Dynamic Fair Division with Indivisible Items

15400 Final Report

DAVID ZENG, Carnegie Mellon University

A common problem in fair division is when we divide a set of indivisible items fairly among some number of agents. There are many offline algorithms that solve this problem. In this paper, we consider when one item arrives every step, but we need to maintain a fair allocation at each step. We consider several different settings where our algorithm is allowed to change the past, know the future, or maintain a cache of fractional items. We present several algorithms that achieve different fairness propoerties under these settings.

1 INTRODUCTION

1.1 Background

We first consider some underlying assumptions that apply to all the settings we consider. As mentioned, we are trying to allocate indivisible goods. We suppose that we have a set of n agents and and items arriving. We will refer to the i^{th} item to arrive as I_i. In addition, each agent i has a value of v_{it} for item t. We assume that agents' utilities are additive. Formally, for any set of items S, the value to agent i, $V_i(S)$ is defined as $V_i(S) = \sum_{t \in S} v_{it}$. Finally, we can define an allocation as a partition of the items into sets $A_1, \cdots, A_n$, where we given the items in A_i to agent i.

Two common measures of fairness that we care about are proportionality and envy-freeness. An allocation is envy-free if no agent envies another agent's allocation. Formally, we have $V_i(A_i) \geq V_i(A_j)$ for any two agents $i, j \in [n]$. An allocation is proportional if each agent values their own allocation at least $\frac{1}{n}$ of their value for all items. So if S is the set of all items, we must have $V_i(A_i) \geq \frac{1}{n}V(S)$ for all $i \in [n]$. Both of these fairness measures are impossible to achieve in the indivisible goods setting. To see why, just consider allocating one indivisible item among 2 agents. Thus, recent work has instead focused on the discrete version of envy-freeness called envy-freeness

up to one good (EF1), used in a paper by Lipton et al. [5]. Specifically, for any pair $i, j \in [n]$, there exists an $I \in A_j$ such that $V_i(A_i) \geq V_i(A_j \setminus I)$

In the offline setting, there always exists an EF1 allocation. Many different offline algorithms have been given for finding an EF1 allocation. We now consider the online setting, where one item arrives each step and the algorithm must immediately allocated this item. Thus, an EF1 allocation algorithm in the online setting must allocate items such that at each step, the allocation is EF1. Other fairness measures can be defined similarly for online algorithms. A recent paper by Benade et al. [2] shows that impossibility of an online EF1 allocation algorithm. In fact, envy is guaranteed to grow by $\tilde{O}(\sqrt{T/n})$ as T goes to infinity.

Thus, we consider when the algorithm is given several different abilities. The first, which can be though of as changing the past, is when the algorithm is allowed to disrupt up to d items at each step. At each step, in addition to allocating the new item that arrived, the algorithm can also move up to d items that were previously allotted between agents. A second setting, which can be thought of as knowing the future, is when only a fixed number of items will arrive, T, and the algorithm is informed of the values of each item to each agent ahead of time. However, the items still arrive one per step and the allocation at each step must be fair. Finally, we also consider a third setting, fair division with a cache. In this case, the algorithm is allowed to allocate d items fractionally at any step. These fractionally divided items may be reallocated at later steps but all other items are permanently allocated. In the case of of fractional items, let x_{it} be the amount of item t assigned to agent i. We require that $\sum_{i \in [n]} x_{it} = 1$ for all t. Then we can redefine $V_i(A_j)$ to be $V_i(A_j) = \sum_{k \in [t]} x_{jk} v_{ik}$. So the value of a fractional item to an agent is proportional to the fraction the agent got.

So at each time step an item I_t arrives. Let A_i^t be the allocation to agent i at time t. For the knowing the future setting, we also refer to $A_i^{s,t}$, which is the allocation to agent i at time t, but only considering items that arrived at or after step s. We can define the envy between i, j at time t as:

$$\text{ENVY}_{i,j}^t = V_i(A_j^t) - V_i(A_i^t)$$

Similarly, we might also be interested in:

$$\text{ENVY}_{i,j}^{s,t} = V_i(A_j^{s,t}) - V_i(A_i^{s,t})$$

Finally, we care about the overall ENVY at any time t which is:

$$\text{ENVY}^t = \max_{i,j} \text{ENVY}_{i,j}^t$$

1.2 Main Results

We obtain the following results for each of the above settings:

For the disruptions setting, we find that when there are just 2 agents, we can achieve EF1 at each step using just 1 disruptions per step. We describe such an algorithm in Theorem 2.3.

For the cache setting, we provide an algorithm that uses just $n - 1$ fractional cache items in Theorem 2.2.

Finally, in the prefix fairness setting, we find an algorithm that for 2 agents, achieves prefix EF1 fairness. We describe the algorithm in Section 3.

1.3 Related Work

Benade et al. study the purely online setting in [2]. Aleksandrov et al. study online fair division when agent utilities are in $\{0, 1\}$ [1]. They discuss applications to distributing goods for a food bank.

Fair division of indivisible items has been studied by Lipton et al. in [5]. Online algorithms with disruptions has been studied before in the context of job scheduling by Sanders et al. [6] and in the context of bin packing by Gupta et al.[4].

2 FRACTIONAL ITEM ELIMINATION

In this section, we introduce an algorithm used for both the cache setting and the disruptions setting. To motivate the algorithm, we first consider what happens when the items are divisible. Then, we can easily obtain an equitable division, or a an allocation where every agent values **every**

allocation equally, just by giving each agent $\frac{1}{n}$ of each item. However, this requires that every single item is given out fractionally. The hope wold be that equitability can be achieved with fewer items. It turns out that there always exists an allocation with at most n^2 fractional items that is equitable. A similar question to ask is how many fractional items are needed to achieve a proportional allocate. Theorem 2.1 shows that $n-1$ items are sufficient. Its also easy to see that $n-1$ items are necessary, by considering a situation with $n-1$ identical items.

THEOREM 2.1. *For any proportional allocation A, there exists a proportional allocation A' which allocates at most $n-1$ items fractionally and any integrally allocated items in A are allocated the same way in A'.*

PROOF. Consider the bipartite graph G induced by A, where edges are between two sets P, I where the set P represents the agents and the set I represents the items. Let there be an edge between (p, i) when $0 < x_{pi}$. We claim that we can find a new allocation A' such that G' has no cycles and each agent's value for their own items does not decrease.

Suppose G has a cycle of size $2n$. Since the graph is bipartite, it contains n agents and n items. Note that integrally allocated items cannot be part of the cycle. Let the agents be $\alpha_1, \cdots \alpha_n$ and items be $I_1, \cdots, I_n$. For simplicity of notation, let v_{ij} be how much agent α_i values item I_j. If we index the agents and items appropriately, we have that each agent α_i is connected to two items I_i, I_{i+1} by the cycle. Similarly, each item I_i is connected to agents α_i, α_{i-1}.

Now suppose that for each item I_i, we were to move δ_i (possibly negative), of the item from α_i to α_{i-1}. Consider the vector Δ, where Δ_i is the change in each agents' value for their own items. We can express Δ using the following linear equation:

$$\begin{bmatrix} -v_{1,1} & v_{1,2} & 0 & 0 & \cdots & 0 \\ 0 & -v_{2,2} & v_{2,3} & 0 & \cdots & 0 \\ \vdots & & & & & \vdots \\ v_{n,1} & 0 & 0 & 0 & \cdots & -v_{n,n} \end{bmatrix} \cdot \begin{bmatrix} \delta_1 \\ \delta_2 \\ \vdots \\ \delta_n \end{bmatrix} = \begin{bmatrix} \Delta_1 \\ \Delta_2 \\ \vdots \\ \Delta_n \end{bmatrix}$$

Let V be the value matrix define above. We case on whether V is invertible. Suppose it is invertible. Then there exists a vector $\vec{\delta}$ such that $V \cdot \vec{\delta} = \vec{\Delta}$, where $\vec{\Delta'} = (\varepsilon, \varepsilon, \cdots \varepsilon)$ for any choice of ε. If we choose $\varepsilon > 0$ small enough, we will have that $\vec{\delta}$ is a list of item transfers that is implementable

(after the transfer, no agent will have negative of any item) but ensures that there is at least one x_{ij} that is now equal to 0. However, after implementing the transfers $\vec{\delta}$, we have just increased each agents value for their own items and reduced number of edges by at least one.

Now, if V is not invertible, then there exists a non-zero vector $\vec{\delta}$ such that $V \cdot \vec{\delta} = \vec{0}$. Since for any constant c, $V \cdot c\vec{\delta} = \vec{0}$ we can just choose c small enough that $\vec{\delta}$ is implementable but ensures that there is at least one x_{ij} that is now equal to 0. This reduces the number of edges by at least one and agents' values for their own items has not changed.

We repeat the above until no cycles exist, which is guaranteed to happen because the number of edges is strictly decreasing each step. The final allocation must be proportional.

Any graph with no cycles has at most $|V| - 1$ edges. So if there are n agents and m items, there are at most $n + m - 1$ edges. Since each item must have an edge to at least one agent, this implies that the number of fractional items is at most $n - 1$.

The idea here is similar to the proof of Lemma 1 from [3]. □

2.1 Algorithm

Theorem 2.2. *In the cache setting, $n - 1$ fractional items are sufficient to guarantee proportionality.*

Proof. We can use Theorem 2.1 to create an algorithm that guarantees a proportional allocation for n agents with only $n - 1$ fractional items in cache. Whenever a new item arrives, we initially assign it fractionally, giving $\frac{1}{n}$ of the item to each of the n agents. We then perform cycle elimination until we have at most $n - 1$ fractional items. It is possible that we will go below $n - 1$ fractional items but that is not an issue.

We can use induction to show the allocation at each step is proportional. □

Theorem 2.3. *In the disruptions setting, when $n = 2$, 1 disruption per step is sufficient to guarantee an EF1 allocation at every step.*

Proof. Again, we use Theorem 2.1. Note that when $n = 2$, Theorem 2.1 tells us that with just 1 fractional item in cache, we can guarantee a proportional allocation. Thus, our algorithm will just round the item in cache at each step to the agent with the larger portion of the item.

We first show that the algorithm maintains an EF1 allocation. From 2.1, we know that at any step, the fractional allocation (S_1, S_2) satisfies $V_1(S_1) \geq V_1(S_2)$ and $V_2(S_2) \geq V_2(S_1)$. Let I_j be the item in the cache.

First, suppose agent 1 is assigned the cache item. Then at the end of the step, agent 1 is temporarily assigned the cache item. Agent 1 is clearly not envious because he is unenvious in the fractional division scenario is now doing even better. For agent 2, consider the allocation without the cache item, $(S_1 \setminus I_j, S_2 \setminus I_j)$. Because agent 1 got the item, we know that in the fractional allocation, agent 1 received at least half of it. So we can say $V_2(S_2 \setminus I_j) \geq V_2(S_2) - 0.5v_{2j} \geq V_2(S_1) - 0.5v_{2j} \geq V_2(S_1 \setminus I_j)$. So if agent 2 removes the cache item from agent 1's allocation, he is no longer envious. The case where agent 2 is temporarily assigned the item is similar.

Finally, note that at any step, the only item that might need to be reallocated is the item in the cache at the start of the step. Since there is only one item in cache, only one disruption is needed. □

Since we know that with 0 disruptions per step, EF1 is not achievable, we know that the above algorithm is optimal by only using at most 1 disruption per step.. However, one might ask if the problem becomes easier if disruptions are amortized. For example, while in the above setting, we are allowed 1 disruption per step, we might not always use the disruption. However, it seems possible that when we instead simply given C "flexible" disruptions to use over T steps, we might be able to use significantly less disruptions than T. However, we can also show that for $n = 2$, $\Omega(T)$ flexible disruptions are required to maintain EF1.

Theorem 2.4. *There exists an adversary strategy such that any EF1 allocation algorithm must use at least $\lfloor \frac{T}{6} \rfloor$ disruptions over T steps.*

Proof. The adversary strategy is built around a repeated 6-step sequence. Sequence s_j will describe the values for items $6j+1$ to $6j+6$. For $t = 6j+1, 6j+3, 6j+5$, we will have $v_{1,t} = v_{2,t} = 14^j$. For steps $t = 6j+2, 6j+4, 6j+6$, the values of the item will depend on the assignment of item $t-1$.

Specifically, if agent i received item $t-1$, then the adversary will set $v_{i,t} = 14^j$ and set the value of the item for the other agent to $\frac{1}{4}14^j$.

To show the lower bound, we just need to prove that any EF1 allocation algorithm must use one disruption for each 6-step sequence. Assume for sake of contradiction that for some sequence s_j, no disruptions are used during that sequence.

One important observation is that if the largest item value is v, in any EF1 allocation, all agents must have envy at most v.

Using this observation, we can show that if no disruptions are used for items $6j+1$ to $6j+6$, then for each item pair $(6j+1, 6j+2)$, $(6j+3, 6j+4)$, $(6j+5, 6j+6)$ each agent must receive exactly one item from each pair. Let $S_{6j} = \mathsf{ENVY}^{1,2}_{6j} + \mathsf{ENVY}^{2,1}_{6j}$ (also remember that we are allowing ENVY to be negative here). We can give the following lower bound $S_{6j} \geq -\sum_{t=1}^{6j} |v_{1t} - v_{2t}| = -3\sum_{i=1}^{j-1} \frac{3}{4}14^i > -\frac{9}{52}14^j$. Since the allocation at step $6j$ is EF1 and the largest item value at that step is 14^{j-1}, this implies that $\mathsf{ENVY}^{1,2}_{6j}, \mathsf{ENVY}^{2,1}_{6j} \leq 14^{j-1}$. Putting the two together, we have that $\mathsf{ENVY}^{1,2}_{6j}, \mathsf{ENVY}^{2,1}_{6j} > -\frac{9}{52}14^j - 14^{j-1} = -\frac{89}{364}14^j > -\frac{1}{4}14^j$. Thus, if item $6j+1$ is given to agent i, item $6j+2$ cannot also be given to agent i or else the allocation would not be EF1 since the other agent would have envy greater than 14^j. To show this also holds for the remaining item pairs, note that after the arrival of any item pair, the envy of each agent can only increase.

Finally, we know that at least one agent received at least 2 of the items $6j+2, 6j+4, 6j+6$. Wlog, suppose it was agent 1. Then we have $\mathsf{ENVY}^{1,2}_{6j+6} \geq \mathsf{ENVY}^{1,2}_{6j} + 2 \cdot (14^j - \frac{1}{4}14^j) > 14^j$, we means this allocation is not EF1.

Below, we give an example of a sequence assuming no disruptions. The lower bounds are non-inclusive and the brackets denote who the item was given to.

t	$6j$	$6j+1$	$6j+2$	$6j+3$	$6j+4$	$6j+5$	$6j+6$
Value of Agent 1	$\cdots$	$[14^j]$	14^j	14^j	$[\frac{1}{4}14^j]$	$[14^j]$	14^j
Value of Agent 2	$\cdots$	14^j	$[\frac{1}{4}14^j]$	$[14^j]$	14^j	14^j	$[\frac{1}{4}14^j]$
Lower bound for $\mathsf{ENVY}^{1,2}_t$	$-\frac{1}{4}14^j$	$-\frac{5}{4}14^j$	$-\frac{1}{4}14^j$	$\frac{3}{4}14^j$	$\frac{1}{2}14^j$	$-\frac{1}{2}14^j$	$\frac{1}{2}14^j$
Lower bound for $\mathsf{ENVY}^{2,1}_t$	$-\frac{1}{4}14^j$	$\frac{3}{4}14^j$	$\frac{1}{2}14^j$	$-\frac{1}{2}14^j$	$\frac{1}{2}14^j$	$\frac{3}{2}14^j$	$\frac{5}{4}14^j$

□

3 ENVY CYCLE ELIMINATION

We now turn to the setting where the algorithm the stetting where we must allocate items step by step but know the future (prefix fairness). Lipton et al. introduce envy cycle elimination as a EF1 allocation algorithm in the offline setting [5]. We can adapt that algorithm into one that works well in the prefix fairness setting.

Algorithm. Suppose we are given T items to allocate among 2 agents. Our algorithm will simulate a modified envy cycle elimination protocol to produce an final allocation A for all T items. Specifically, at each step, we will assign the new item to an arbitrary unenvied agent. Next, we will keep track of the last time s in which the allocation was envy-free. If at step t there is ever an envy cycle when considering at items assigned after time s, we will swap all items from $s+1$ to t along the cycle. For example, if agent 1 envies agent 2 and vice versa, we will give all items agent 1 recieved during step $s+1$ to t to agent 2 and vice versa.

We apply our above algorithm to the case when $n = 2$.

Lemma 3.1. *For any $t \leq T$, let s be the last step before t in which allocation A was envy-free. Let $M_1 = \max\{v_{1j} : j \in [s+1, t]$ and agent 2 owns I_j in $A\}$. Let M_2 be defined similarly. Then $\mathsf{ENVY}_{1,2}^{s+1,t} \leq M_1, \mathsf{ENVY}_{2,1}^{s+1,t} \leq M_2$. In other words, this section of the allocation is EF1.*

Proof. We will show $\mathsf{ENVY}_{1,2}^{s+1,t} \leq M_1$. The proof for $\mathsf{ENVY}_{2,1}^{s+1,t} \leq M_2$ is similar.

Let A' be the allocation where each item is given to the agent it was first assigned to (before any envy cycle elimination). Let $A[x, y]$ refer to the allocation of items x to y. Since we swap all items that arrived after the last envy-free step, there are two possibilities for the final allocation of items $s+1$ to t in A: either $A[s+1, t] = A'[s+1, t]$ or $A[s+1, t]$ is the opposite of $A'[s+1, t]$. Note that the first case occurs when the allocation becomes envy-free without the need for a swap and the latter occurs when the allocation becomes envy-free because of a swap.

For the following, let any ENVY values refer the envy under A'. So we want to show $\mathsf{ENVY}_{1,2}^{s+1,t} \leq M_1$ if no swap occurred and $\mathsf{ENVY}_{1,2}^{s+1,t} \geq -M_1$ if there was a swap.

Suppose no swap occurred. Now assume for sake of contradiction that $\mathsf{ENVY}_{1,2}^{s+1,t} > M_1$ and consider the first t' for which $\mathsf{ENVY}_{1,2}^{s+1,t'} > M_1$. Then at t', we must have given agent 2 an item.

However, the value of this item to agent 1 is bounded by M_1, so before this step, agent 1 must have envied agent 2. But then we just gave an item to an envied agent, which contradicts the algorithm.

Now suppose a swap did occur. We want to show show $\text{ENVY}_{1,2}^{s+1,t} \geq -M_1$. Assume for sake of contradiction $\text{ENVY}_{1,2}^{s+1,t} < -M_1$ and let t' be the first step where $\text{ENVY}_{1,2}^{s+1,t'} < -M_1$. Then we know that at step t', we assigned the new item to agent 1. Since a swap occurred, M_1 is actually the maximum value agent 1 has for their own items. Together, this implies agent 1 was not envious right before step t'. Since our algorithm gave the new item to agent 1, agent 2 also was not envious right before step t'. Therefore, the allocation of items from $s+1$ to $t'-1$ is envy-free. Hence, the allocation of items from 1 to $t'-1$ is also envy-free. But since s is the last step before t in which A was envy free, we should then have $s = t'-1$. This is a contradiction since we know $s+1 \leq t'-1$. □

Theorem 3.2. *The above algorithm generates an allocation A that is prefix EF1.*

Proof. Using the definition of s from Lemma 3.1, for any t, we decompose $\text{ENVY}_{1,2}^{t} = \text{ENVY}_{1,2}^{s} + \text{ENVY}_{1,2}^{s+1,t}$ and similarly for $\text{ENVY}_{2,1}^{t}$. By definition of s, $\text{ENVY}_{1,2}^{s} \leq 0$ and $\text{ENVY}_{1,2}^{s+1,t} \leq M_1$ so $\text{ENVY}_{1,2}^{t} \leq M_1$. Since M_1 is the value of some item assigned to agent 2, agent 1 is not envious of agent 2 after removing this item. Therefore, allocation is prefix EF1. □

4 DISCUSSION

The above results either achieve proportionality with n agents, or EF1 with just 2 agents. An important area where there are still few results is when there are more than 2 agents but the goal is to achieve envy-freeness. For example, in the disruptions setting, there are known allocation algorithms that achieve envy-freeness up to value $O(n\sqrt{\log n})$ (assuming item values are between 0 and 1) that use n^2 disruptions per step. However, there still do not exist any non-trivial lower bounds, so this area seems like a likely area of improvement.

REFERENCES

[1] Martin Aleksandrov, Haris Aziz, Serge Gaspers, and Toby Walsh. Online fair division: analysing a food bank problem. *CoRR*, abs/1502.07571, 2015.

[2] Gerdus Benade, Aleksander M. Kazachkov, Ariel D. Procaccia, and Christos-Alexandros Psomas. How to make envy vanish over time. 2018.

[3] Anna Bogomolnaia, Hervé Moulin, Fedor Sandomirskiy, and Elena Yanovskaya. Dividing goods or bads under additive utilities. *CoRR*, abs/1608.01540, 2016.

[4] Anupam Gupta, Guru Guruganesh, Amit Kumar, and David Wajc. Fully-dynamic bin packing with limited repacking. *CoRR*, abs/1711.02078, 2017.

[5] R. J. Lipton, E. Markakis, E. Mossel, and A. Saberi. On approximately fair allocations of indivisible goods. In *Proceedings of the 5th ACM Conference on Electronic Commerce*, EC '04, pages 125–131, New York, NY, USA, 2004. ACM.

[6] Peter Sanders, Naveen Sivadasan, and Martin Skutella. Online scheduling with bounded migration. *Math. Oper. Res.*, 34(2):481–498, May 2009.