Characterisaton of EF allocations for k-approval problem
MTD350 End-Term Presentation

Rakshitha - 2021MT10904
Ayush Gupta - 2021MT10697
Supervisor: Prof.Ashutosh Rai

May 7, 2024

Fair Division

- The Fair Division problem, situated within Computational Social Choice, intersects disciplines like Economics and Computer Science. It revolves around fairly allocating resources among agents without envy, a condition where no agent prefers another's share.
- Originally named the "Cake Cutting Problem," it tackled divisible goods' fair division. Efficient algorithms have been developed for computing envy-free allocations, given various utility functions.
- Recent focus has shifted to indivisible goods, posing challenges due to binary allocations. Achieving envy-freeness becomes intricate without the option of fractional allocations.
- Our project delves into literature on envy-free allocation of indivisible goods and introduces a new problem for investigation, aiming to contribute to fair resource allocation solutions.

Current Work on EF Allocations

- Envy free allocations may not exist for the indivisible problem setting.
- One can very quickly show that it is NP-hard to determine if there exists an EF allocation, even with two agents with identical, additive, monotone valuations.
- Determining whether a given instance admits an envy-free allocation is NP-complete even for binary utilities.
- There exists algorithms like Round Robin Algorithm(given additive valuation functions) and Envy Cycle Eliminations(given monotone valuation functions) for finding EF1 allocations.

Problem Statement

- We study the special case of the general fair allocation problem and try to find tractable results for the special case, the "k-approvals problem".
- We define an instance (N, M, V) of the k-approvals problem by:
  1. $N = [n]$, a set of n agents,
  2. M, a set of m indivisible goods
  3. $V = (v_1, v_2, \ldots, v_n)$, a set of n valuation functions
- Additional constraints:
  1. Every v_i is an additive valuation function.
  2. $v_i(\{j\}) \in \{0, 1\}\ \forall i \in N$ and $j \in M$
  3. $v_i(M) \leq k\ \forall i \in N$.
- All of the additional constraints imposed together can be seen as the "k-approvals problem", in the sense that each agent gives her approval for utmost k items in M and denies the rest, where $v_i(j) = 1$ is treated as an approval.
- **Aim: To find if an envy free allocation exists for a given instance of the problem tractably. We are majorly trying to look at FPT algorithms in the introduced parameter, k.**

Brute Force Analysis

- The most naive approach to characterising the existence of an EF allocation would be to check all possible allocations (partitions).
- For a setting with n agents and m items, each of the m items could go into any of the n agents' bundles. Hence, there are n^m possible partitions. Thus, we will require $O(n^m)$ queries, each query needing $O(n^2)$ time.

A slight improvement

- In the case where each agent values exactly k items,if there exists an EF allocation then every agent i must have value $v_i(A_i) \geq \lfloor \frac{k_i}{n} \rfloor$ where $k_i = v_i(M)$.

Proof.

Let $A = (A_1, A_2, ...A_n)$ be an *EF* allocation for given instance (N, M, V). To the contrary,suppose agent i is such that $v_i(A_i) \leq \lfloor \frac{k_i}{n} \rfloor$.This implies:

$$\sum_{j!=i} v_i(A_j) = v_i(M) - v_i(A_i) > k_i - \lfloor \frac{k_i}{n} \rfloor$$

By the rule of averages,$\exists \quad j! = i$ such that $v_i(A_j) \geq \frac{k_i - \lfloor \frac{k_i}{n} \rfloor}{n-1}$.Therefore we have:

$$v_i(A_j) \geq \frac{n * \frac{k_i}{n} - \frac{k_i}{n}}{n-1} = \frac{k_i}{n} \geq \lfloor \frac{k_i}{n} \rfloor \Rightarrow v_i(A_j) \geq v_i(A_i)$$

We get a contradiction. □

In particular for k=2(contd)

- For the k=2 case, if any agent has a private good (that agent is the only agent who prefers the good), then the good can be allocated to that agent, and both the agent and the good can be removed from the picture. We can assure that the removed agent is envy-free.
- Hence, let us consider a setting with no agents with private goods.Every agent has atleast two preferred goods and every item has atleast two agents preferring it or in other words $2n \geq 2m \implies n \geq m$.
  1. **Case-01:** $n = m$
     Every item has exactly two agents who prefer the item. We get a 2-bipartite graph and hence EF allocation exists.
  2. **Case-02:** $n > m$
     This will be the case where in every allocation that we look at, there will be at least one agent who has an empty allocation. Hence, no allocation can be envy free. (see lemma 3.1 given below)

Clever Manifestation of the search space

Lemma

In any envy free allocation, each agent has at least one of her preferred goods in her bundle.

Proof.

We see that every agent has at least one preferred good in the item set M. If the agent i has none of her preferred goods in her bundle, then her self valuation is zero. But due to non-wastefulness of allocation, some agent must have a positive valuation with respect agent i (due to having one of agent i 's preferred goods). Hence agent i is no longer envy free. □

Exploration of Search Space(contd)

We will use the lemma proved above to get a reduction in our search space. We look at partitions using the method of branching.

- **Case 1**: $m < n$.

Corollary

If $m \leq n$ then there exists no EF allocation.

Proof.

This follows from the Lemma 3.2 that every agent values at least one item. □

- **Case 2**: $m = n$
  This is equivalent to finding a bipartite matching in a graph. We can hence use Ford Fulkerson which will help us check if EF allocation exists.

Exploration of Search Space(contd)

- **Case 3**: $m > n$
  Suppose every agents prefers atmost k items. We present an algorithm which generates partitions via recursion:
  1. Check if agent 1 values exactly k items. If not, move to the next agent.
  2. Let agent i be the first agent that value exactly k items. Allot agent i one of any of her preferred items.
  3. Remove this preferred item from the list of preferred items of all other agents.
  4. Find the next agent who values exactly k items.
  5. Repeat the above process until all agents value at most $k-1$ items. And run the partition generator on $(k-1)$

An approach using bipartite graphs and network flows

- We see, as we explore the branching approaches, that the problem bears an uncanny resemblance to the matching problem, except here it is going to be a two sided matching instead.
- We can generate a bipartite graph, with one part being the agents, and the other part being the items. We can connect an agent and an item by an edge if the agent prefers the item. This can also be converted into a flow.
- We can work on investigating how an EF allocation looks like through the lens of a network flow. This would be an interesting problem.

An approach using Integer Linear Programs

- Let:

$$x_{ij} = \begin{cases} 1 & \text{if agent } i \text{ gets item } j \\ 0 & \text{otherwise} \end{cases} \qquad \mu_{ij} = \begin{cases} 1 & \text{if agent } i \text{ approves item } j \\ 0 & \text{otherwise} \end{cases}$$

- Our objective function would be to maximize the minimum of negative of "enviness" between agents.Let $min_{i,k}(\sum_j x_{kj} * \mu_{ij} - \sum_j x_{ij} * \mu_{ij}) = z$.We would then obtain:

$$max \quad z$$

subject to:

1. $\sum_j x_{kj} * \mu_{ij} - \sum_j x_{ij} * \mu_{ij} \leq 0 \quad \forall \quad i, k \in [n]$
2. $z \leq \sum_j x_{kj} * \mu_{ij} - \sum_j x_{ij} * \mu_{ij} \quad \forall \quad i, k \in [n]$
3. $\sum x_{ij} = 1 \quad \forall \quad j \in [m]$
4. $x_{ij} \in \{0, 1\} \quad \forall \quad i \in [n], j \in [m]$

An approach using Integer Linear Programs(contd)

- We see that the matrix associated with the ILP is very structured. Firstly, it is a 0-1 matrix.
- We can also impose restrictions on the sum of rows and columns of the matrix based on the parameter k.
- The matrix will be sparse matrix due to the parameter. Hence, we speculate that this property could be used to our advantage.

Conclusion

- We are still not been able to find an FPT algorithm which runs exponential in parameter k.Hence it might be possible that the given instance we considered is NP complete.
- We therefore plan to explore possible reductions from different NP complete problems to our case.
- The idea around Integer Linear Programs could prove very useful but the equations involved are a bit more challenging and hence require time.We therefore plan to continueour search in this direction as well.

References

- https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=296112b59023726acbc7d9e01ec56cc40a56cff7-Liptonet.al - Envy Cycle Elimination
- https://arxiv.org/pdf/2211.06458 - On Local Envy Freeness
- https://events.csa.iisc.ac.in/algorithmic-fairness-2023/ - A Workshop on Algorithmic Fairnes, IIIT Hyderabad, Dec 2023
- https://arxiv.org/pdf/1908.01669 - On Efficient Fair Division with Minimal Sharing
- https://dl.acm.org/doi/abs/10.1145/3328526.3329649 - On using ILPs for Fair Division