CS 6002: Selected Areas of Mechanism Design **Jan-Apr 2025**

Lecture 9: Fair Division of Indivisible Objects

Lecturer: Swaprava Nath *Scribe(s): Deeksha Dhiwakar*

9.1 Recap

In the last lecture, we studied fair division of indivisible objects in the additive valuations setting. We concluded that EF cannot be satisfied, but EF-1 can always be satisfied using the Round Robin (RR) algorithm.

9.2 Non-additive Valuations Setting

An example of such a setting could be the allocation of bundles, where the value of an item depends on the bundle that contains it. This general setting is represented by the equation

$$v_i(\{A, B\}) \neq v_i(A) + v_i(B)$$

In this lecture, we focus only on monotone valuations.

Monotone Valuation: A valuation function is monotone if $\forall i \in N, v_i(S) \leq v_i(T)\ \forall S \subseteq T \subseteq M$

Does the round robin algorithm satisfy EF-1 in this setting?

No. We can construct a counter example as follows.

	A	**B**	**C**	**D**	**E**	**{B,D}**	**{A,C}**	**{A,E}**	**{C,E}**	**{A,C,E}**	...
A1	**10**	8	**6**	5	**1**	..	..	..	..	20	...
A2	10	**8**	6	**5**	1	9	10	11	12	13	...

In this modified setting, each agent picks the object that maximizes their marginal increase in valuation. The RR allocation is shaded in the table and gives the allocations $A1 : \{A, C, E\}(20)$, $A2 : \{B, D\}(9)$. Clearly $A2$ envies $A1$. Even upon removing any of the objects from $A1$'s bundle, $A2$ still values it more than $\{B, D\}$, meaning this algorithm is not EF-1.

9.3 Envy Cycle Elimination Algorithm

Before describing the algorithm, we define some terms.

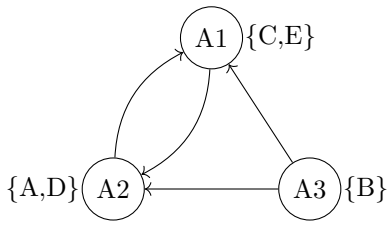
9.3.1 Envy Graph

Definition 9.1 (Envy Graph) *AN envy graph is a graph in which each vertex represents an agent along with its partial allocations. There is a directed edge from vertex i to vertex j if agent i envies agent j under the current partial allocation.*

Note: Unlike the graphs in the TTC algorithm, there can be multiple outgoing edges from a vertex, since here we draw edges to **all** envied bundles, not just the favourite bundle.

Consider the valuations below where the allocations are represented by the shaded cells. We assume the valuations are additive to simplify the table. The envy graph is as follows:

	A	**B**	**C**	**D**	**E**
A1	4	2	**5**	4	**1**
A2	**1**	0	5	**1**	1
A3	1	**1**	5	1	1



Definition 9.2 (Envy Cycle) *An envy cycle is a directed cycle in the envy graph.*

In the above example, $A1 - A2$ form an envy cycle.

Definition 9.3 (Source Node) *Source node is a node in the envy graph that does not have any incoming edges.*

In the above example, $A3$ is a source node.

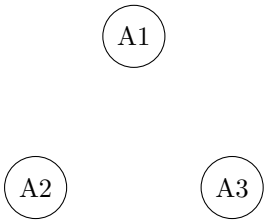
9.3.2 ECE Algorithm

	A	**B**	**C**	**D**	**E**
A1	0	2	0	1	4
A2	1	2	5	10	1
A3	1	4	2	10	1

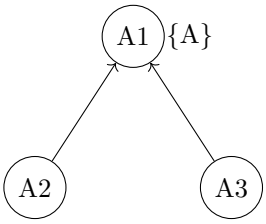
We illustrate the above algorithm with an example. Consider the valuations shown on the left. Note that there is no assumption of additive valuations; we assume we have an oracle using which we can find the valuation of any bundle.

Algorithm 1 Envy Cycle Elimination

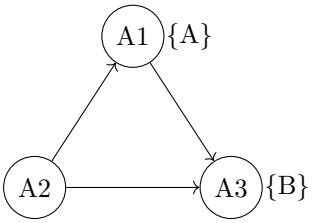
```
1: while there is an unallocated object X do
2:     there is an unallocated object X
3:     if the envy graph has a source vertex v then
4:         Allocate X to v
5:     else
6:         Resolve cycles until a source vertex shows up
7:     end if
8: end while
```



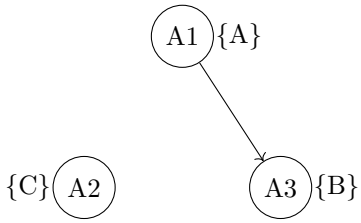
(a) Initially, no objects allocated. Source nodes: $A1$, $A2$, $A3$.



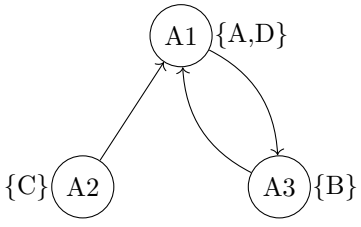
(b) Round 1: Unallocated object A given to source node $A1$. Source nodes: $A2$ and $A3$.



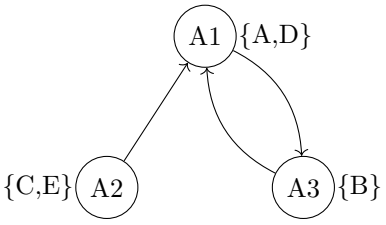
(c) Round 2: Unallocated object B given to source node $A3$. Source node: $A2$.



(d) Round 3: Unallocated object C given to source node $A2$. Source nodes: $A1$, $A2$.



(e) Round 4: Unallocated object D given to source node $A1$. Source node: $A2$.



(f) Round 5: Unallocated object E given to source node $A2$. Note that we did not resolve the cycle from the previous round because we still had a source node.

Final allocation: $A1 : \{A, D\}, A2 : \{C, E\}, A3 : \{D\}$.

An interesting point to note is that although we can clearly achieve a more optimal allocation by resolving the envy cycle at the end, the ECE algorithm does not resolve it. As we will show shortly, even in spite of this the algorithm satisfies several desirable properties.

Does the algorithm terminate in poly time?

Observation: There are m objects that are to be allocated. Since each round may consist of envy cycle resolutions, if we can show that resolving an envy cycle terminates in finite polynomial steps, we are done.

Theorem 9.4 *After resolving any envy cycle, the number of edges in the envy graph strictly decreases.*

Corollary 9.5 *The number of cycle resolutions is upper bounded by* $O(n^2)$

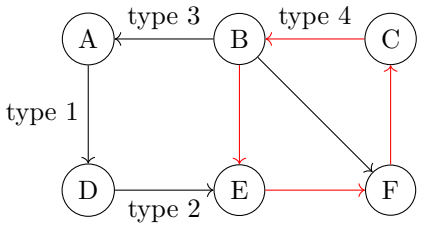
Proof: In any envy graph with an envy cycle (there could be multiple cycles but we focus on the one we intend to resolve in the next round), we have the following types of edges:

- **Type 1:** Neither the source nor the destination are in the cycle.
- **Type 2:** The destination is in the cycle but not the source.
- **Type 3:** The source is in the cycle but not the destination.
- **Type 4:** Both the source and the destination are in the cycle.

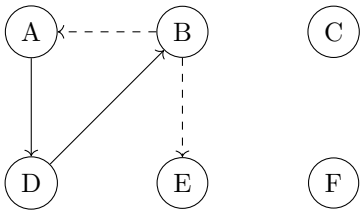
After resolving the envy cycle, the number of edges of each type changes as follows:

- **Type 1:** All such edges are unaffected. Number stays the same.
- **Type 2:** The destination changes to the new owner of the bundle that the source was envious of. Number stays the same.
- **Type 3:** Due to reallocation at source, some may disappear. Number does not increase.
- **Type 4:** Some will disappear due to cycle resolution. Edges not part of the cycle may disappear. Number strictly decreases.

To illustrate the effect of cycle resolution on the different types of edges, consider the following example. Dashed lines in the second figure indicate that the edge may or not be present.



(a) Before cycle resolution



(b) After resolving red cycle

Combining the observations on all four edge types, we see that the number of edges in the envy graph strictly decreases after a cycle resolution. This proves our theorem, and hence proves that the ECE algorithm terminates in poly time. ■

Does the algorithm give an EF-1 allocation?

Yes. We prove this by induction on the allocation rounds.

Base Case: Initially none of the objects are allocated i.e., all agents have empty allocations. None of them envy another, making this EF-1.

Induction Assumption: We assume that the allocation is EF-1 at round k, and use this to prove that EF-1 holds at round $k+1$ as well.

Proof for round $k+1$:

- **Case 1:** There is a source vertex in the envy graph. Then we allocate the object to this agent. Since it had no incoming edges (was not envied by anyone) before, EF-1 is preserved as we can remove the object to obtain a bundle that no agent envies.
- **Case 2:** There is no source vertex, and we resolve cycles. We consider all four types of edges defined before. In each case we denote the agent at the source vertex by A and the bundle at the destination (which they envy) by B.
  - **Type 1:** These edges are unaffected by cycle resolutions, hence do not affect EF-1.
  - **Type 2:** The bundles in the cycle (including B) are just shifted around, so we can still remove the same object as before from B to satisfy EF-1 for A.
  - **Type 3 and Type 4 (non-cycle edge):** After cycle resolution, A gets a more preferred bundle than her previous allocation, meaning she is either not envious of B anymore, or we can still remove the same object as before from B to obtain a bundle which she does not envy.
  - **Type 4 (cycle edge):** A gets B, so she is no longer envious of it.

  In all four cases, EF-1 is preserved.

Hence proved that the ECE algorithm returns an EF-1 allocation.

9.4 Measures of Fairness

So far, we have considered PROP, EF and EF-1. Several other measures of fairness also exist, such as MMS (Max Min Share).

However, fairness alone is not enough; the empty allocation is EF but not practically desirable. We also require **maximality** (all objects should be allocated).

We now introduce a new desirable quality we want our allocation rules to possess- Pareto Optimality.

Pareto Optimality: An allocation A is Pareto Optimal if $\nexists B \neq A$ such that

$$\forall i \in N,\ v_i(B_i) \geq v_i(A_i)$$

$$\exists j \in N,\ v_j(B_j) > v_j(A_j)$$

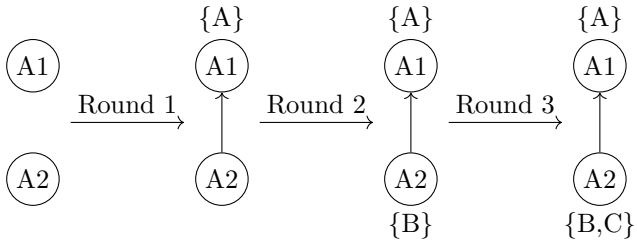
Is Round Robin allocation in an additive valuation setting PO?

	A	**B**	**C**	**D**	**E**
A1	**4**	3	**1**	1	**1**
A2	5	**2**	1	**1**	1

No. Consider the example on the left, with the RR allocation shaded. The allocation $A1 : \{A, B\}$, $A2 : \{C, D, E\}$ is weakly better for $A2$ and strictly better for $A1$, meaning RR is not PO.

Is the Envy Cycle Elimination algorithm PO?

	A	**B**	**C**
A1	4	3	1
A2	5	2	1



No. The above figure shows the ECE allocation for the valuations in the table on the left. This is Pareto dominated by the allocation $A1 : \{B, C\}$, $A2 : \{A\}$, meaning ECE is not PO.

Idea: Start with any EF-1 allocation and do Pareto improvements that maintain EF-1

	A	**B**	**C**	**D**	**E**
A1	**10**	8	**6**	4	**2**
A2	8	**4**	$4-\epsilon$	**3**	2

We start with the EF-1 allocation shown in the table, $A1 : \{A, C, E\}$, $A2 : \{B, D\}$. This is Pareto dominated by the allocation $A1 : \{B, C, D, E\}$, $A2 : \{A\}$. However upon making this improvement we lose the EF-1 property.

We need a different mechanism that satisfies both EF-1 and PO, which we will study in the next lecture.