) with capacity $c = 1$. Next, for each $A_{i,j} \in A$ and each $B_k \in B$, we construct an edge $(A_{i,j}, B_k)$ with capacity $c = 1$ if and only if the item corresponding to B_k is in the first j-th equivalent class of agent i.

If there exists a maximum flow with strength m in this flow network G, we can construct an allocation by assigning each item k to the agent i such that there exists one blocking edge $(A_{i,j}, B_k)$ for one $j \in [k_i]$. In this allocation, by placing the items owned by each agent as more preferred as possible, we can show that it must satisfy SD proportionality.

On the other hand, if there exists a possibly fair allocation, it must meet all capacity conditions in the flow newtork G by successively blocking the edges pointing from $S_{i,j}$ to the items in i's bundle in the increasing order of j for each agent i. □

Theorem 4.2. EXISTSCERTAINLYFAIR and HIGHESTPROB with regard to SD proportionality are NP-hard.

Proof. We only prove the NP-hardness for EXISTSCERTAINLYFAIR and it directly implies NP-hardness for HIGHESTPROB.

In order to prove the NP-hardness of EXISTSCERTAINLYFAIR, we reduce from the NP-hard problem denoted as X3C[Johnson and Garey, 1979].

X3C: Given a collection T of subsets of a set of $3s$ elements S, where each subset in T has exactly 3 elements in S, decide whether exists s subsets in T such that each elements in S appears exactly once.

We reduce from X3C to this problem. Considering a X3C problem instance F with $3s$ elements $S = \{S_1, \ldots, S_{3s}\}$ and t 3-sized sets $T = \{T_1, \ldots, T_t\}$ where each T_i contains exactly three elements $T_{i,1}, T_{i,2}, T_{i,3}$ in S. Without loss of generality, we suppose $t \geq 3s$ and $s > 3$. Based on F we construct a problem instance of EXISTSCERTAINLYFAIR with t agents and $4t$ items. Each agent $i \in [t]$ corresponds to the 3-sized set $T_i \in T$.

The $4t$ items are divided into the following five sets:

- A set of $3s$ items corresponding to each element in S, where we still denote this set as $S = \{S_1, \ldots, S_{3s}\}$ for simplifying the notations;
- A set of t items $A = \{A_1, \ldots, A_t\}$, which is denoted to meet the fair condition when considering only the most preferred item of each agent;
- Three sets of size $(t - s)$: $B = \{B_1, \ldots, B_{t-s}\}$, $C = \{C_1, \ldots, C_{t-s}\}$ and $D = \{D_1, \ldots, D_{t-s}\}$, which are used to compensate the agents whose corresponding 3-sized sets are not selected.

To simplify the descriptions of the preference lists, for set A and each agent $i \in [t]$, we partition the set $A - \{A_i\}$ into 4 disjoint subsets $A'_{i,1}, A'_{i,2}, A'_{i,3}, A'_{i,4}$ where $|A'_{i,1}| = s - 1, |A'_{i,2}| = s - 3, |A'_{i,3}| = s, |A'_{i,4}| = t - 3s + 3$. The intuition for this partition is to properly fill the items in the end of every t items.

Next, we can construct the preference lists for agents. The preference list of agent $i \in [t]$ is:

$$A_i \succ (A'_{i,1}, B) \succ (T_i) \succ (C, A'_{i,2}) \succ (D, A'_{i,3}) \succ (\text{others}).$$

Here, the items in the previous term (others) contain the items in $A'_{i,4}$ and the items in $S - T_i$.

We can observe that $A_i, (A'_{i,1}, B)$ form the first t items, and $(T_i), (C, A'_{i,2})$ form the second t items, while $(D, A'_{i,3})$ and (others) form the third and the fourth t items respectively.

If the X3C instance has a satisfying assignment, we construct an allocation as following:

- For each chosen set T_i, the corresponding agent i picks the items $A_i, T_{i,1}, T_{i,2}, T_{i,3}$.
- For each of the remaining unselected sets T_i, items $A_i, B_{\sigma(i)}$, $C_{\sigma(i)}, D_{\sigma(i)}$ are assigned to agent i. Here, σ is a bijection from the remaining unselected sets to $\{1, ..., t-s\}$.

We can easily check the above allocation is always SD proportional.

On the other hand, assume there is an allocation that is SD proportional with probability one, we can also construct a satisfying assignment for the X3C instance. First, to meet the fair conditions, each item A_i must be assigned to agent i. Then, among the first t items, only $t - s$ items in B have not been assigned now, which means that at least $t - (t - s) = s$ agents cannot pick any item other than her corresponding item in set A among the first t items. Thus, every agent i of these s agents should pick all 3 items in their corresponding T_i to meet the fair condition certainly where agent i is allocated at least $\lceil \frac{k}{t} \rceil$ of her top k items when $k = t + 1$. Further, because $|S| = 3s$, these s agents exactly cover all elements in S. To summarize, the satisfying assignment for the corresponding X3C instance is to choose the 3-sized sets whose corresponding agents didn't pick any $B_i \in B$. □

C BASELINE ALGORITHMS

We explain in details about the four algorithms we test in the experiment section.

1. **BASELINE (B)**: We assign each item to a random agent to get a random allocation. We repeat this process B times to get B random allocations and choose the one with the largest fair probability. We set $B = 4m$ in the experiment. We also tested our algorithm with B increased to $8m$, and did not observe any noticeable improvement on the algorithm's performance.
2. **LOCALSEARCH (LS)**: Two allocations are called neighbors if one allocation can be derived from the other by moving one item from some agent to another agent. In this Local Search algorithm, we start from a random allocation and iteratively move to a neighbor allocation that has a higher fair probability, until a locally optimal allocation is reached. We repeat the above process L times, each time with a new random initial allocation, and choose the final allocation with the highest fair probability. We set $L = n$ in the experiment. We also tested our algorithm with L increased to $2n$, and did not observe any noticeable improvement on the performance.
3. **MATCHING (M)**: Since in a weak SD proportional allocation, every agent needs to receive $\lfloor \frac{r}{n} \rfloor + 1$ items with ranking not worse than r for some r. We set the value of an agent getting an item with ranking r as

$\frac{1}{\lfloor \frac{r}{n} \rfloor + 1}$. For each agent i and item o, let $\mathsf{avg}(i, o)$ be the average value of i getting o.

This algorithm runs for $\frac{m}{n} + 1$ rounds and assigns at most one item to each agent each round. In round k, let R_i be the total value that agent i has received in previous rounds, then for each item o, we set the value of agent i getting item o in this round as $(1 - R_i) \cdot \mathsf{avg}(i, o)$. In addition, we remove all edges whose values are less than a certain threshold L. Then we find a maximum weight matching between agents and items and assign the item to the matched agent. Finally, we update R_i for each agent and remove all agents who receive a total value larger than another threshold U from consideration in future rounds. After finishing $\frac{m}{n} + 1$ rounds, we allocate the remaining items to the agents with less than $\frac{m}{n} + 1$ items arbitrarily.

In our experiment, we enumerate 5 different lower bound L from $\frac{1}{\lfloor \frac{m}{n} \rfloor + 1}$ to $\frac{1}{\lfloor \frac{m}{n} \rfloor + 1} + 0.1$ equidistant and 11 different upper bound U from 1 down to 0.7 equidistant. We run the algorithm for each pair of L and U, and choose the allocation with the largest fair probability among all solutions. We also test the algorithm with L higher than $\frac{1}{\lfloor \frac{m}{n} \rfloor + 1} + 0.1$ or with U lower than 0.7 and do not observe significant performance improvement.

4. **GREEDY (G)**: For each agent i and item o, we adopt notion $\mathsf{avg}(i, o)$ from the previous MATCHING algorithm to be the average value of i getting o. Then for each agent i and a set S of items, when calculating the R_i in the above MATCHING algorithm, we can find that it's always equal to $1 - \prod_{o \in S}(1 - \mathsf{avg}(i, o))$ no matter in what order the items in S are adding to agent i's bundle. Thus, we let $f_i(S) = \min(U, 1 - \prod_{o \in S}(1 - \mathsf{avg}(i, o)))$ denote the value of this agent receiving S, where U is a threshold parameter. One can check that f_i is a submodular function, and our question becomes a general submodular welfare maximization problem. We then use a greedy algorithm from Lehmann et al. [2006] which always outputs a 2-approximation solution for the submodular welfare maximization problem.

   In the experiment, we run this algorithm with 11 different values of U from 1 down to 0.7, and choose the allocation with the largest fair probability. We also tested our algorithm with the value of U less than 0.7, and found it did not improve the algorithm's performance.

References

Haris Aziz, Serge Gaspers, Simon Mackenzie, and Toby Walsh. Fair assignment of indivisible objects under ordinal preferences. *Artificial Intelligence*, 227:71–92, 2015.

Philip Hall. On representatives of subsets. In *Classic Papers in Combinatorics*, pages 58–62. Springer, 2009.

David S Johnson and Michael R Garey. *Computers and intractability: A guide to the theory of NP-completeness*, volume 1. WH Freeman San Francisco, 1979.

Benny Lehmann, Daniel Lehmann, and Noam Nisan. Combinatorial auctions with decreasing marginal utilities. *Games and Economic Behavior*, 55(2):270–296, 2006.