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

Lecture 11: Bads and Mixed Items

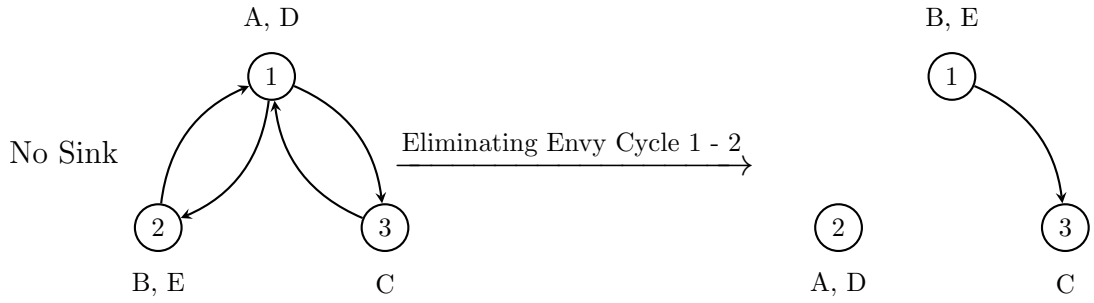
Lecturer: Swaprava Nath *Scribe(s): Atharva Tambat and Kanishk Sharma*

11.1 Recap

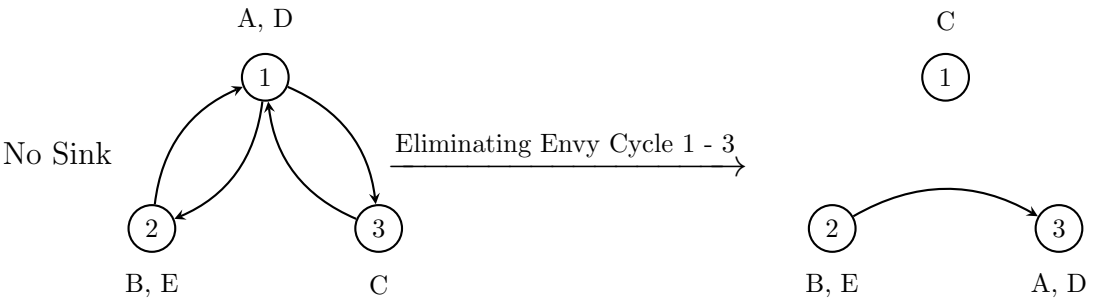
In the previous class we considered Envy Cycle Elimination (ECE) algorithm on the following example containing chores, with ties broken in the order $1, 2, 3$ and the chore assignment order being A, B, C, D, E.

	A	B	C	D	E
1	-1	-3	-2	-9	-3
2	0	-1	-4	-1	-2
3	0	-2	-1	0	-4

We saw that ECE might yield the following allocation which is not EF-1.



Agent 1 is envious of agent 3's allocation even after dropping B or E individually. In this example the original bundle of 1 had one large negative chore, which she could drop and be envy free. The exchanged bundle does not have any such chore. Notice, the following envy cycle elimination preserves EF-1 property of the original envy graph.



Thus, arbitrary envy cycle elimination does not work. The main observation here is that 1 envies 3 more than 2 i.e. 1 would be happier if he has 3's bundle rather than 2's bundle. Thus, the second elimination reduced 1's envy by the maximum amount. This hints at a new algorithm in case of chores.

11.2 Max Envy Cycle Elimination (Top Trading ECE)

First we define max-envy graph as follows.

Definition 11.1 *A **max-envy graph** is a directed graph constructed based on agents' preferences over allocated bundles.*

Formally, let N be the set of agents, and let $\{B_j\}_{j\in N}$ be the set of allocated bundles. Define the preference relation of agent i over these bundles as $\succ_i$. The set of agents to whom i is envious is given by:

$$S_i = \{j \in N \mid B_j \succ_i B_k \text{ for all } k \in N \setminus \{j\}\}.$$

*A **max-envy edge** is a directed edge from i to each $j \in S_i$, meaning an agent points to all agents who have the most preferred allocated bundle.*

*The **max-envy graph** is then defined as the directed graph $G = (N, E)$, where the vertex set N consists of all agents, and the edge set E consists of all max-envy edges.*

The following is the max-envy cycle elimination algorithm.

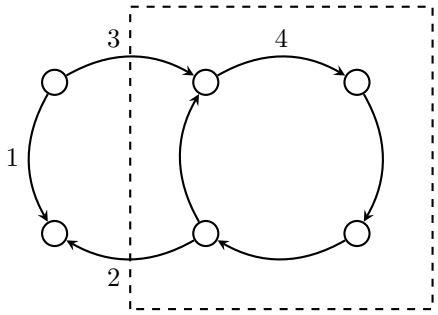
Algorithm 1 Max Envy-Cycle Elimination Algorithm (Top-Trading EC)

```
while there exists an unallocated chore do
    if the max-envy graph has a sink vertex then
        Assign the chore to that agent.
    else
        Resolve any max-envy cycle until a sink appears.
        Assign the chore to the sink agent.
    end if
end while
```

Note: If there is no sink in the max-envy graph, then there must exist a cycle in the graph.

11.2.1 Why does Max-Envy Cycle Elimination Converge?

Consider the following representative max-envy graph at some step in the algorithm. It has 4 types of edges number 1-4. Whenever we are eliminating a max-envy cycle, let us analyze what happens to the number of edges of each type.



- **Type 1 edges:** Where the source and target node both are outside the max-envy cycle being eliminated. These type of edges are not affected by max-envy cycle elimination hence the number of such cycles remain the same.
- **Type 2 edges:** Where the source node, say i, is part of the max-envy cycle, but the target node, say j, is outside the max-envy cycle. Since i gets her most preferred bundle - she does not envy j - because the bundle she received in the max-envy cycle elimination is of the same utility as j's bundle.
- **Type 3 edges:** Where the target node, say j, is part of the max-envy cycle being eliminated, but the source node, say i, is not. Since, j' s bundle will go to some other node in the max-envy cycle, say k, this type of edge will just change the target node from j to k. Thus the number of such edges remains the same.
- **Type 4 edges:** Where both the source and target nodes are in the max-envy cycle being eliminated at a certain step. After the elimination, such edges disappear.

Thus, the number of edges in the graph strictly decreases in each elimination step. After the allocation of all the new objects only a finite number of new edges can be added to the max-envy graph. Since in each envy-cycle elimination step, the number of edges strictly decreases, thus the algorithm terminates. In fact, the algorithm terminates in polynomial time.

11.2.2 Why is Max-Envy Cycle Elimination EF1?

We establish the correctness of the algorithm by using induction on the number of allocation rounds.

In the beginning, no objects have been assigned, making the initial state EF-1.

Now, assume that after k rounds, the allocation remains EF-1. We need to show that this property continues to hold in round $k + 1$.

There are two possible scenarios:

- **Case 1: If a sink vertex exists in the max-envy graph -** we allocate the chore to this agent (say i). Since node i is a sink - agent i has her most preferred bundle. After the new chore has been allocated, she can just drop this chore and she will be not envy any other agent. Since this new object is a chore, therefore other agents who could drop an item from their bundle to be envy free to i's allocation can still drop that item and be envy free.
- **Case 2: If no sink vertex exists, it means the graph contains at least one max-envy cycle.** In this case, we apply the max-envy cycle elimination process and analyze the effect on the envy of agents inside and outside the max-envy cycle being eliminated. Consider the representative max-envy graph given above:

- **Agents inside the cycle**: Their envy disappears since they are getting their most preferred bundle—this has the highest value to them. Hence, they won't envy others.
- **Agents outside the cycle**: Their outgoing envy edge can shift nodes, but the magnitude of their envy does not change. Also, they retain their bundles. Hence, the item they could drop earlier to ensure EF (Envy-Freeness) still remains with them.

Thus the new allocation at $k + 1^{th}$ step is also EF-1.

Conclusion: Hence, by induction, we have proved that the Max-Envy Cycle Elimination (ECE) algorithm returns an EF-1 allocation.

11.3 Recap of Algorithms for Chores

- **Chores with additive valuation:** Round Robin algorithm is EF-1 for chores with additive valuation.
- **Chores with monotone valuation:** Max-Envy Cycle Elimination algorithm is EF-1 for chores with monotone valuation.

11.4 Mixed Items

There can be some items that are good for some people and chores for other people. Such items are called mixed items. E.g. noise level of room, ISMP/ DAMP mentorship etc. Since mixed items allocation problem is a generalization of goods/chores allocation, we cannot ensure Envy Free allocation of mixed items. Let us formalize the definition of EF-1 (Envy Freeness Upto 1 Item) in the case of Mixed Items.

Definition 11.2 (Envy-Freeness up to One Item (EF-1)) *An allocation $A = (A_1, A_2, \ldots, A_n)$ is said to be Envy-Free up to One Item (EF-1) if for all agents i, j, there exists an item $x \in A_i \cup A_j$ such that:*

$$v_i(A_i \setminus \{x\}) \geq v_i(A_j \setminus \{x\}).$$

Note in the above x can either belong to A_i or A_j since A_i & A_j are disjoint. Informally, this means that if:

- *If x is a **good**, check envy-freeness after removing it from the other agent's bundle.*
- *If x is a **chore**, check envy-freeness after removing it from our own bundle.*

Next we want to device algorithms that are EF-1 for mixed items. Let us try out the algorithms that are already discussed.

11.4.1 Round Robin for Mixed Items with Additive Valuations

Simple round robin fails for mixed item allocation under additive valuations. Consider the following example with the order in the round robin as $1, 2$. Then the allocations under Round Robin algorithm are shown circled in the table below.

	A	B
1	(1)	-1
2	1	(-1)

Table 11.1: Counterexample for Round Robin for Mixed Items with Additive Valuation

In the above counterexample, 2 envies 1 even after A is removed from 1's bundle or B is removed from 2's bundle. To address this we consider a variant of the round robin algorithm - Doubly Round Robin Algorithm.

11.5 Double Round Robin Algorithm for Mixed Item Allocation with Additive Valuations

Under this setting we divide all possible mixed items into two types:

Definition 11.3 *The set of items can be classified into **positive** and **negative** items based on their valuation by agents.*

- ***Positive Items:*** *An item is considered positive if at least one agent assigns it a strictly positive value. Formally, an item g belongs to the positive set if:*

$$\exists j \in N \text{ such that } v_j(g) > 0$$

This means at least one agent considers it a good.

- ***Negative Items:*** *An item is considered negative if all agents assign it a non-positive value, meaning it is either a chore or neutral for everyone. Formally, an item g belongs to the negative set if:*

$$\forall j \in N, \quad v_j(g) \leq 0$$

This includes items that are disliked by all agents or have zero value for everyone.

Examples of ***positive items*** and ***negative items*** are given below

	A	B	C	D	E
1	-4	-1	-2	2	-4
2	0	-1	-5	-2	-1
3	-4	-2	-5	0	2
	Negative Items			Positive Items	

Table 11.2: Valuation Matrix with Positive and Negative Items

The following is an extension of the Round Robin Algorithm for handling Mixed Items.

- Fix an ordering $a_1, a_2, \ldots, a_n$ over the agents.
- If the number of negative items is not a multiple of n, add dummy items (with value zero for all agents) to make it a multiple of n.
- **Phase 1:** Run round-robin allocation of the negative items in the order $a_1, a_2, \ldots, a_n$. The dummy items get allocated first as a consequence, unless someone has zero value for the real items i.e., each agent picks their favorite unallocated negative item.
- **Phase 2:** Once all negative items are allocated, run round-robin allocation of the positive items in the reverse order $a_n, a_{n-1}, \ldots, a_2, a_1$, with the caveat that agents are allowed to skip their turn if none of the remaining items are of positive value to them.

The intuition behind running round robin two times, with the order reversed the second time is as follows. To simplify the allocation process, let us consider the allocation of goods and chores separately. If chores are allocated in an order in which i comes before j, then possibly j might envy i. Then during distribution of goods, surely j must be allowed to choose before i, so that, j does not envy i's allocation.

11.5.1 Does Doubly Round Robin Algorithm Converge?

Yes, Doubly Round Robin Algorithm terminates. The proof for the same is given below.

Proof: The Phase 1 of the Doubly Round Robin Algorithm goes on for maximum $N \left\lceil \frac{I_{neg}}{N} \right\rceil$ (where I_{neg} is the number of negative items). Since the agents are forced to choose - and do not skip their chance.

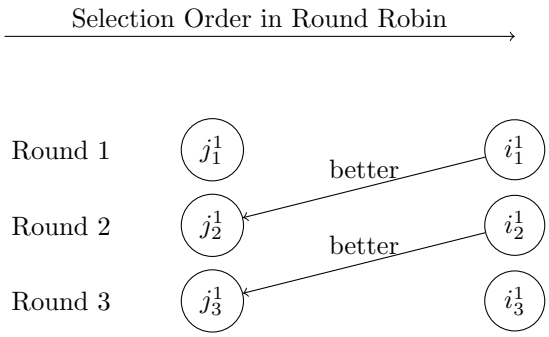
In Phase 2 of the algorithm, the agents are allowed to skip their turn, if the positive item is a chore for them. So, there is a risk of infinite skipping and the algorithm could never terminate. However, by the definition of positive items, g, $\exists j \in N$ such that $v_j(g) > 0$. In the reverse order, in a round robin fashion, when the choice comes to the first agent j for which some positive items has a strictly positive valuation - she will chose that item, and the number of items strictly decreases by at least 1 in a single round of the round robin. Therefore, infinite skipping loops cannot occur. Thus, Phase 2 of the algorithm also terminates. ■

11.6 Is Doubly Round Robin Allocation EF-1?

Yes, the Double Round Robin Algorithm is EF-1 for additive valuations.

Proof: Phase 1 of the Doubly Round Robin Algorithm is EF-1.

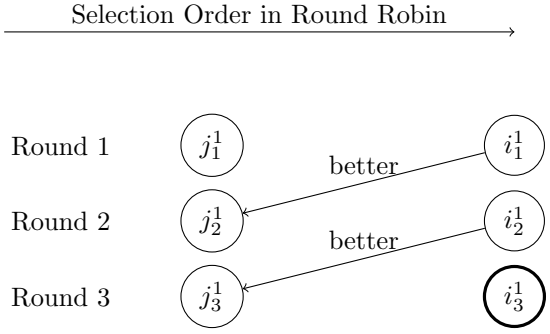
- **Case 1:** If i picks before j, then, at the end of Phase 1, i's bundle is more valuable than j's (up to one chore).
- **Case 2:** If i picks after j (example depicted below), since the items are negatives i can just drop the least valuable chore from its allocation and its valuation for its remaining chores would be higher than the valuation of $j_2^1, j_3^1, \cdots j_n^1$. Since j_1^1 is a chore (whose valuation is ≤ 0 for everyone), the valuation of i for its allocation after dropping, would be greater than that of $j_1^1, j_2^1, \cdots j_n^1$.



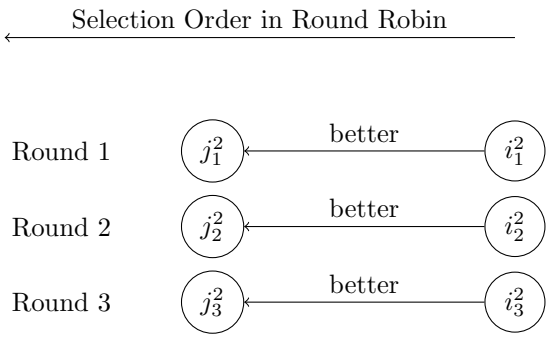
For Phase 2, say agent j picks before agent i in Phase 1, we will show that EF-1 property holds for both i and j.

- **Case 1: Agent i is Envy Free Upto one item for agent j's allocation**

Phase 1



Phase 2



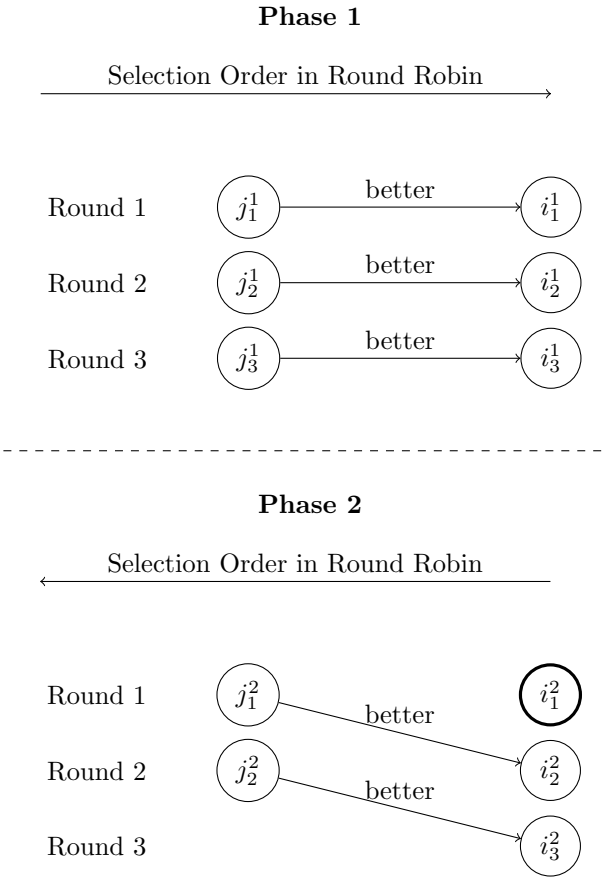
In Phase 1, Agent i's choice in round k is better than agent j's choice in round $k+1$ (in terms of valuation of agent i).

In Phase 2, agent i's choice in round k is better than agent j's choice in round k (in terms of the valuation of agent i).

Also, j_1^1, the object that agent j chose in Round 1 of Phase 1 is a negative item (valuation ≤ 0 for everyone). Thus, if agent i drops the chore that she received in the last round of Phase 1, then the valuation of her own bundle will be greater than the valuation of j's bundle. A small representative example is shown indicating the inequalities used in the argument and the item dropped by i shown in bold.

Note: The argument still remains the same if in the last round of Phase 2, agent j does not get to chose an item (due shortage of items). This is because the object chosen by i in the last round will have positive valuation for her, which will only increase her valuation.

- **Case 2: Agent j is Envy Free Upto one item for agent i's allocation**



In Phase 1, Agent j's choice in round k is better that agent i's choice in the same round, in terms of the valuation of agent j.

In Phase 2, Agent j's choice in round k is better than agent i's choice in round $k+1$, in terms of the valuations of j.

We can drop the first item allocated to i in Phase 2, so that the total valuation of agent j's own bundle is greater than agent i's bundle (in terms of valuation of agent j).

A small representative example is shown indicating the inequalities used in the argument and the item dropped by i shown in bold.

Note: The same argument holds even if agent j had been allocated an item in the last round of Phase 2. Since that item would have been a good for agent j, it would only have increased her valuation.

Thus, Doubly Round Robin Algorithm returns EF-1 allocation. ■

11.7 Doubly Monotone Valuations for Mixed Item Allocation

Consider allocation under the following restricted class of valuations called doubly monotone valuations.

Definition 11.4 (Doubly Monotone Valuations) *All the items can be partitioned into **goods** or **bads**. An agent's valuation function satisfies the following conditions:*

Good: *If g is a good for agent i, then:*

$$v_i(S \cup \{g\}) \geq v_i(S), \quad \forall S \subseteq M \setminus \{g\}$$
$$v_i(\bar{S} \cup \{g\}) > v_i(\bar{S}), \quad \exists \bar{S} \subseteq M \setminus \{g\}.$$

Bad: *If c is a chore for agent j, then:*

$$v_j(S \cup \{c\}) \leq v_j(S), \quad \forall S \subseteq M \setminus \{c\}.$$

Similar to the additive case, we define positive and negative items for the doubly monotone valuation case.

- ***Positive Items:*** An item is positive if it is a ***good*** for at least one agent.
- ***Negative Item:*** An item is a negative item if it is ***bad*** / ***chore*** for all agents.

11.8 Proposed Algorithm for Allocation under Monotone Valuations

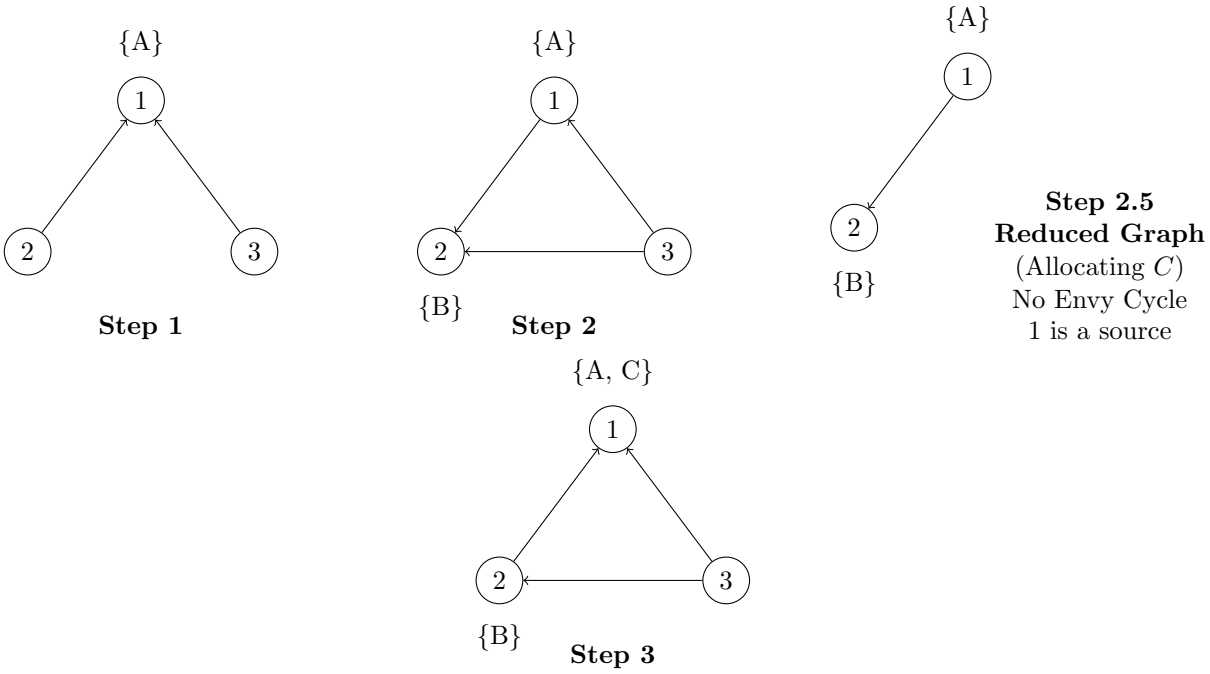
For doubly monotone items, the following two-phase allocation algorithm returns an EF1 (Envy-Free up to one item) allocation:

- **Phase 1:** Assign positive items via *envy-cycle elimination*, considering only the agents who perceive them as "good".
- **Phase 2:** Assign negative items via *top-trading envy-cycle elimination (TT ECE).*

11.8.1 Example Run of Phase 1

Let the valuation matrix of three positive items for three players be as follows. The allocation order is A, B, C and the tie break order is 1, 2, 3.

	A	B	C
1	1	10	10
2	2	5	10
3	3	5	-10



The above example shows the difference between a normal envy cycle elimination step and the modified step. We look for sources in the reduced graph corresponding to agents which have positive valuation for the item.

11.9 Summary of Allocation of Indivisible Items

11.9.1 Goods

- **Additive**: Round Robin
- **Monotone non-decreasing**: Envy cycle elimination
- Additional PO $\rightarrow$ Nash optimal

11.9.2 Bads

- **Additive**: Round Robin
- **Monotone non-increasing**: Max envy-cycle elimination
- Additional PO: Known for few special types of valuation, e.g., bivalued.

11.9.3 Mixed

- **Additive**: Doubly round robin
- **Doubly monotone**: Envy cycle + max envy cycle
- **General**: Open