COV886 Special Module in Algorithms: Computational Social Choice

Lecture 11

Fair Division of Indivisible Goods

March 26, 2022 | Rohit Vaish

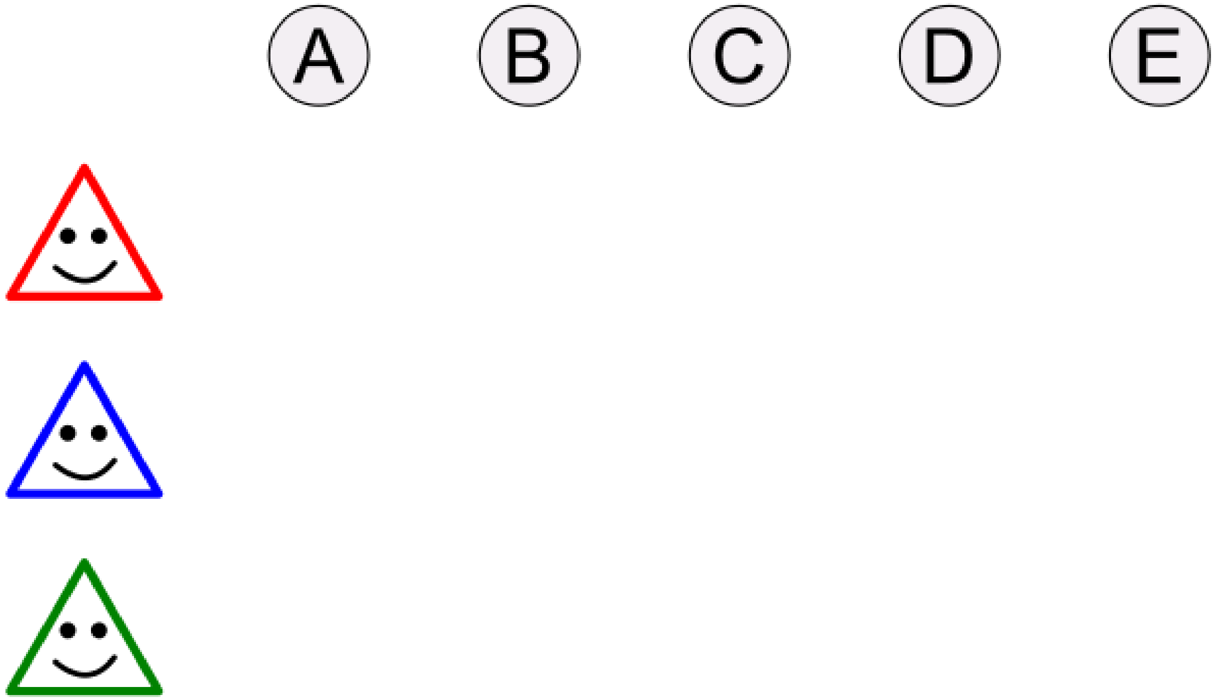
Reminder about starting recording

Vaccine COVID-19
Vaccine COVID-19
Vaccine COVID-19

The Model

The Model

The Model



The Model

	A	B	C	D	E
	4	1	2	2	2
	1	0	5	1	1
	1	1	5	1	1

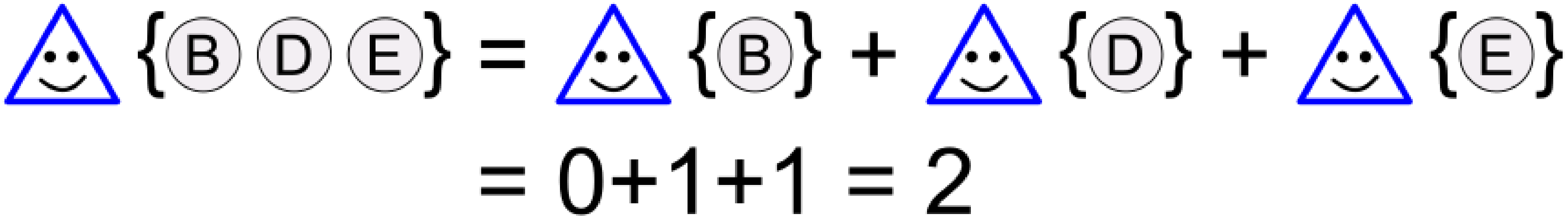
The Model

	A	B	C	D	E
	4	1	2	2	2
	1	0	5	1	1
	1	1	5	1	1

The Model

	A	B	C	D	E
	4	1	2	2	2
	1	0	5	1	1
	1	1	5	1	1

Additive valuations



Envy-Freeness [Gamow and Stern, 1958; Foley, 1967]

Envy-Freeness [Gamow and Stern, 1958; Foley, 1967]

Each agent prefers its own bundle over that of any other agent.

Envy-Freeness [Gamow and Stern, 1958; Foley, 1967]

Each agent prefers its own bundle over that of any other agent.

	A	B	C
	4	1	2
	1	1	5

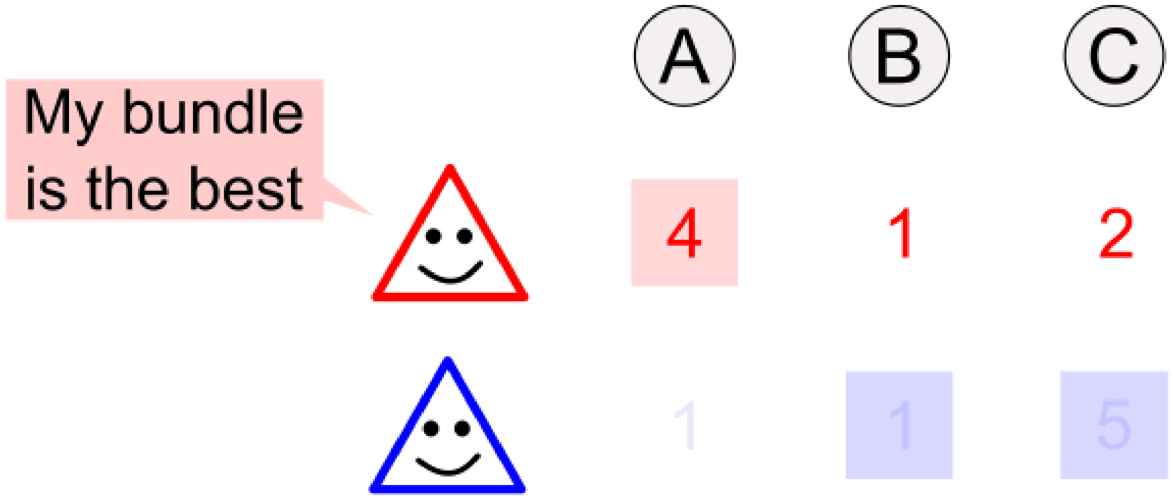
Envy-Freeness [Gamow and Stern, 1958; Foley, 1967]

Each agent prefers its own bundle over that of any other agent.

	A	B	C
	4	1	2
	1	1	5

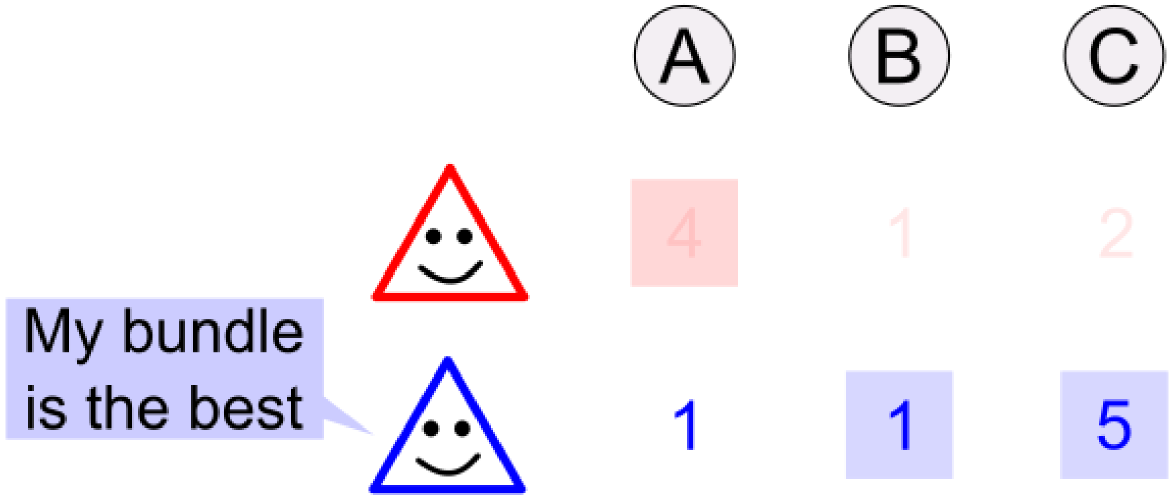
Envy-Freeness [Gamow and Stern, 1958; Foley, 1967]

Each agent prefers its own bundle over that of any other agent.



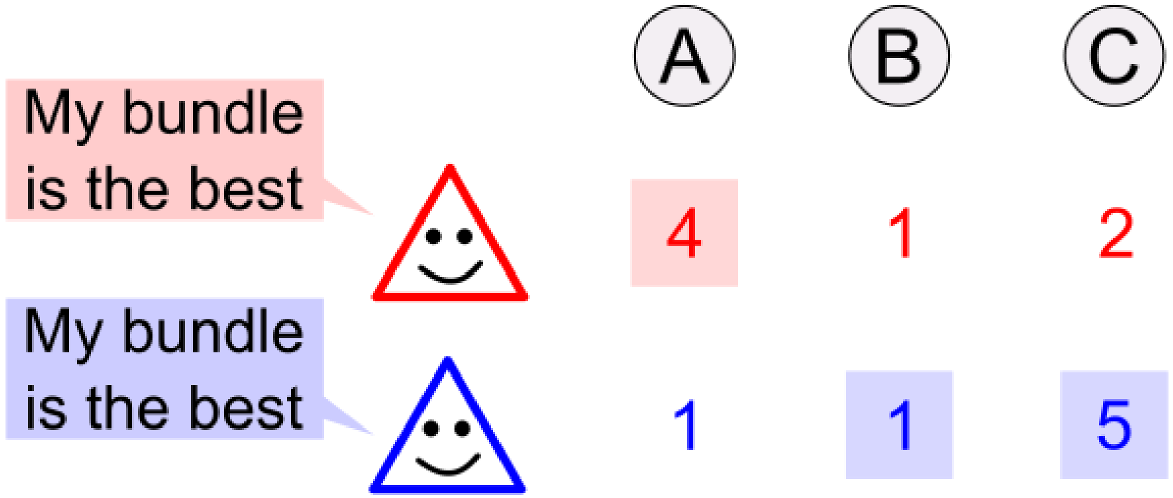
Envy-Freeness [Gamow and Stern, 1958; Foley, 1967]

Each agent prefers its own bundle over that of any other agent.



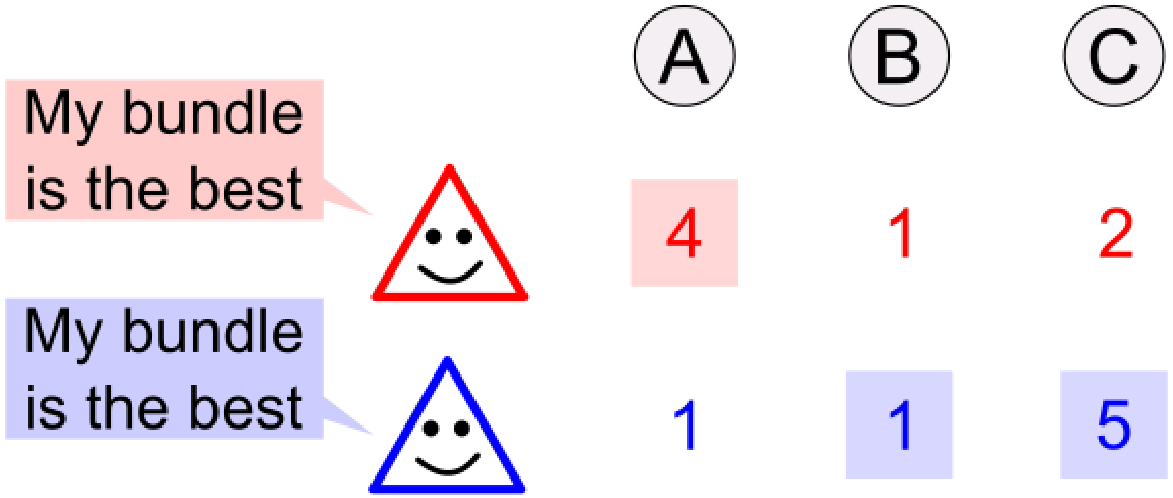
Envy-Freeness [Gamow and Stern, 1958; Foley, 1967]

Each agent prefers its own bundle over that of any other agent.



Envy-Freeness [Gamow and Stern, 1958; Foley, 1967]

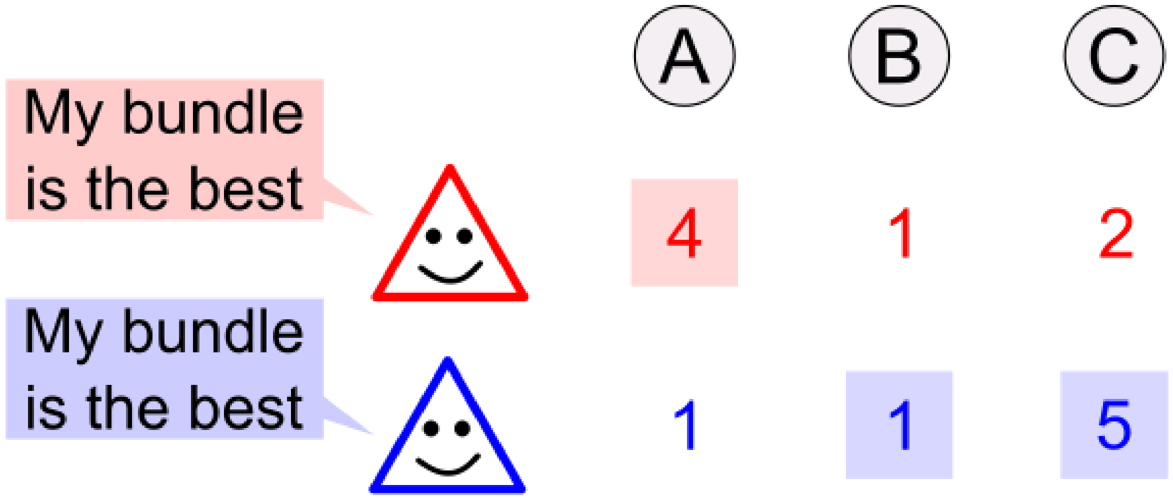
Each agent prefers its own bundle over that of any other agent.



Allocation $A = (A_1, A_2, ..., A_n)$ is EF if for every pair of agents i, k, we have $v_i(A_i) \geq v_i(A_k)$.

Envy-Freeness [Gamow and Stern, 1958; Foley, 1967]

Each agent prefers its own bundle over that of any other agent.

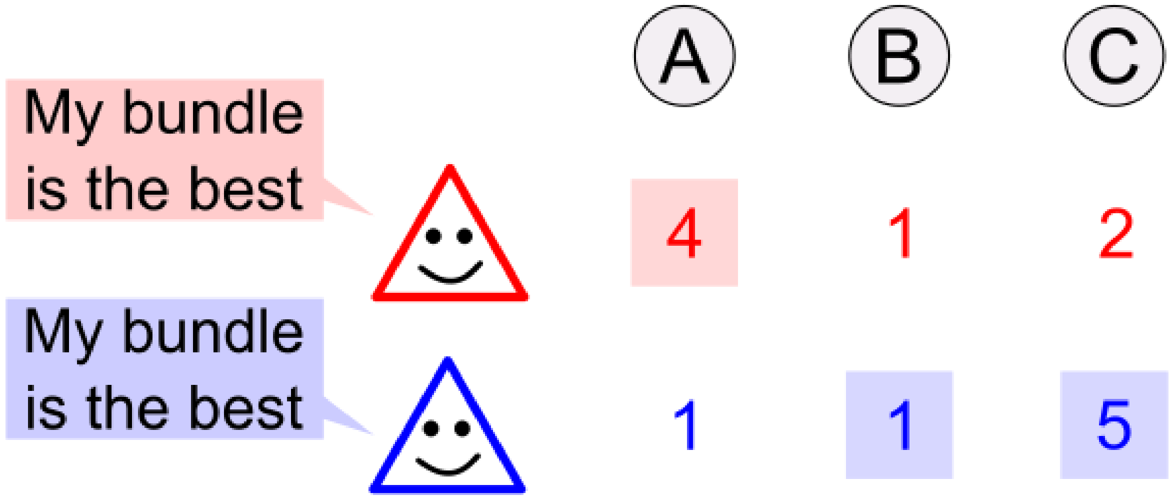


Allocation $A = (A_1, A_2, ..., A_n)$ is EF if for every pair of agents i, k, we have $v_i(A_i) \geq v_i(A_k)$.

Not guaranteed to exist (two agents, one good)

Envy-Freeness [Gamow and Stern, 1958; Foley, 1967]

Each agent prefers its own bundle over that of any other agent.



Allocation $A = (A_1, A_2, ..., A_n)$ is EF if for every pair of agents i, k, we have $v_i(A_i) \geq v_i(A_k)$.

- Not guaranteed to exist (two agents, one good)
- Checking whether an EF allocation exists is NP-complete

Envy-Freeness Up To One Good [Budish, 2011]

Envy-Freeness Up To One Good [Budish, 2011]

Envy can be eliminated by removing some good in the envied bundle.

Envy-Freeness Up To One Good [Budish, 2011]

Envy can be eliminated by removing some good in the envied bundle.

	A	B	C
	4	1	2
	1	1	5

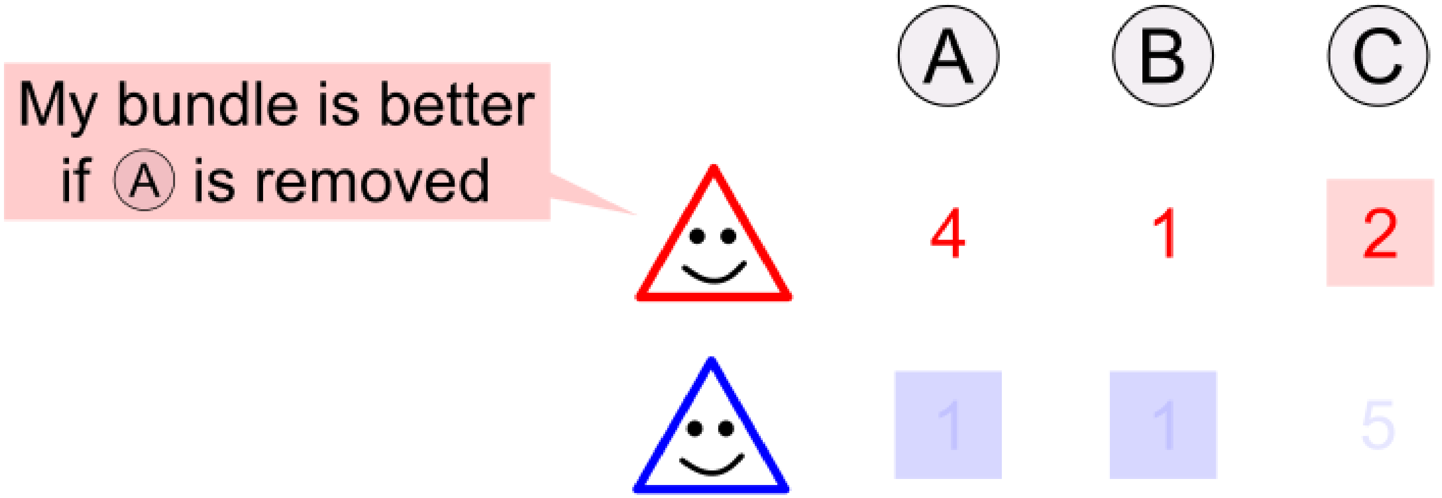
Envy-Freeness Up To One Good [Budish, 2011]

Envy can be eliminated by removing some good in the envied bundle.

	A	B	C
	4	1	2
	1	1	5

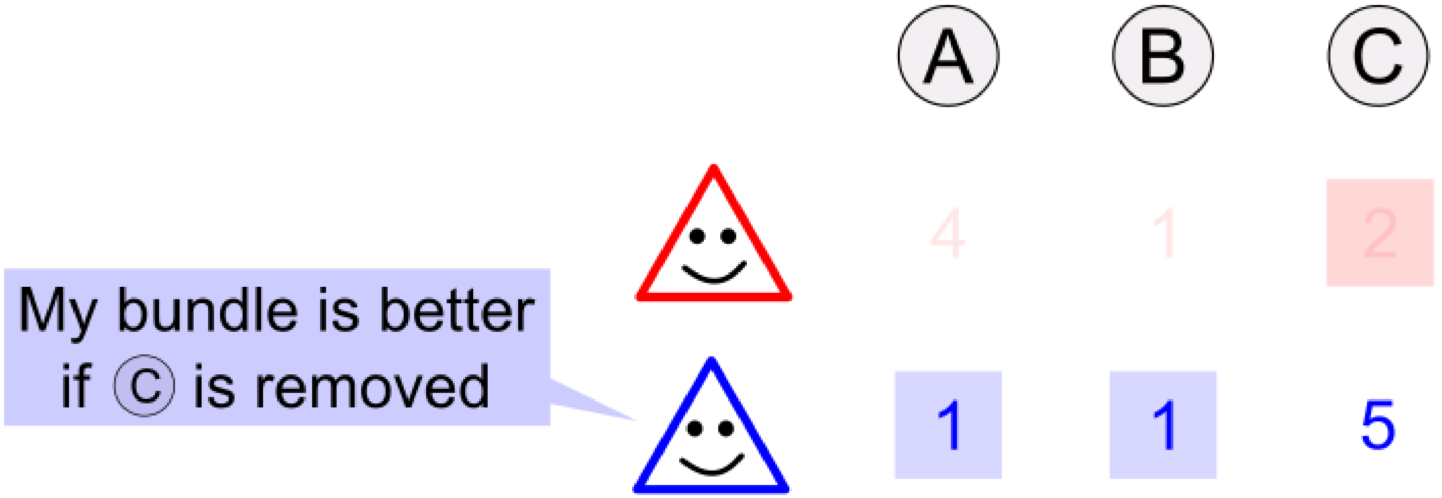
Envy-Freeness Up To One Good [Budish, 2011]

Envy can be eliminated by removing some good in the envied bundle.



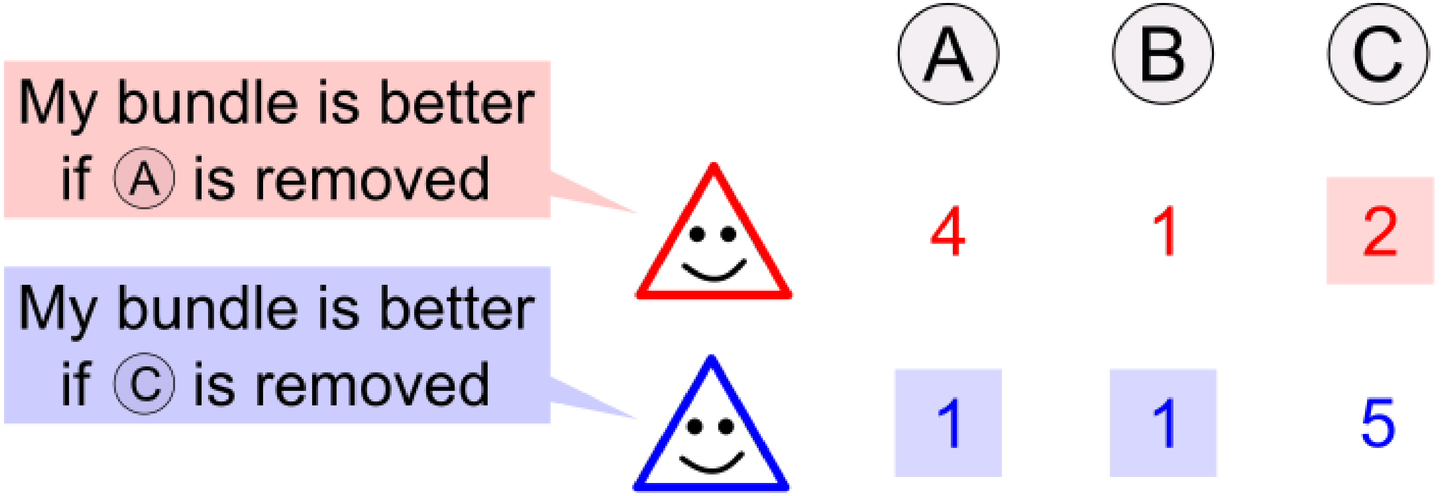
Envy-Freeness Up To One Good [Budish, 2011]

Envy can be eliminated by removing some good in the envied bundle.



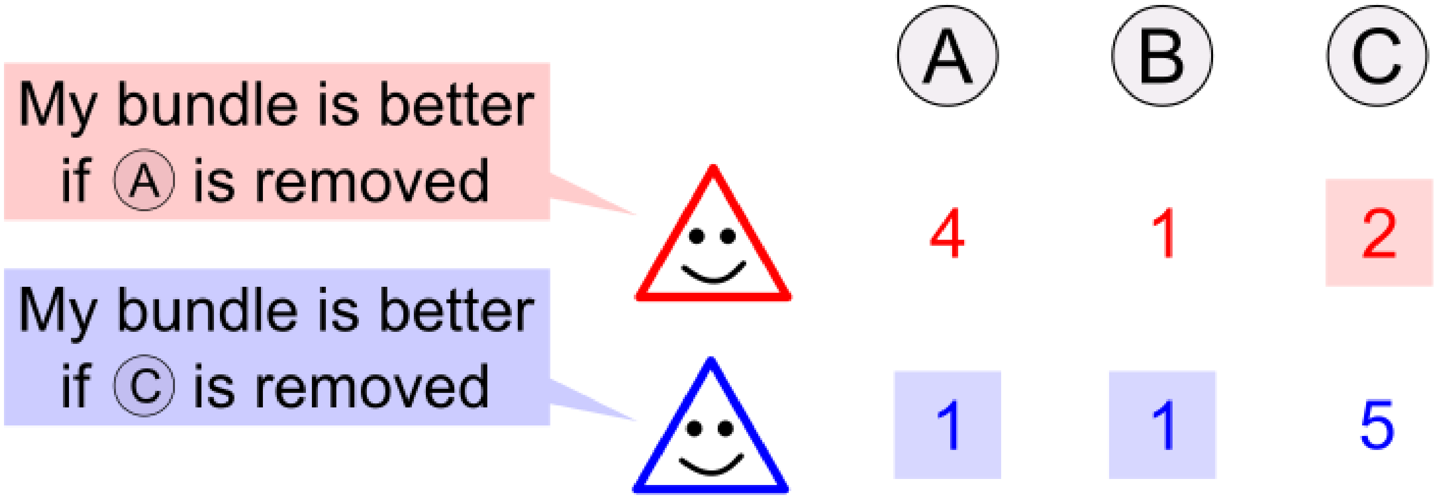
Envy-Freeness Up To One Good [Budish, 2011]

Envy can be eliminated by removing some good in the envied bundle.



Envy-Freeness Up To One Good [Budish, 2011]

Envy can be eliminated by removing some good in the envied bundle.



Allocation $A = (A_1, \ldots, A_n)$ is EF1 if for every pair of agents i, k, there exists a good $j \in A_k$ such that $v_i(A_i) \geq v_i(A_k \setminus \{j\})$.

Envy-Freeness Up To One Good

[Budish, 2011]

Envy can be eliminated by removing some good in the envied bundle.

My bundle is better if Ⓐ is removed

My bundle is better if Ⓒ is removed

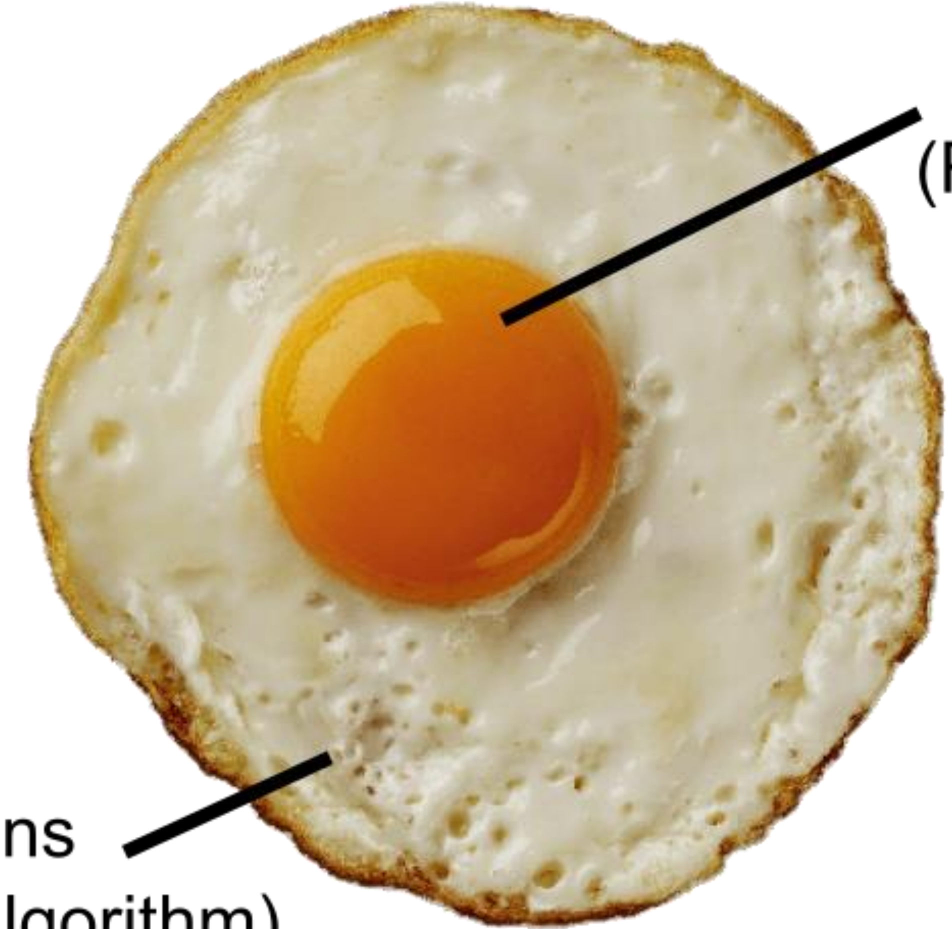
	A	B	C
Red agent	4	1	2
Blue agent	1	1	5

Allocation $A = (A_1, \ldots, A_n)$ is EF1 if for every pair of agents i, k, there exists a good $j \in A_k$ such that $v_i(A_i) \geq v_i(A_k \setminus \{j\})$.

Guaranteed to exist and efficiently computable

Coming Up

Algorithms for finding an EF1 allocation



Additive valuations
(Round-robin algorithm)

Monotone valuations
(Envy-cycle elimination algorithm)

Round-robin algorithm

Round-robin algorithm

- Fix an ordering of the agents, say a_1, a_2, a_3, ..., a_n.

Round-robin algorithm

- Fix an ordering of the agents, say a_1, a_2, a_3, ..., a_n.
- Agents take turns according to the ordering (a_1, a_2, ..., a_n, a_1, a_2, ..., a_n,...) to pick their favorite item from the set of remaining items.

Round-robin algorithm

- Fix an ordering of the agents, say a_1, a_2, a_3, ..., a_n.
- Agents take turns according to the ordering (a_1, a_2, ..., a_n, a_1, a_2, ..., a_n,...) to pick their favorite item from the set of remaining items.

	A	B	C	D	E
	4	1	2	2	2
	1	0	5	1	1
	1	1	5	1	1

Round-robin algorithm

- Fix an ordering of the agents, say $a_1, a_2, a_3, \ldots, a_n$.
- Agents take turns according to the ordering $(a_1, a_2, \ldots, a_n, a_1, a_2, \ldots, a_n, \ldots)$ to pick their favorite item from the set of remaining items.

	A	B	C	D	E
	4	1	2	2	2
	1	0	5	1	1
	1	1	5	1	1

Round-robin algorithm

- Fix an ordering of the agents, say a_1, a_2, a_3, ..., a_n.
- Agents take turns according to the ordering (a_1, a_2, ..., a_n, a_1, a_2, ..., a_n,...) to pick their favorite item from the set of remaining items.

	A	B	C	D	E
	4	1	2	2	2
	1	0	5	1	1
	1	1	5	1	1

Round-robin algorithm

- Fix an ordering of the agents, say $a_1, a_2, a_3, \ldots, a_n$.
- Agents take turns according to the ordering ($a_1, a_2, \ldots, a_n, a_1, a_2, \ldots, a_n, \ldots$) to pick their favorite item from the set of remaining items.

	A	B	C	D	E
	4	1	2	2	2
	1	0	5	1	1
	1	1	5	1	1

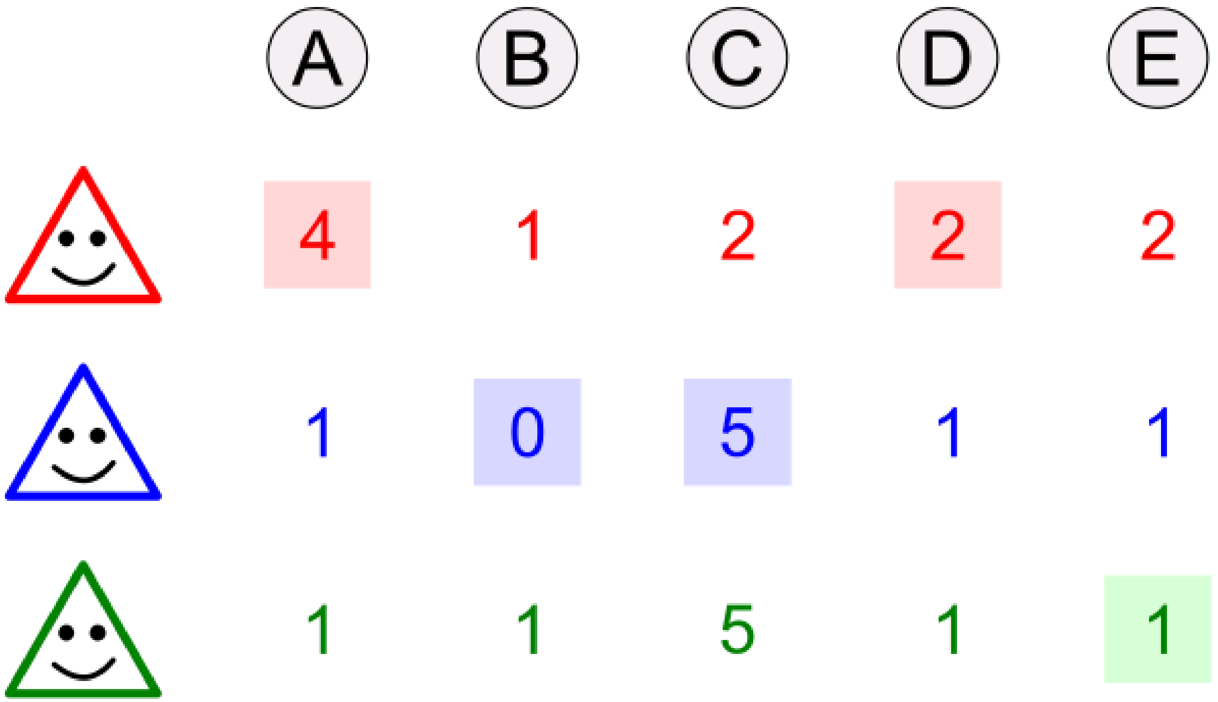
Round-robin algorithm

- Fix an ordering of the agents, say $a_1, a_2, a_3, \ldots, a_n$.
- Agents take turns according to the ordering ($a_1, a_2, \ldots, a_n, a_1, a_2, \ldots, a_n, \ldots$) to pick their favorite item from the set of remaining items.

	A	B	C	D	E
	4	1	2	**2**	2
	1	0	**5**	1	1
	1	1	5	1	**1**

Round-robin algorithm

- Fix an ordering of the agents, say a_1, a_2, a_3, ..., a_n.
- Agents take turns according to the ordering (a_1, a_2, ..., a_n, a_1, a_2, ..., a_n,...) to pick their favorite item from the set of remaining items.



For additive valuations, the allocation computed by round-robin algorithm satisfies EF1.

For additive valuations, the allocation computed by round-robin algorithm satisfies EF1.

a_1 a_2 a_3 . . . a_n

For additive valuations, the allocation computed by round-robin algorithm satisfies EF1.

a_1 a_2 a_3 . . . a_n

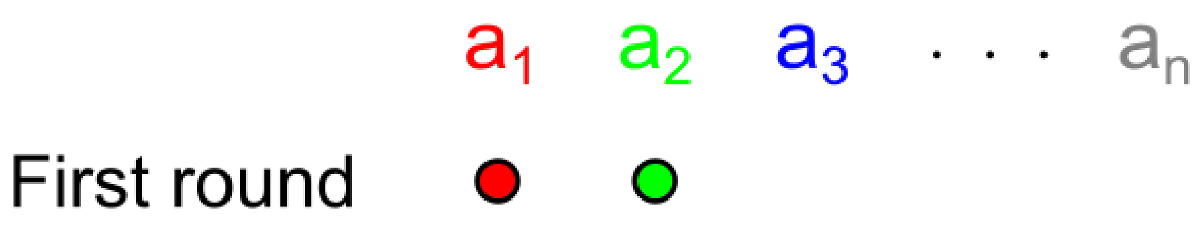
First round

For additive valuations, the allocation computed by round-robin algorithm satisfies EF1.

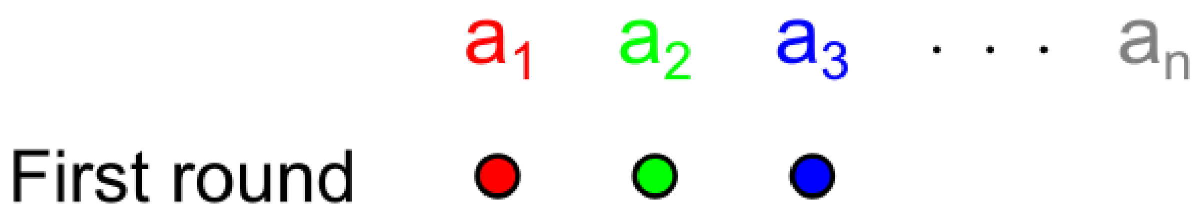
a_1 a_2 a_3 . . . a_n

First round ●

For additive valuations, the allocation computed by round-robin algorithm satisfies EF1.



For additive valuations, the allocation computed by round-robin algorithm satisfies EF1.

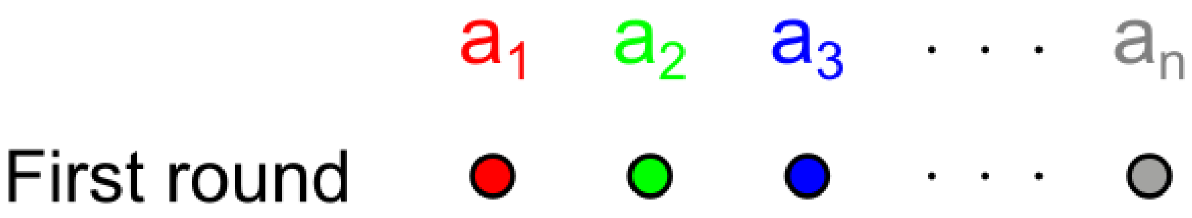


For additive valuations, the allocation computed by round-robin algorithm satisfies EF1.

a_1 a_2 a_3 . . . a_n

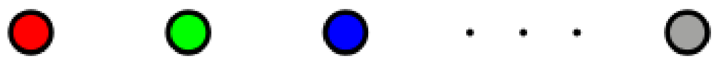
First round ● ● ● . . .

For additive valuations, the allocation computed by round-robin algorithm satisfies EF1.



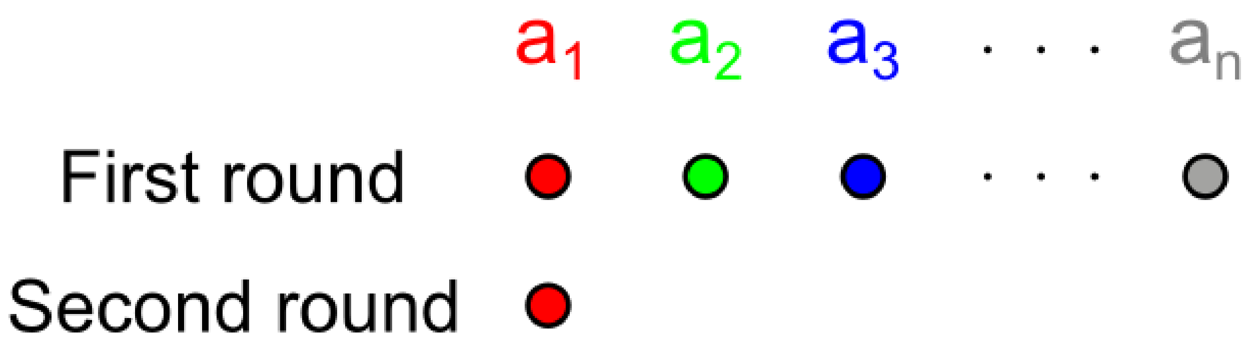
For additive valuations, the allocation computed by round-robin algorithm satisfies EF1.

a_1 a_2 a_3 . . . a_n

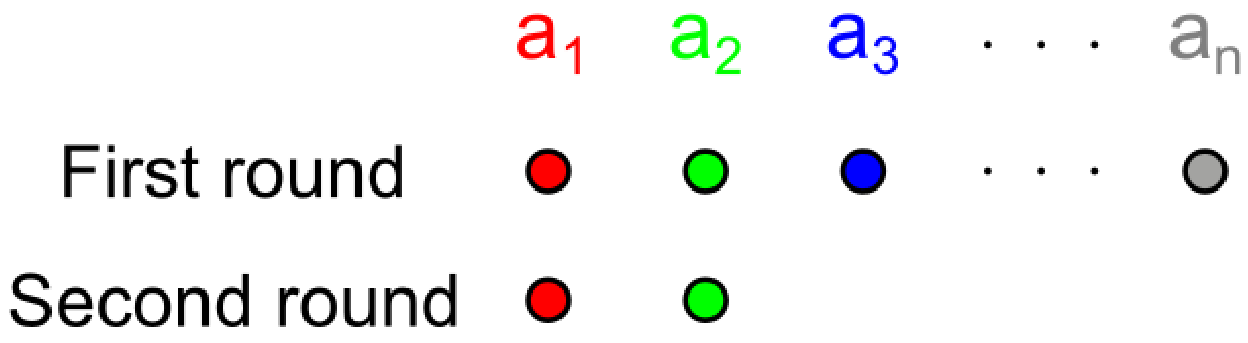
First round 

Second round

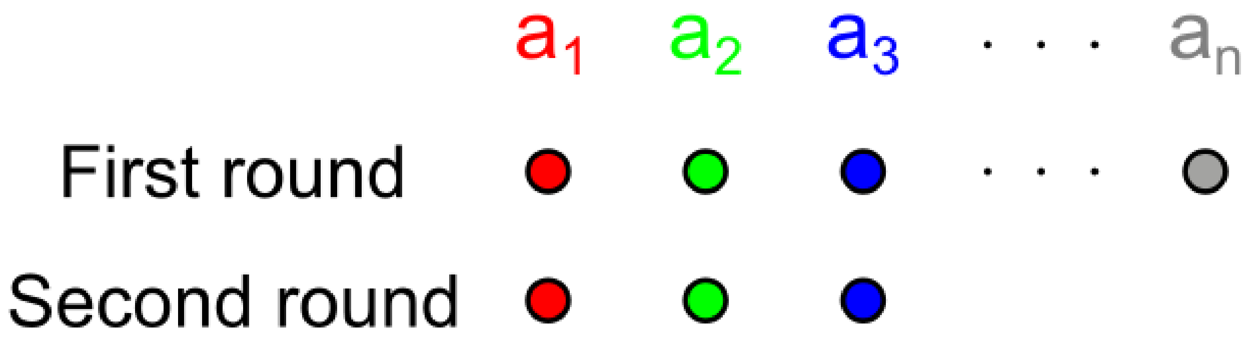
For additive valuations, the allocation computed by round-robin algorithm satisfies EF1.



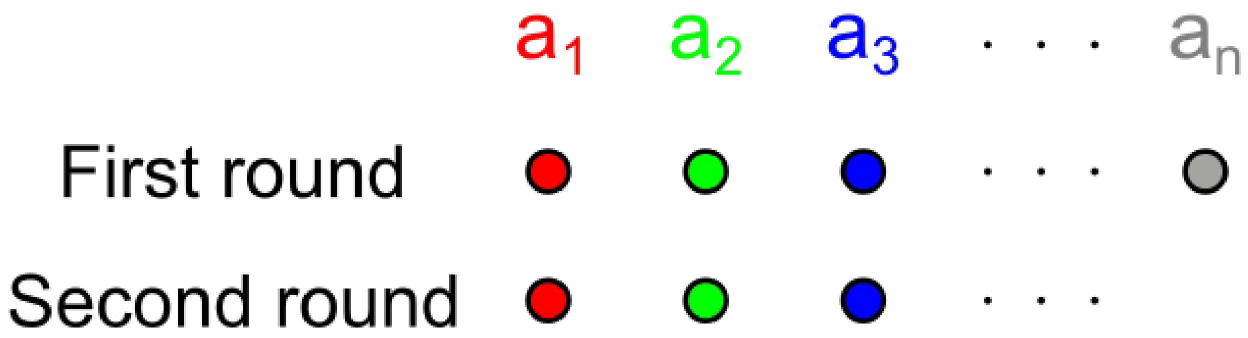
For additive valuations, the allocation computed by round-robin algorithm satisfies EF1.



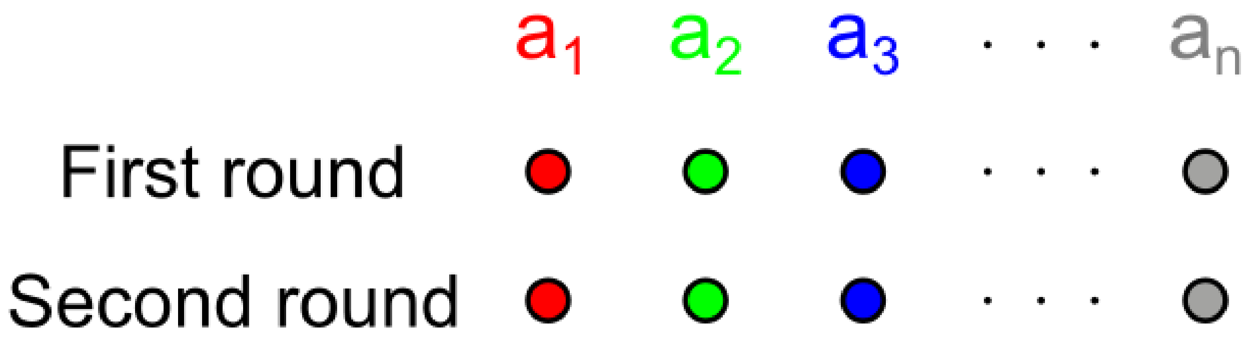
For additive valuations, the allocation computed by round-robin algorithm satisfies EF1.



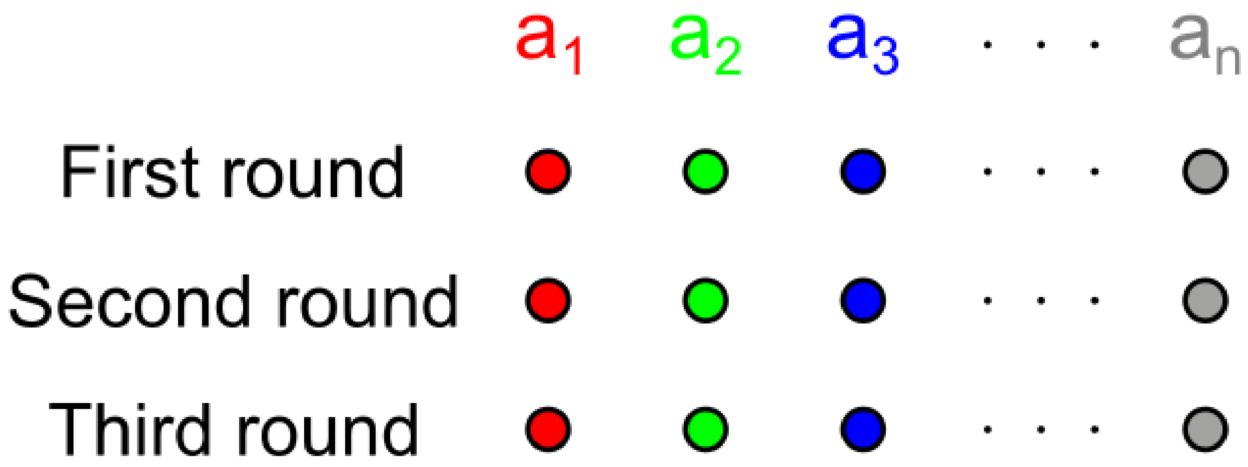
For additive valuations, the allocation computed by round-robin algorithm satisfies EF1.



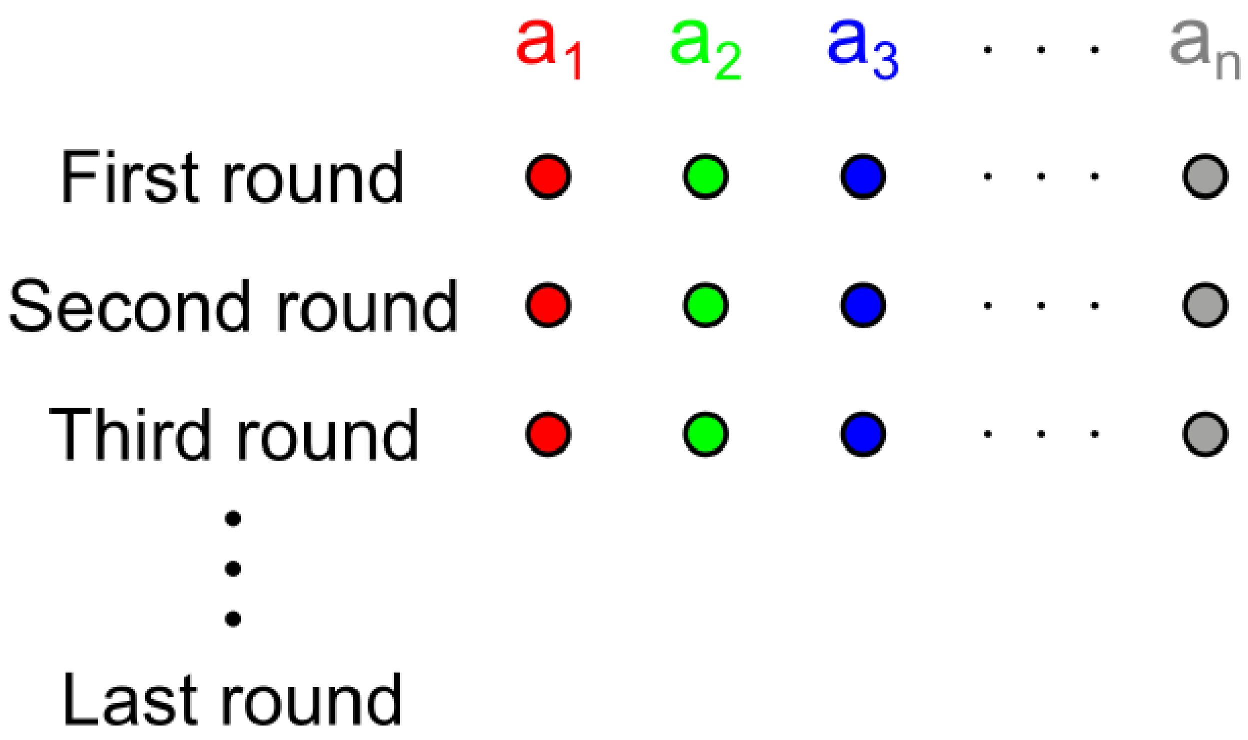
For additive valuations, the allocation computed by round-robin algorithm satisfies EF1.



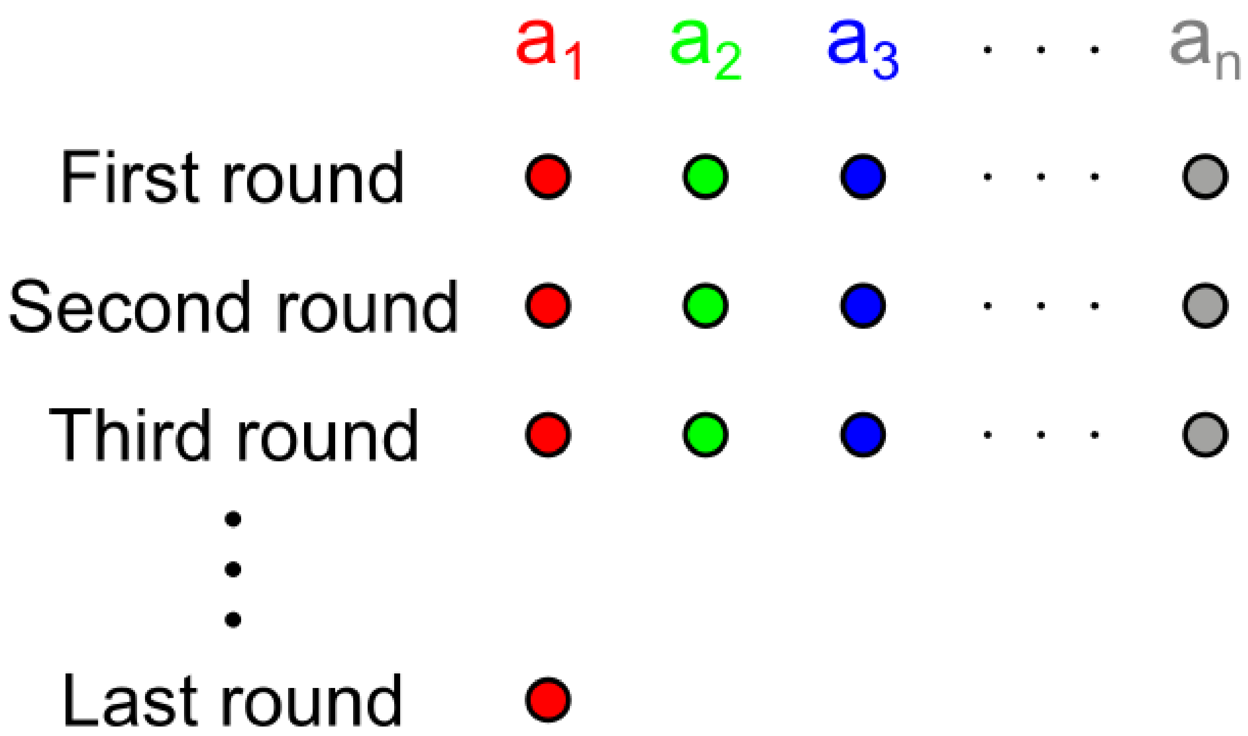
For additive valuations, the allocation computed by round-robin algorithm satisfies EF1.



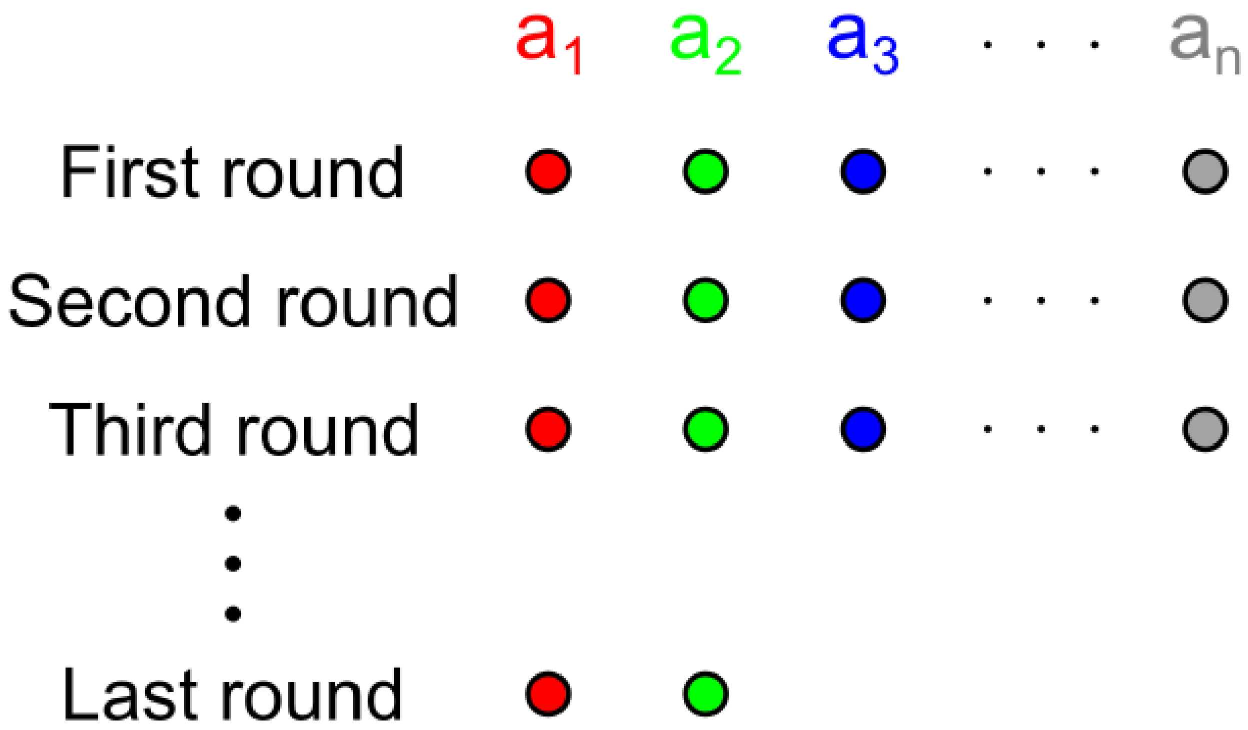
For additive valuations, the allocation computed by round-robin algorithm satisfies EF1.



For additive valuations, the allocation computed by round-robin algorithm satisfies EF1.



For additive valuations, the allocation computed by round-robin algorithm satisfies EF1.



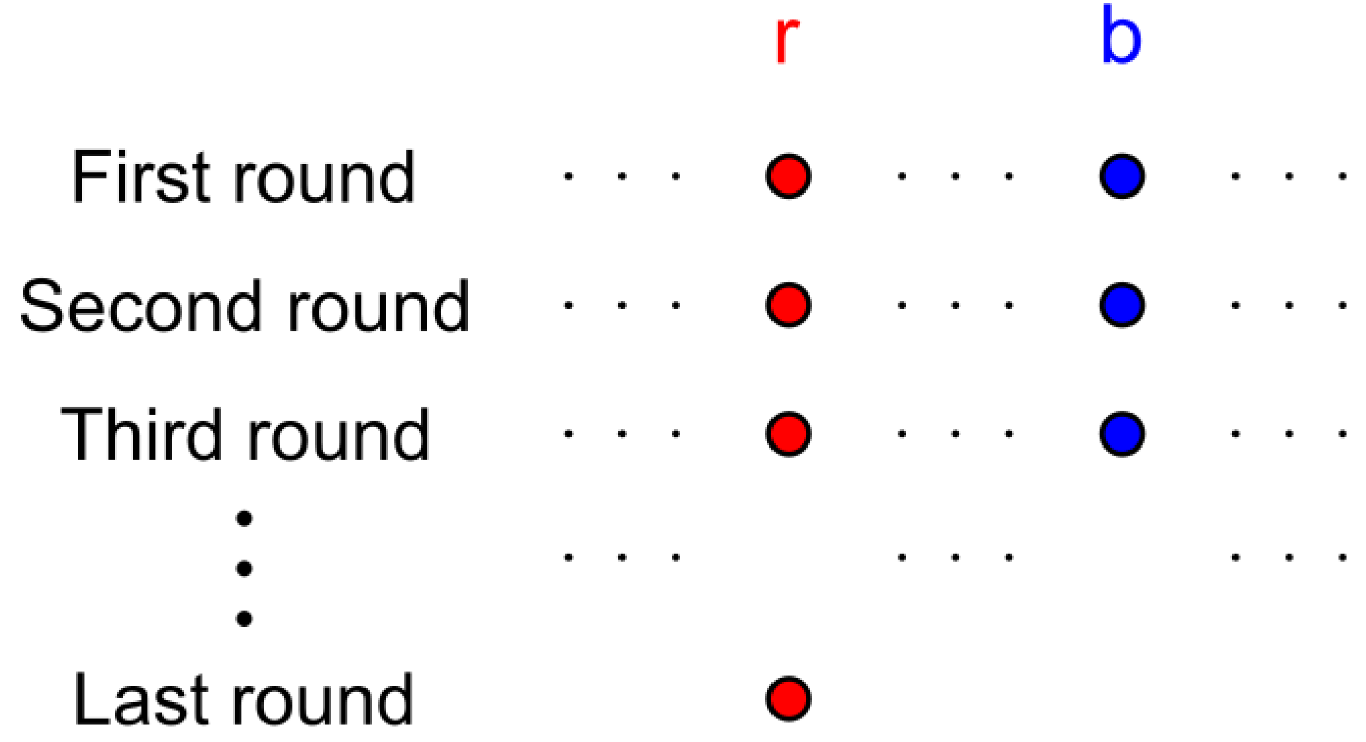
For additive valuations, the allocation computed by round-robin algorithm satisfies EF1.

For additive valuations, the allocation computed by round-robin algorithm satisfies EF1.

Fix a pair of agents (r,b). Analyze envy of r towards b.

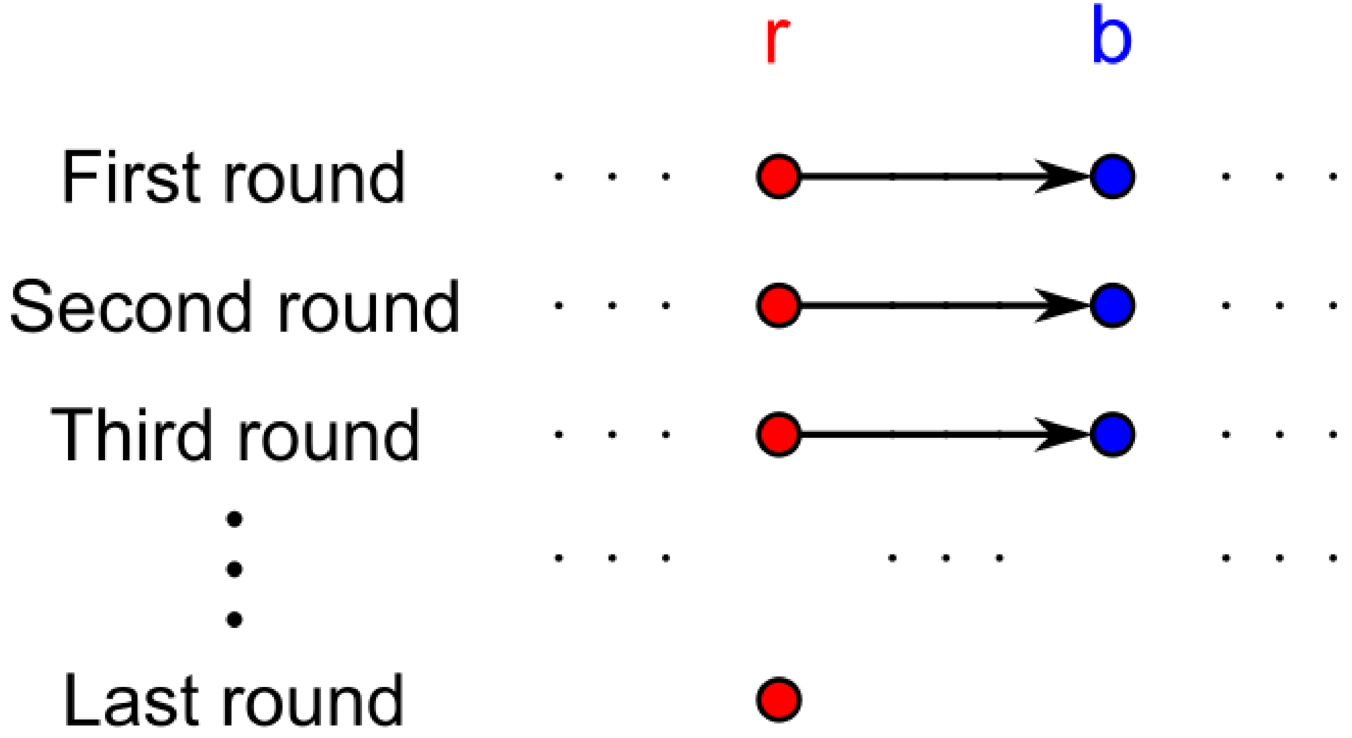
For additive valuations, the allocation computed by round-robin algorithm satisfies EF1.

Fix a pair of agents (r,b). Analyze envy of r towards b.



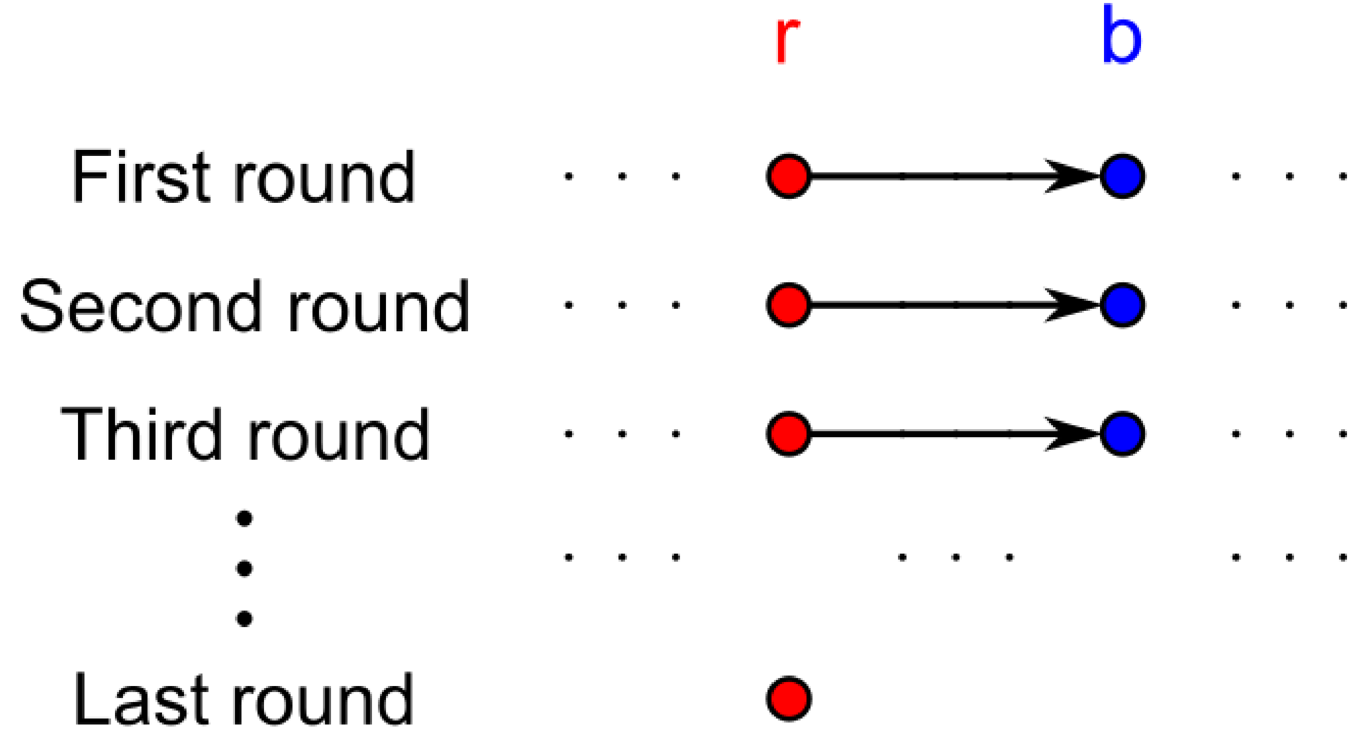
For additive valuations, the allocation computed by round-robin algorithm satisfies EF1.

Fix a pair of agents (r,b). Analyze envy of r towards b.



For additive valuations, the allocation computed by round-robin algorithm satisfies EF1.

Fix a pair of agents (r,b). Analyze envy of r towards b.



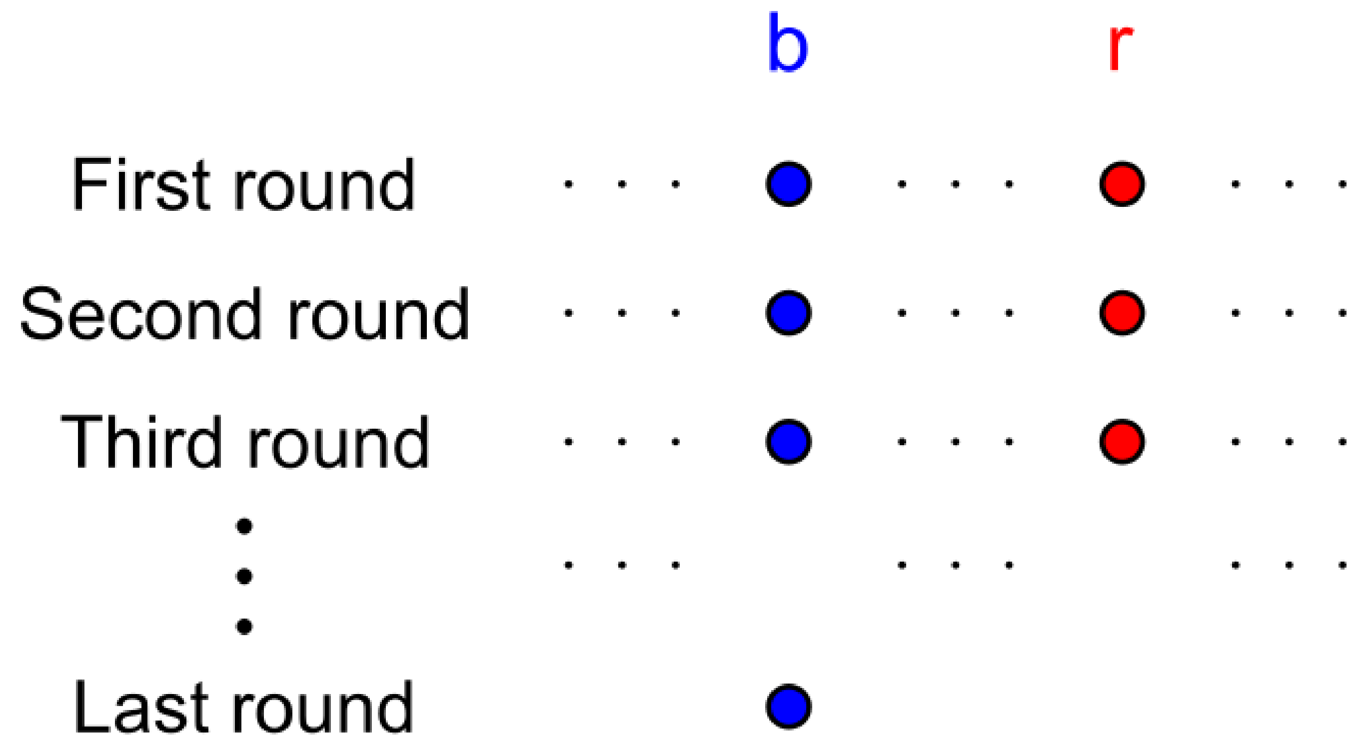
If r precedes b: Then, by additivity, $v_r(A_r) \geq v_r(A_b)$.

For additive valuations, the allocation computed by round-robin algorithm satisfies EF1.

Fix a pair of agents (r,b). Analyze envy of r towards b.

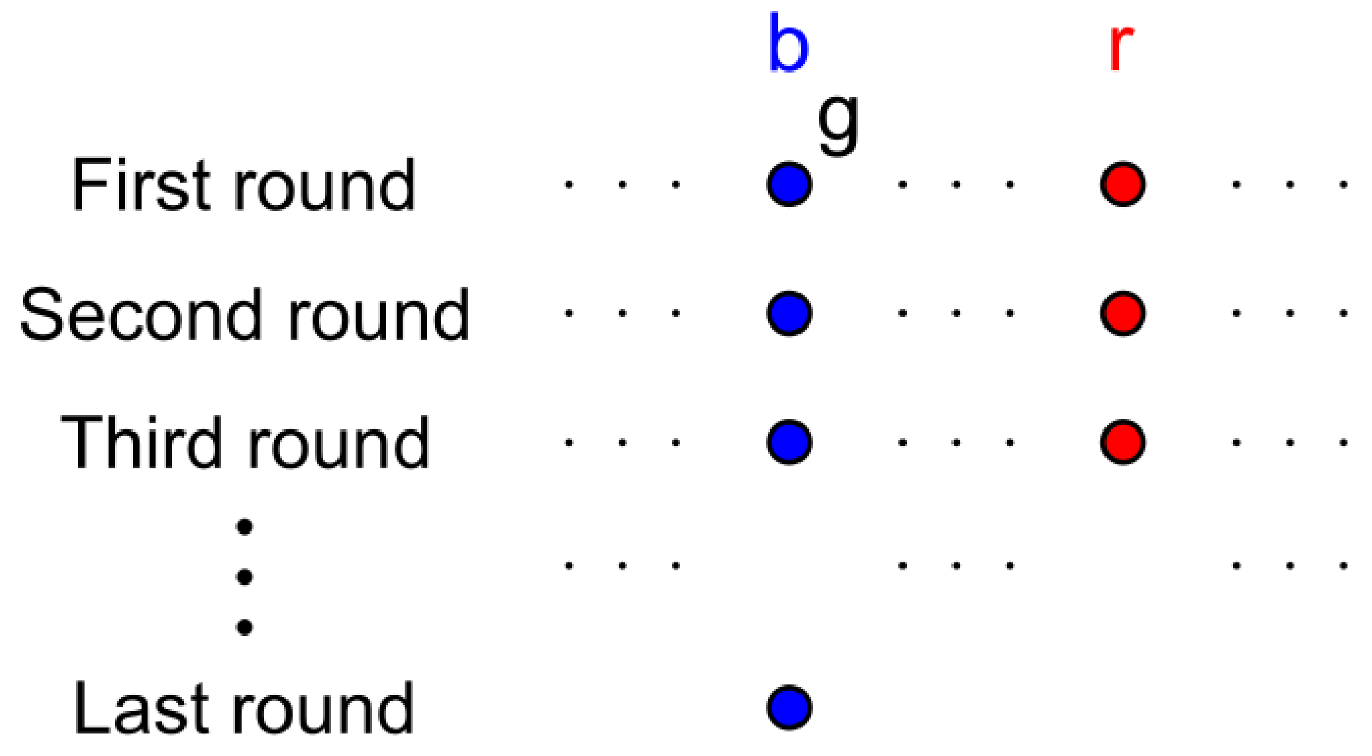
For additive valuations, the allocation computed by round-robin algorithm satisfies EF1.

Fix a pair of agents (r,b). Analyze envy of r towards b.



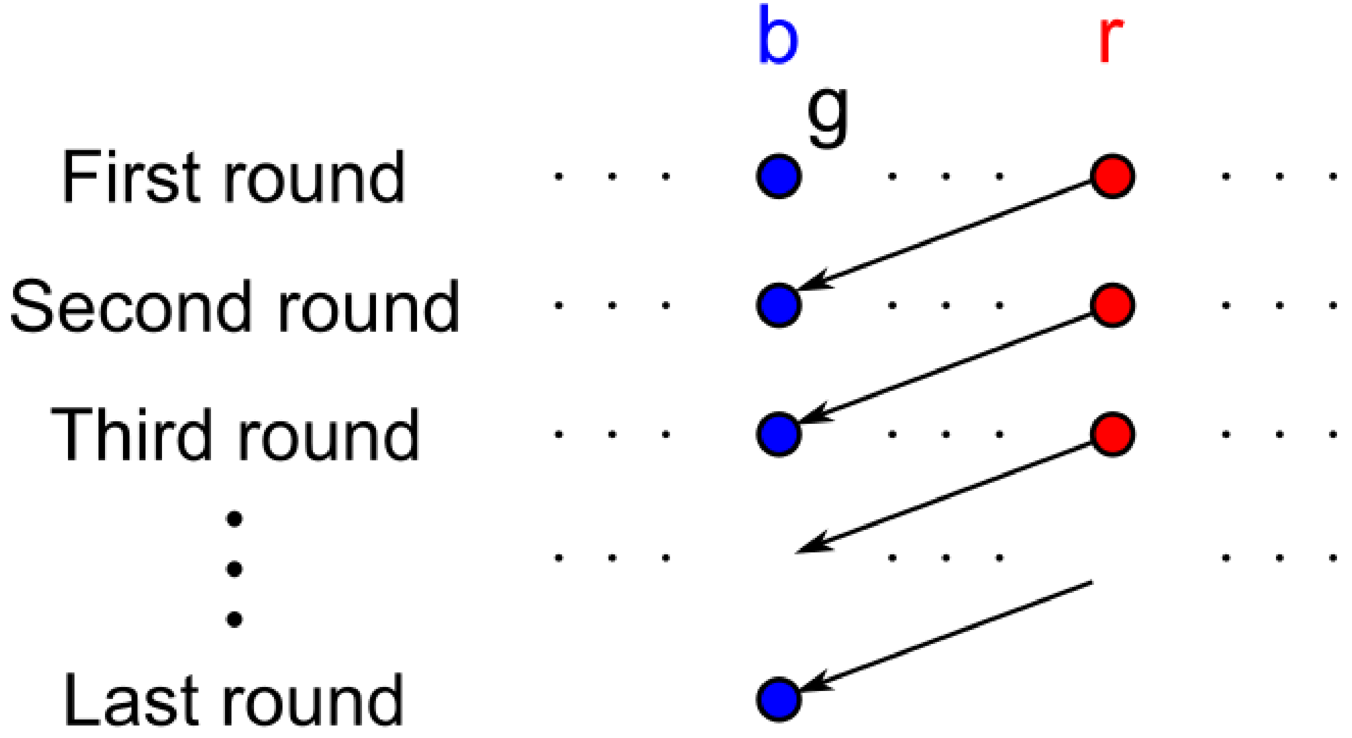
For additive valuations, the allocation computed by round-robin algorithm satisfies EF1.

Fix a pair of agents (r,b). Analyze envy of r towards b.



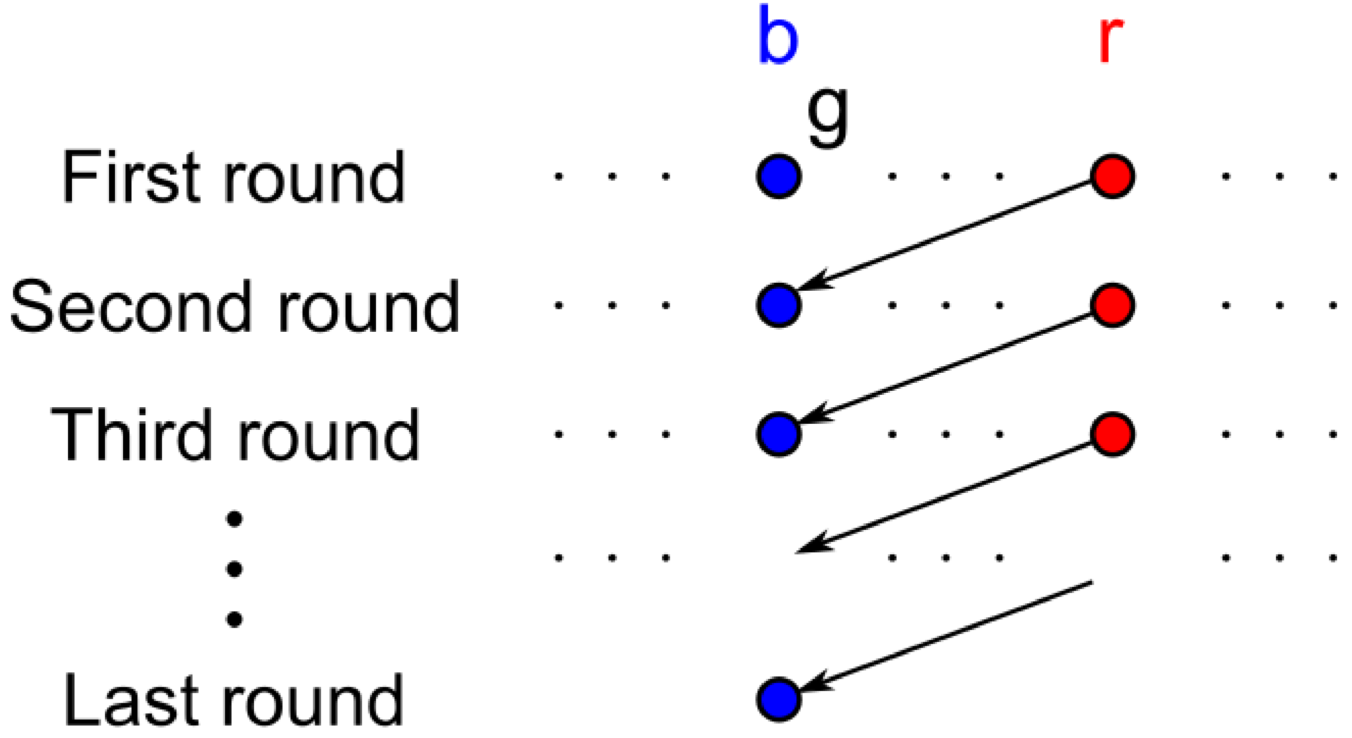
For additive valuations, the allocation computed by round-robin algorithm satisfies EF1.

Fix a pair of agents (r,b). Analyze envy of r towards b.



For additive valuations, the allocation computed by round-robin algorithm satisfies EF1.

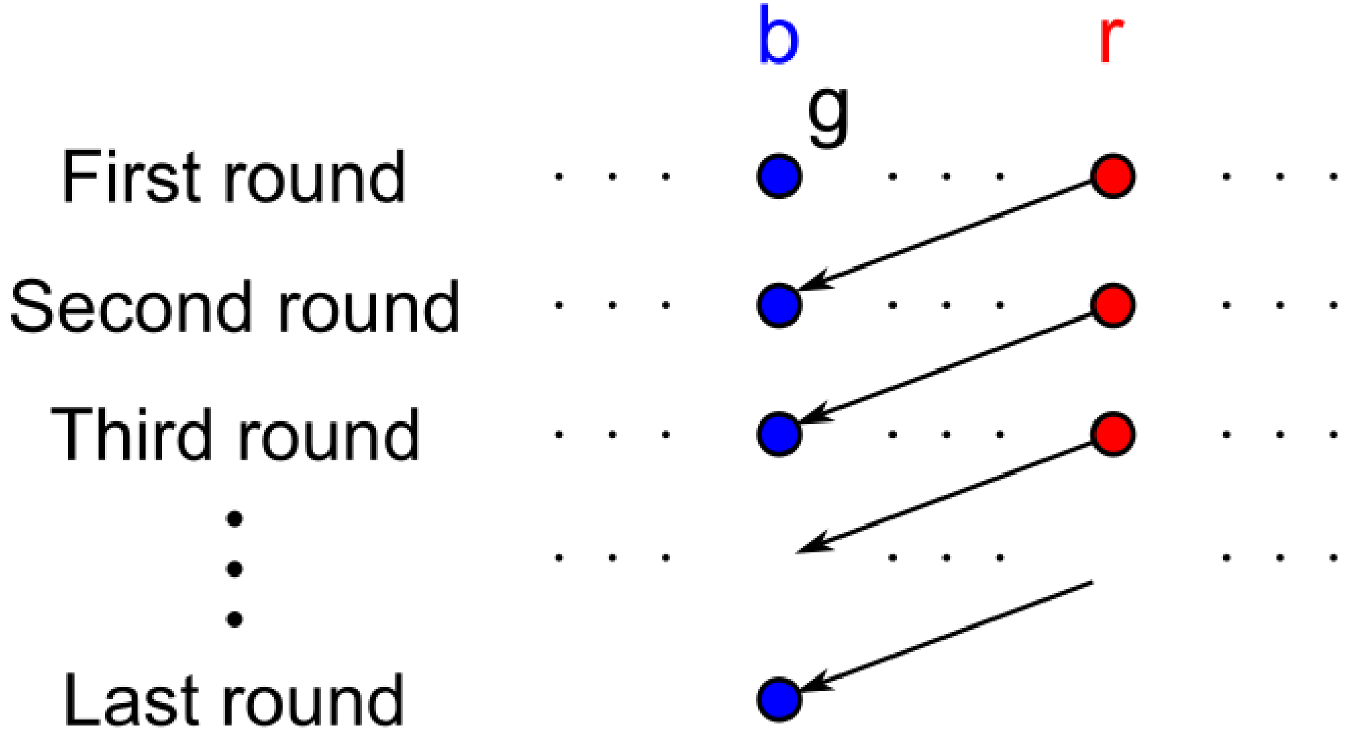
Fix a pair of agents (r,b). Analyze envy of r towards b.



If b precedes r: Again, by additivity, $v_r(A_r) \geq v_r(A_b \setminus \{g\})$.

For additive valuations, the allocation computed by round-robin algorithm satisfies EF1.

Fix a pair of agents (r,b). Analyze envy of r towards b.



If b precedes r: Again, by additivity, $v_r(A_r) \geq v_r(A_b \setminus \{g\})$.

■

Envy graph of an allocation

Envy graph of an allocation

- Vertices = agents
- Edge from vertex i to vertex k if agent i envies agent k in the given allocation.

Envy graph of an allocation

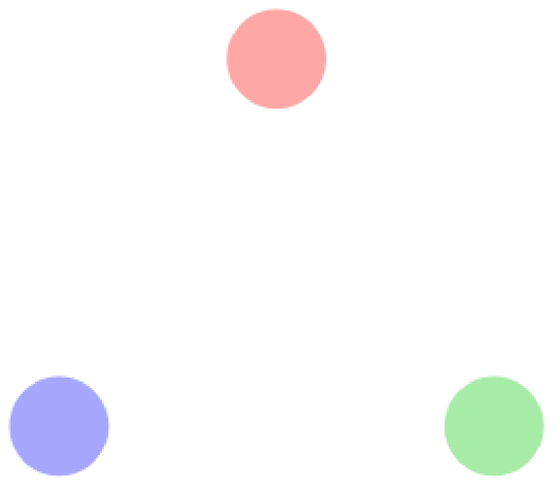
- Vertices = agents
- Edge from vertex i to vertex k if agent i envies agent k in the given allocation.

	A	B	C	D	E
	4	1	2	4	2
	1	0	5	1	1
	1	1	5	1	1

Envy graph of an allocation

- Vertices = agents
- Edge from vertex i to vertex k if agent i envies agent k in the given allocation.

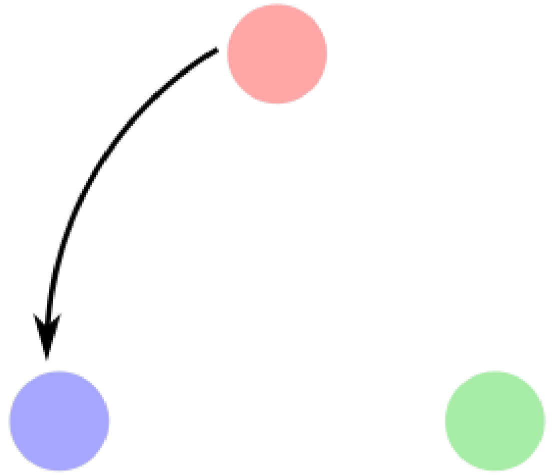
	A	B	C	D	E
	4	1	2	4	2
	1	0	5	1	1
	1	1	5	1	1



Envy graph of an allocation

- Vertices = agents
- Edge from vertex i to vertex k if agent i envies agent k in the given allocation.

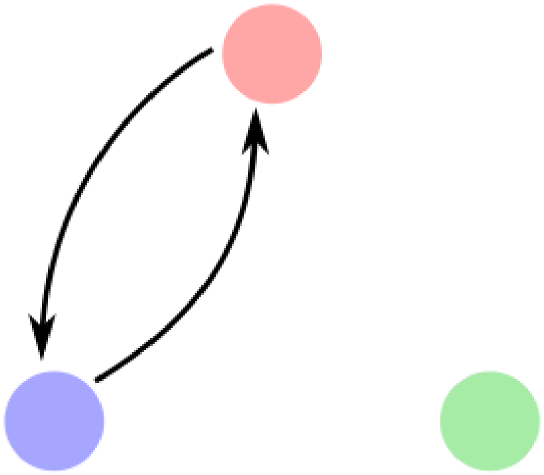
	A	B	C	D	E
	4	1	2	4	2
	1	0	5	1	1
	1	1	5	1	1



Envy graph of an allocation

- Vertices = agents
- Edge from vertex i to vertex k if agent i envies agent k in the given allocation.

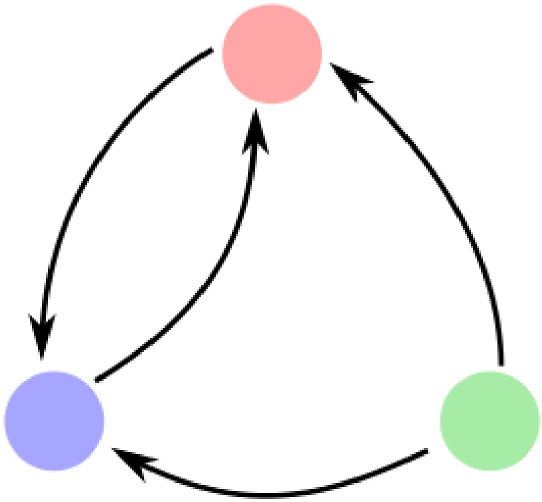
	A	B	C	D	E
	4	1	2	4	2
	1	0	5	1	1
	1	1	5	1	1



Envy graph of an allocation

- Vertices = agents
- Edge from vertex i to vertex k if agent i envies agent k in the given allocation.

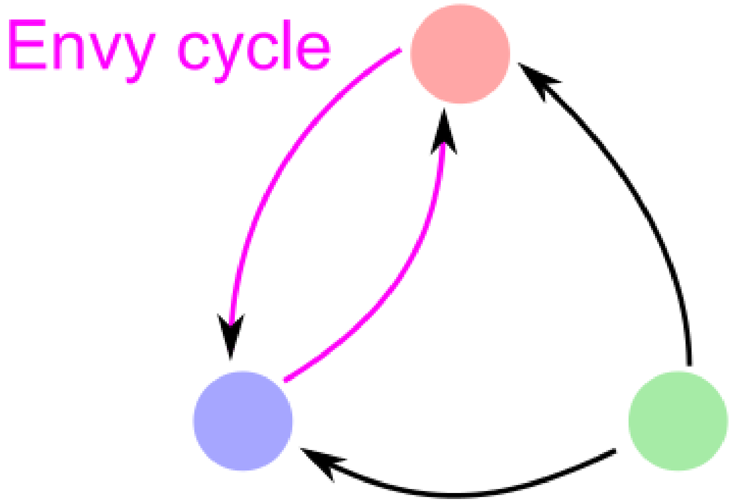
	A	B	C	D	E
	4	1	2	4	2
	1	0	5	1	1
	1	1	5	1	1



Envy graph of an allocation

- Vertices = agents
- Edge from vertex i to vertex k if agent i envies agent k in the given allocation.

	A	B	C	D	E
	4	1	2	4	2
	1	0	5	1	1
	1	1	5	1	1



Envy-cycle elimination algorithm

[Lipton, Markakis, Mossel, and Saberi, *EC* 2004]

Envy-cycle elimination algorithm

[Lipton, Markakis, Mossel, and Saberi, *EC* 2004]

While there is an unallocated good

Envy-cycle elimination algorithm

[Lipton, Markakis, Mossel, and Saberi, *EC* 2004]

While there is an unallocated good

- If the envy graph has a source vertex, assign the good to that agent.

Envy-cycle elimination algorithm

[Lipton, Markakis, Mossel, and Saberi, *EC* 2004]

While there is an unallocated good

- If the envy graph has a source vertex, assign the good to that agent.
- Otherwise, resolve envy cycles until a source vertex shows up, and then assign the good to it.

Envy-cycle elimination algorithm

[Lipton, Markakis, Mossel, and Saberi, *EC* 2004]

While there is an unallocated good

- If the envy graph has a source vertex, assign the good to that agent.
- Otherwise, resolve envy cycles until a source vertex shows up, and then assign the good to it.

each agent in the cycle gets the bundle that it is pointing to

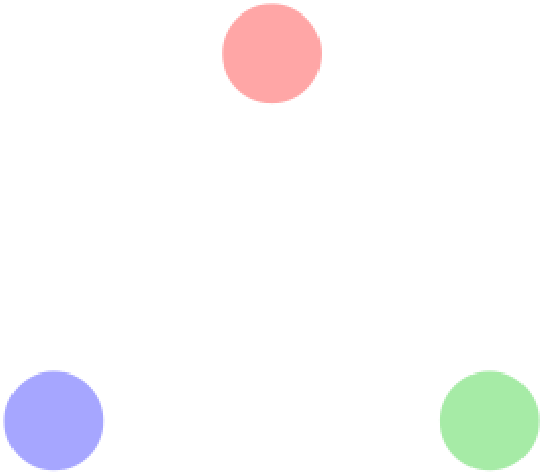
Envy-cycle elimination algorithm

[Lipton, Markakis, Mossel, and Saberi, *EC* 2004]

While there is an unallocated good

- If the envy graph has a source vertex, assign the good to that agent.
- Otherwise, resolve envy cycles until a source vertex shows up, and then assign the good to it.

	A	B	C	D
	0	2	0	1
	1	2	5	10
	1	4	2	10



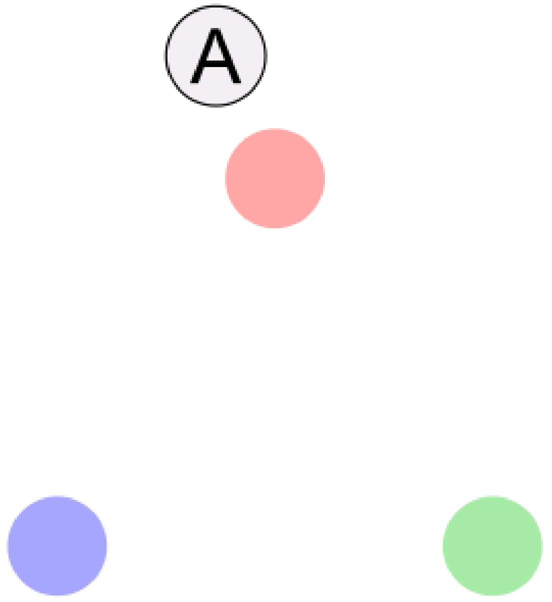
Envy-cycle elimination algorithm

[Lipton, Markakis, Mossel, and Saberi, *EC* 2004]

While there is an unallocated good

- If the envy graph has a source vertex, assign the good to that agent.
- Otherwise, resolve envy cycles until a source vertex shows up, and then assign the good to it.

	A	B	C	D
	0	2	0	1
	1	2	5	10
	1	4	2	10



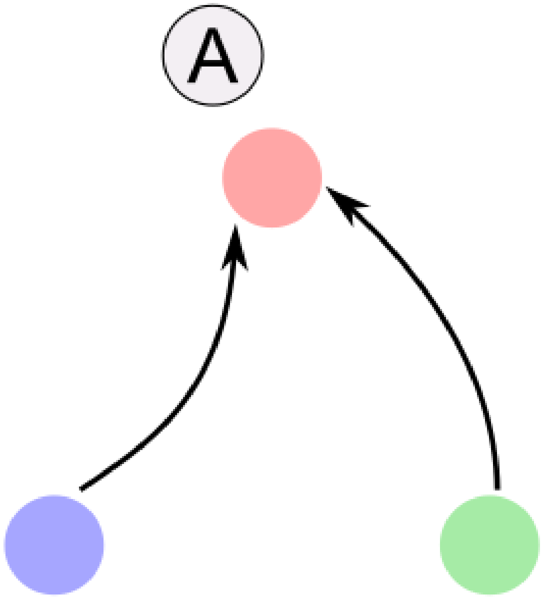
Envy-cycle elimination algorithm

[Lipton, Markakis, Mossel, and Saberi, *EC* 2004]

While there is an unallocated good

- If the envy graph has a source vertex, assign the good to that agent.
- Otherwise, resolve envy cycles until a source vertex shows up, and then assign the good to it.

	A	B	C	D
	0	2	0	1
	1	2	5	10
	1	4	2	10



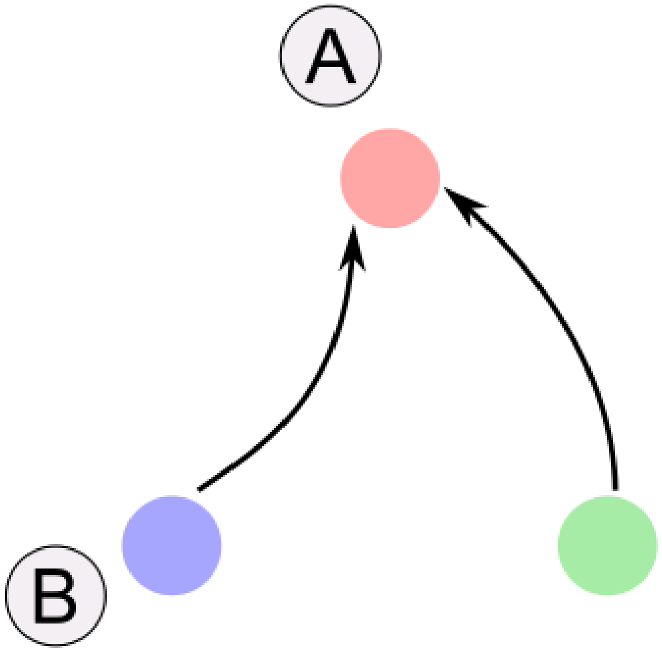
Envy-cycle elimination algorithm

[Lipton, Markakis, Mossel, and Saberi, *EC* 2004]

While there is an unallocated good

- If the envy graph has a source vertex, assign the good to that agent.
- Otherwise, resolve envy cycles until a source vertex shows up, and then assign the good to it.

	A	B	C	D
	0	2	0	1
	1	2	5	10
	1	4	2	10



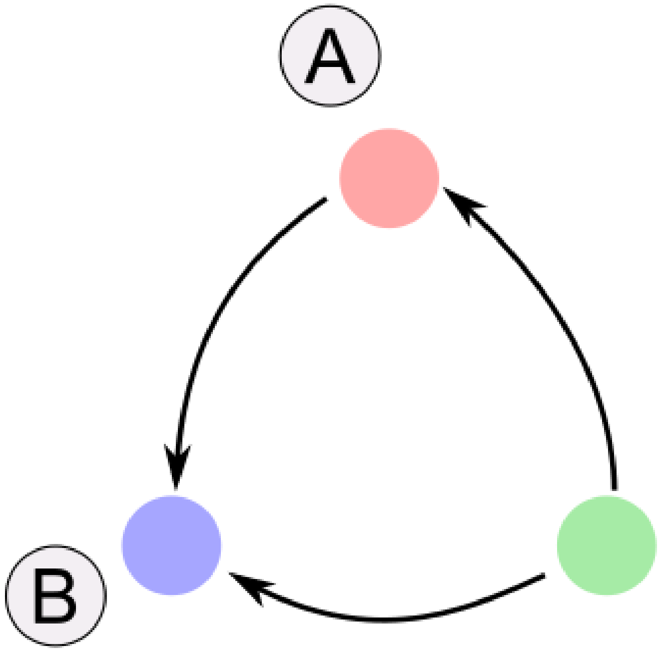
Envy-cycle elimination algorithm

[Lipton, Markakis, Mossel, and Saberi, *EC* 2004]

While there is an unallocated good

- If the envy graph has a source vertex, assign the good to that agent.
- Otherwise, resolve envy cycles until a source vertex shows up, and then assign the good to it.

	A	B	C	D
	0	2	0	1
	1	2	5	10
	1	4	2	10



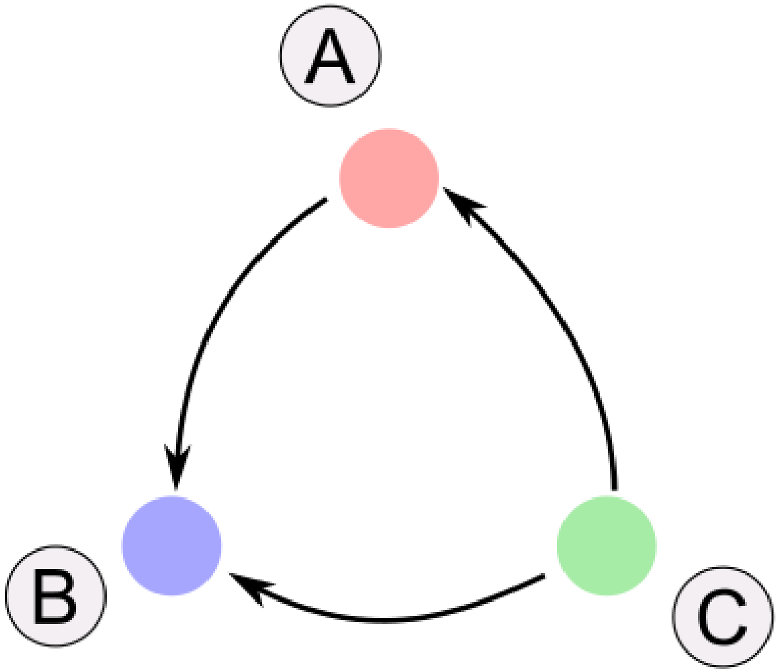
Envy-cycle elimination algorithm

[Lipton, Markakis, Mossel, and Saberi, *EC* 2004]

While there is an unallocated good

- If the envy graph has a source vertex, assign the good to that agent.
- Otherwise, resolve envy cycles until a source vertex shows up, and then assign the good to it.

	A	B	C	D
	0	2	0	1
	1	2	5	10
	1	4	2	10



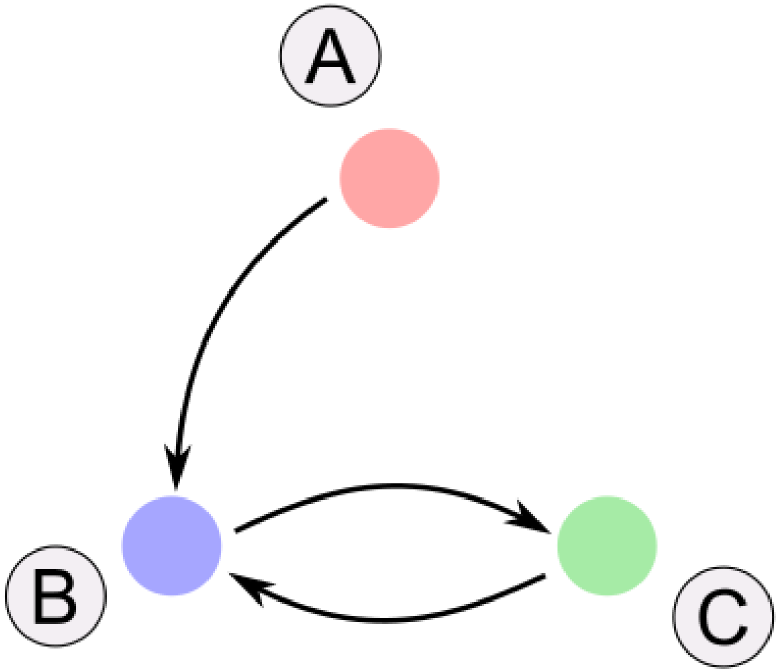
Envy-cycle elimination algorithm

[Lipton, Markakis, Mossel, and Saberi, *EC* 2004]

While there is an unallocated good

- If the envy graph has a source vertex, assign the good to that agent.
- Otherwise, resolve envy cycles until a source vertex shows up, and then assign the good to it.

	A	B	C	D
	0	2	0	1
	1	2	5	10
	1	4	2	10



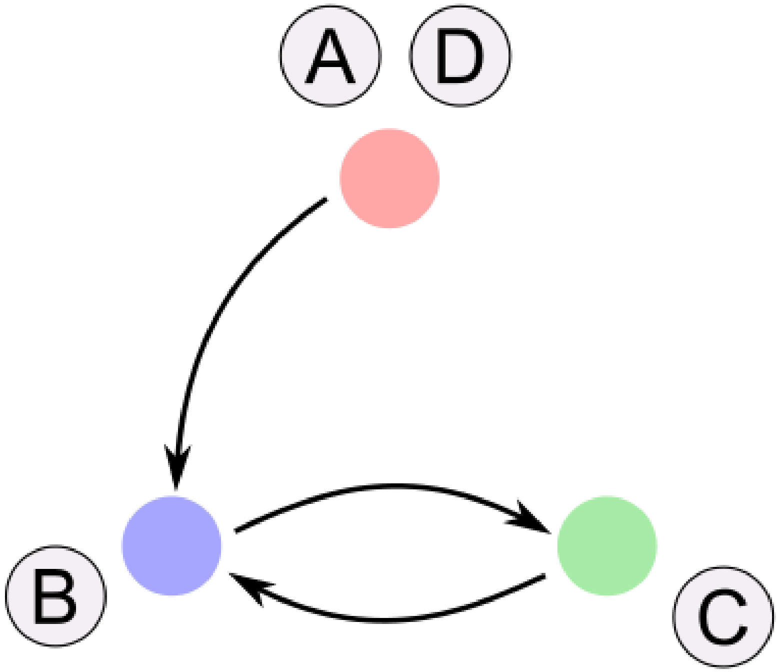
Envy-cycle elimination algorithm

[Lipton, Markakis, Mossel, and Saberi, *EC* 2004]

While there is an unallocated good

- If the envy graph has a source vertex, assign the good to that agent.
- Otherwise, resolve envy cycles until a source vertex shows up, and then assign the good to it.

	A	B	C	D
	0	2	0	1
	1	2	5	10
	1	4	2	10



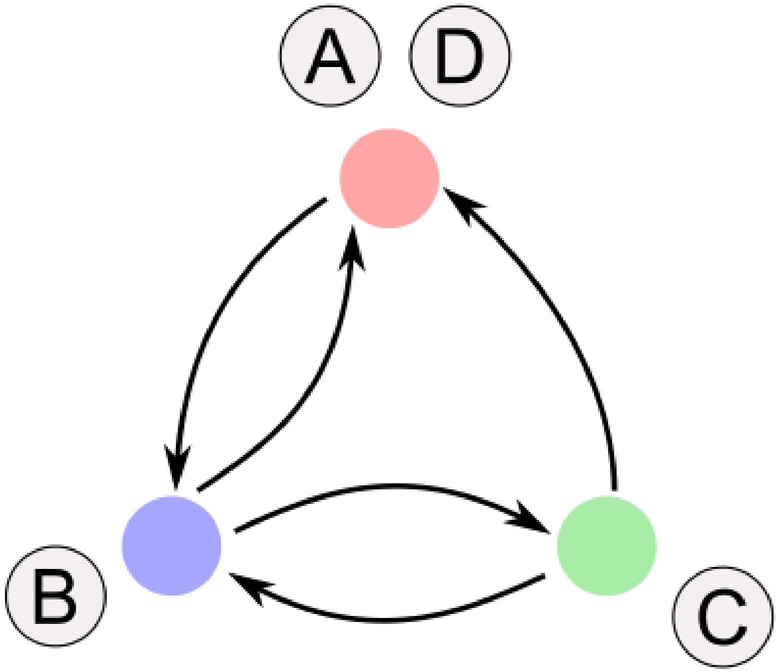
Envy-cycle elimination algorithm

[Lipton, Markakis, Mossel, and Saberi, *EC* 2004]

While there is an unallocated good

- If the envy graph has a source vertex, assign the good to that agent.
- Otherwise, resolve envy cycles until a source vertex shows up, and then assign the good to it.

	A	B	C	D
	0	2	0	1
	1	2	5	10
	1	4	2	10



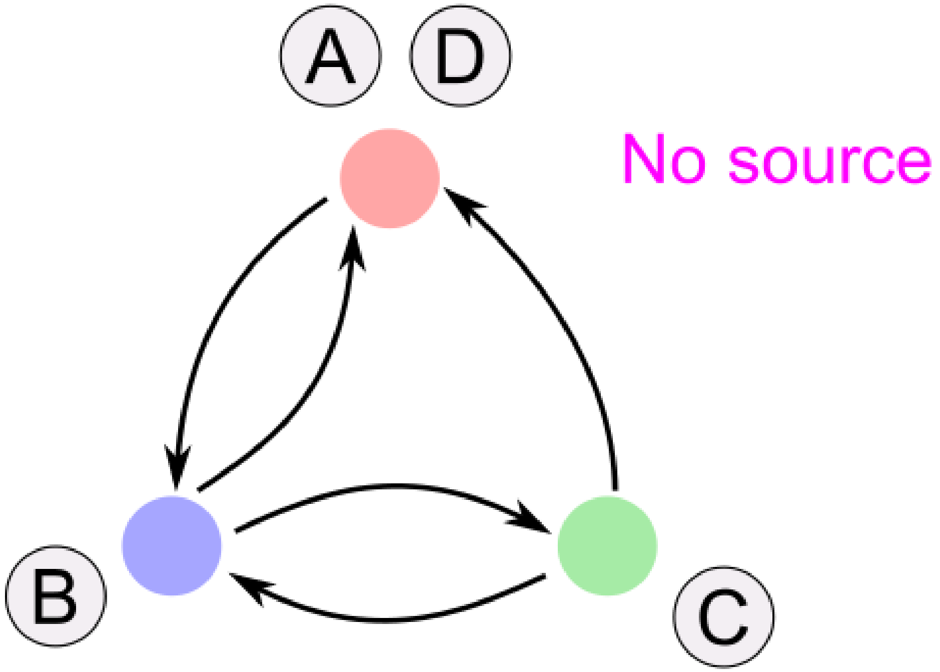
Envy-cycle elimination algorithm

[Lipton, Markakis, Mossel, and Saberi, *EC* 2004]

While there is an unallocated good

- If the envy graph has a source vertex, assign the good to that agent.
- Otherwise, resolve envy cycles until a source vertex shows up, and then assign the good to it.

	A	B	C	D
	0	2	0	1
	1	2	5	10
	1	4	2	10



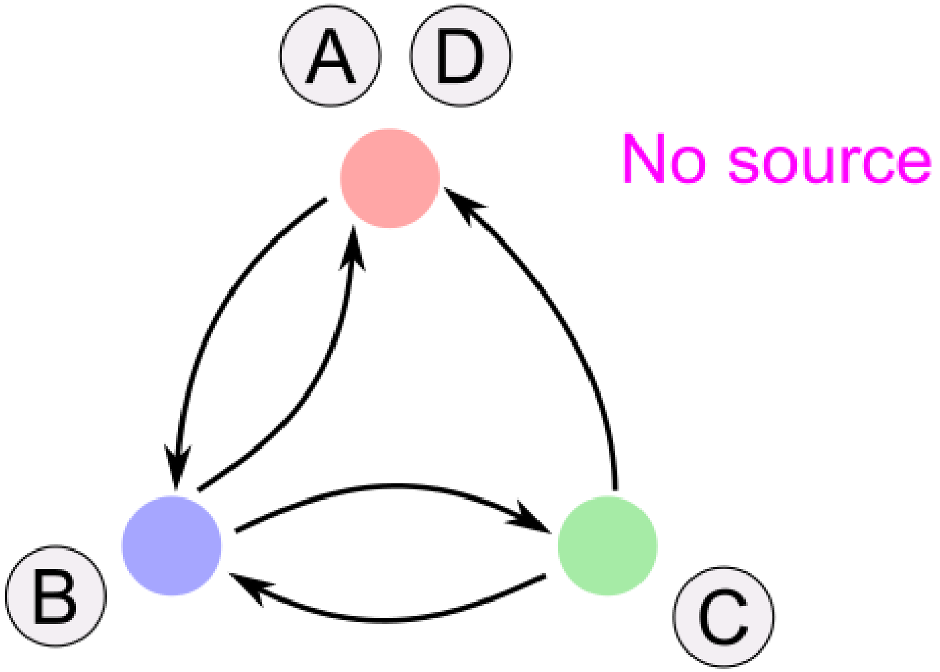
Envy-cycle elimination algorithm

[Lipton, Markakis, Mossel, and Saberi, *EC* 2004]

While there is an unallocated good

- If the envy graph has a source vertex, assign the good to that agent.
- Otherwise, resolve envy cycles until a source vertex shows up, and then assign the good to it.

	A	B	C	D	E
	0	2	0	1	1
	1	2	5	10	1
	1	4	2	10	1



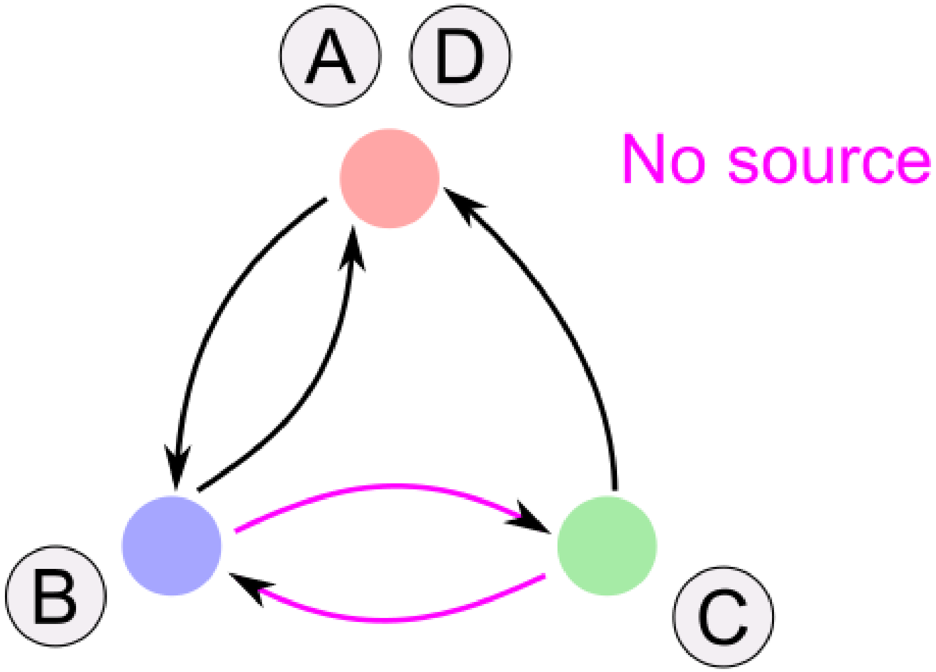
Envy-cycle elimination algorithm

[Lipton, Markakis, Mossel, and Saberi, *EC* 2004]

While there is an unallocated good

- If the envy graph has a source vertex, assign the good to that agent.
- Otherwise, resolve envy cycles until a source vertex shows up, and then assign the good to it.

	A	B	C	D	E
Red agent	0	2	0	1	1
Blue agent	1	2	5	10	1
Green agent	1	4	2	10	1



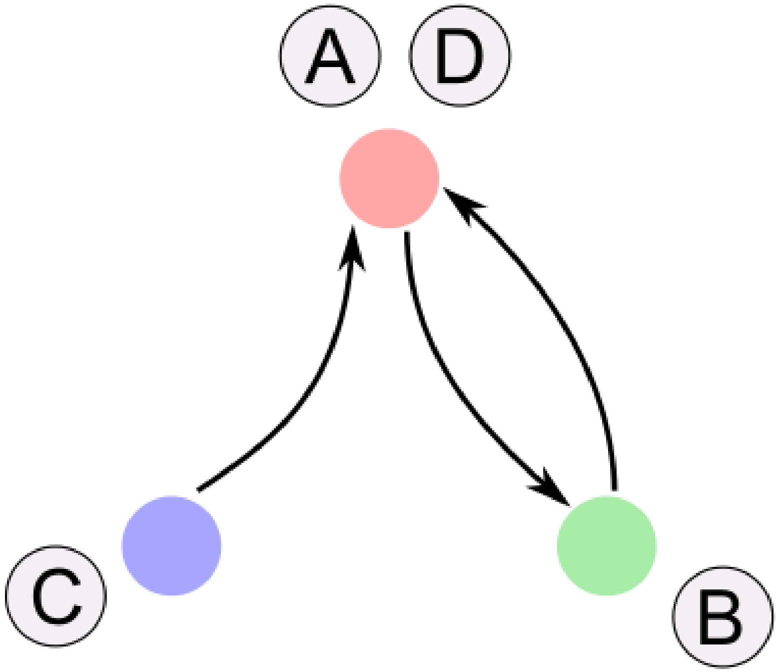
Envy-cycle elimination algorithm

[Lipton, Markakis, Mossel, and Saberi, *EC* 2004]

While there is an unallocated good

- If the envy graph has a source vertex, assign the good to that agent.
- Otherwise, resolve envy cycles until a source vertex shows up, and then assign the good to it.

	A	B	C	D	E
	0	2	0	1	1
	1	2	5	10	1
	1	4	2	10	1



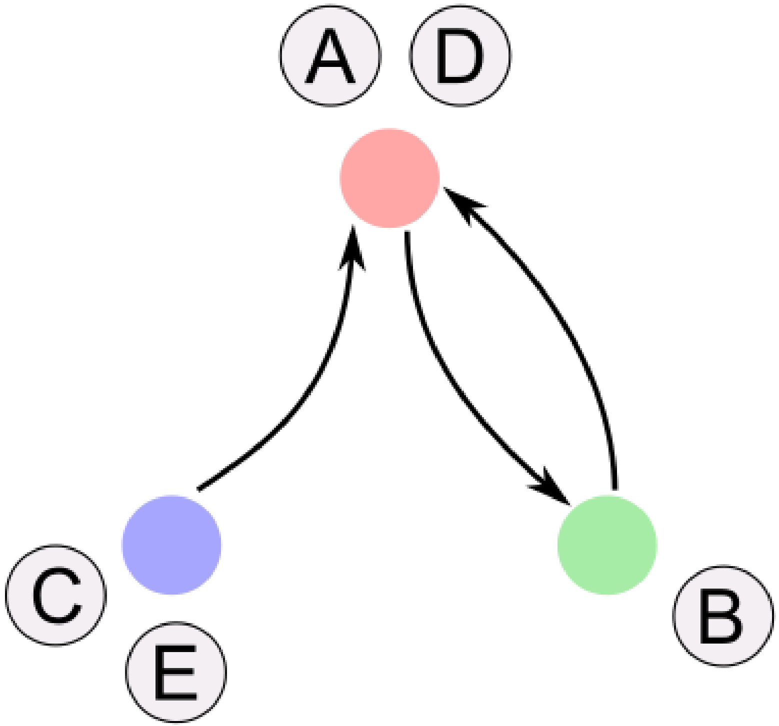
Envy-cycle elimination algorithm

[Lipton, Markakis, Mossel, and Saberi, *EC* 2004]

While there is an unallocated good

- If the envy graph has a source vertex, assign the good to that agent.
- Otherwise, resolve envy cycles until a source vertex shows up, and then assign the good to it.

	A	B	C	D	E
	0	2	0	1	1
	1	2	5	10	1
	1	4	2	10	1



Envy-cycle elimination algorithm

[Lipton, Markakis, Mossel, and Saberi, *EC* 2004]

1 Does the algorithm terminate?

2 Does the algorithm terminate in polynomial time?

3 Does the allocation returned by the algorithm satisfy EF1?

Envy-cycle elimination algorithm

[Lipton, Markakis, Mossel, and Saberi, *EC* 2004]

1. Does the algorithm terminate?

2. Does the algorithm terminate in polynomial time?

3. Does the allocation returned by the algorithm satisfy EF1?

Does envy-cycle elimination algorithm terminate?

Does envy-cycle elimination algorithm terminate?

Suffices to show that after a finite number of envy-cycle resolution operations, a source vertex must appear.

Does envy-cycle elimination algorithm terminate?

Suffices to show that after a finite number of envy-cycle resolution operations, a source vertex must appear.

We will show that:

Does envy-cycle elimination algorithm terminate?

Suffices to show that after a finite number of envy-cycle resolution operations, a source vertex must appear.

We will show that:

After resolving any envy cycle, the total number of edges in the envy graph strictly decreases.

Does envy-cycle elimination algorithm terminate?

Suffices to show that after a finite number of envy-cycle resolution operations, a source vertex must appear.

We will show that:

After resolving any envy cycle, the total number of edges in the envy graph strictly decreases.

- With n agents, at most $O(n^2)$ cycle resolutions required to create a source.

Does envy-cycle elimination algorithm terminate?

Suffices to show that after a finite number of envy-cycle resolution operations, a source vertex must appear.

We will show that:

> After resolving any envy cycle, the total number of edges in the envy graph strictly decreases.

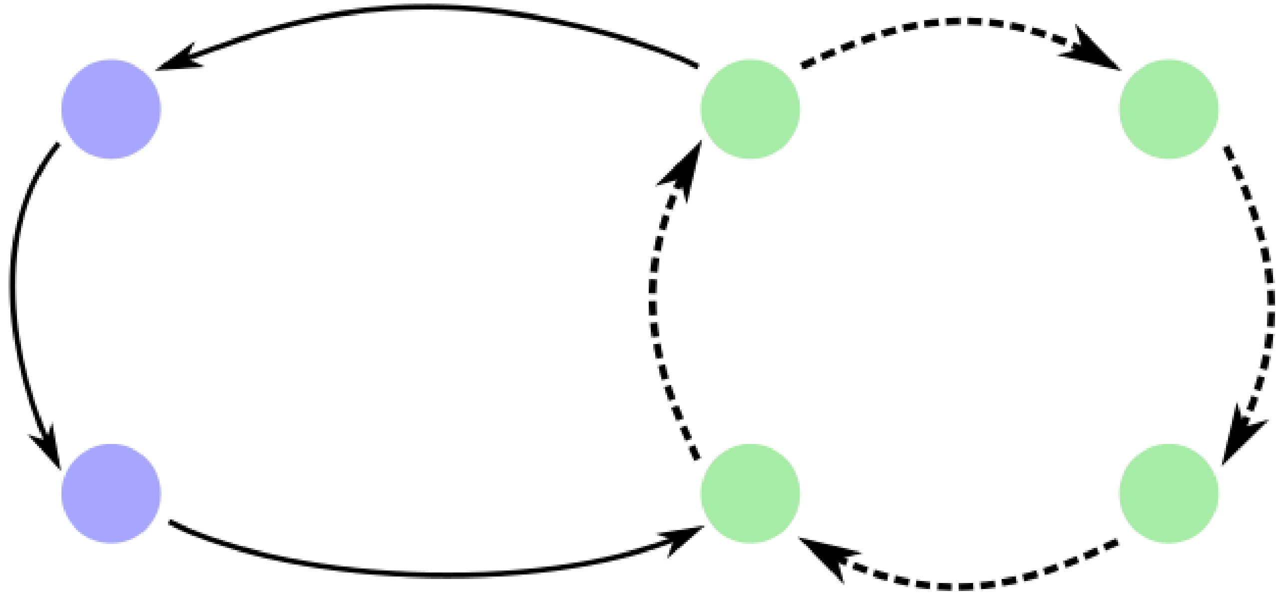
- With n agents, at most $O(n^2)$ cycle resolutions required to create a source.
- Polynomial running time!

Does envy-cycle elimination algorithm terminate?

After resolving any envy cycle, the total number of edges in the envy graph strictly decreases.

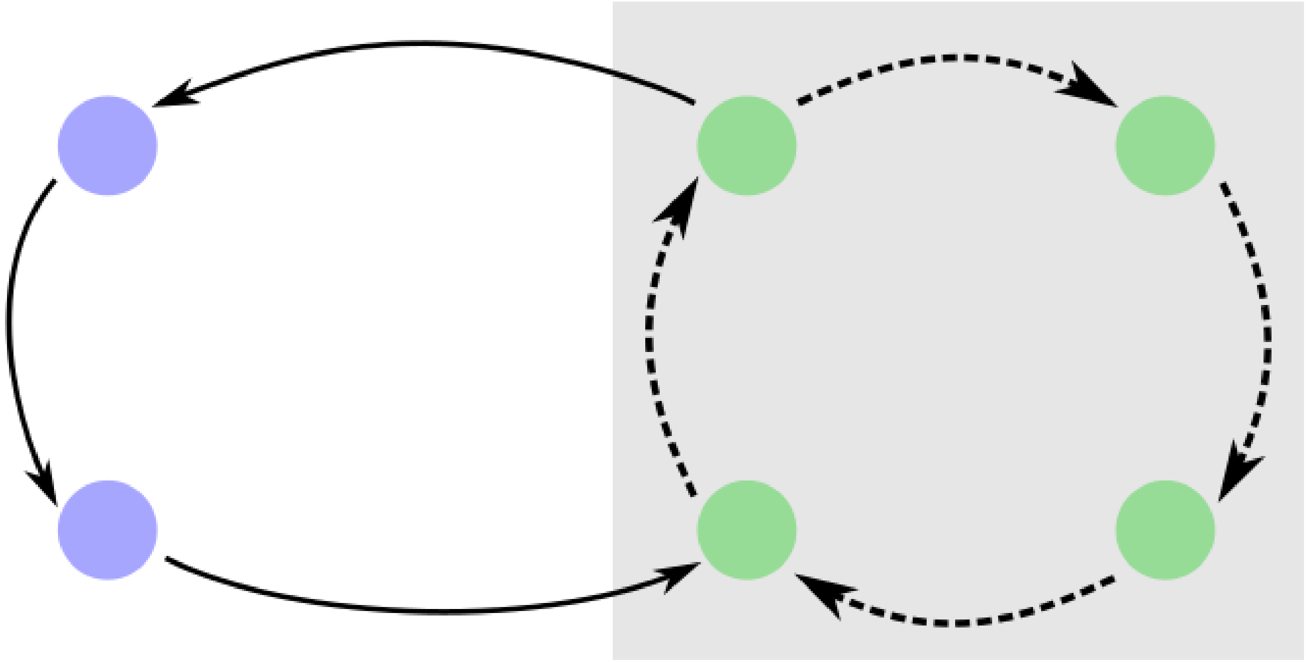
Does envy-cycle elimination algorithm terminate?

After resolving any envy cycle, the total number of edges in the envy graph strictly decreases.



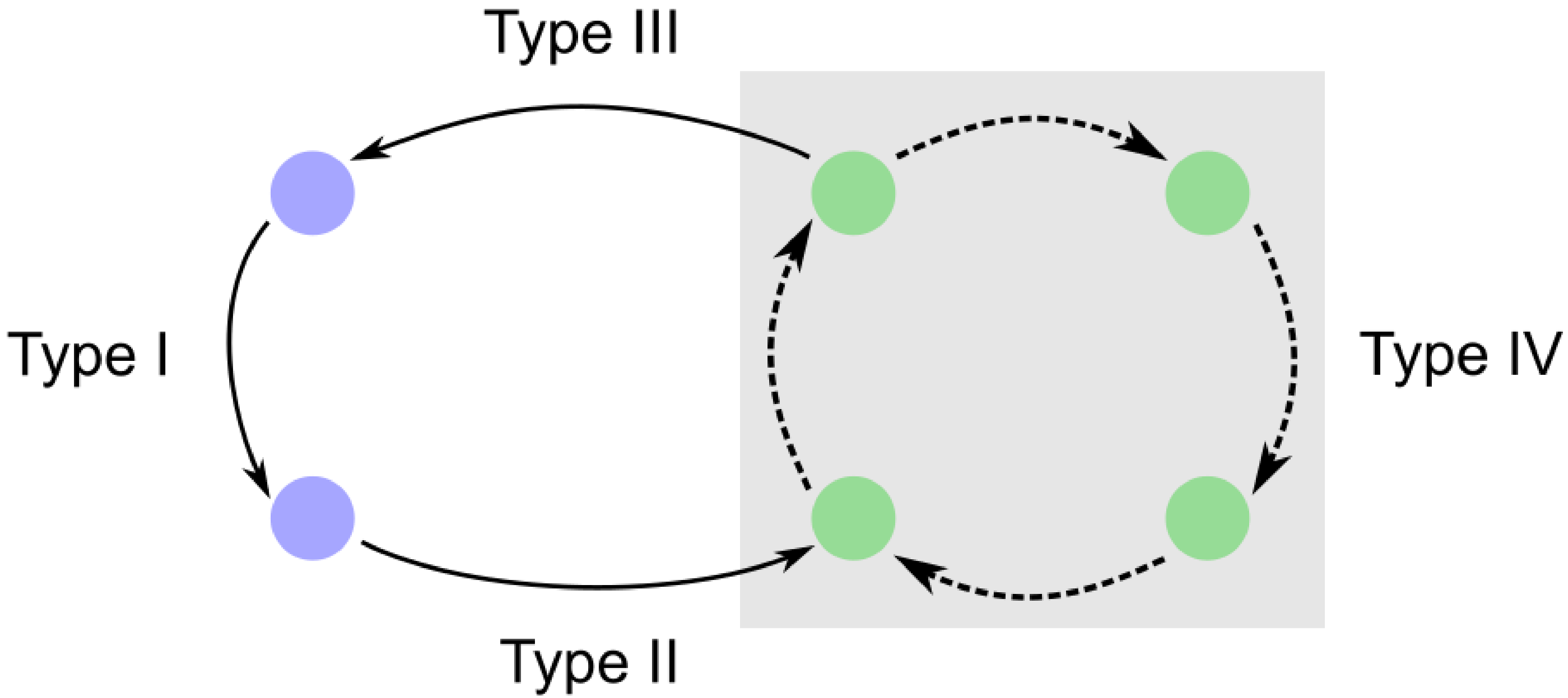
Does envy-cycle elimination algorithm terminate?

After resolving any envy cycle, the total number of edges in the envy graph strictly decreases.



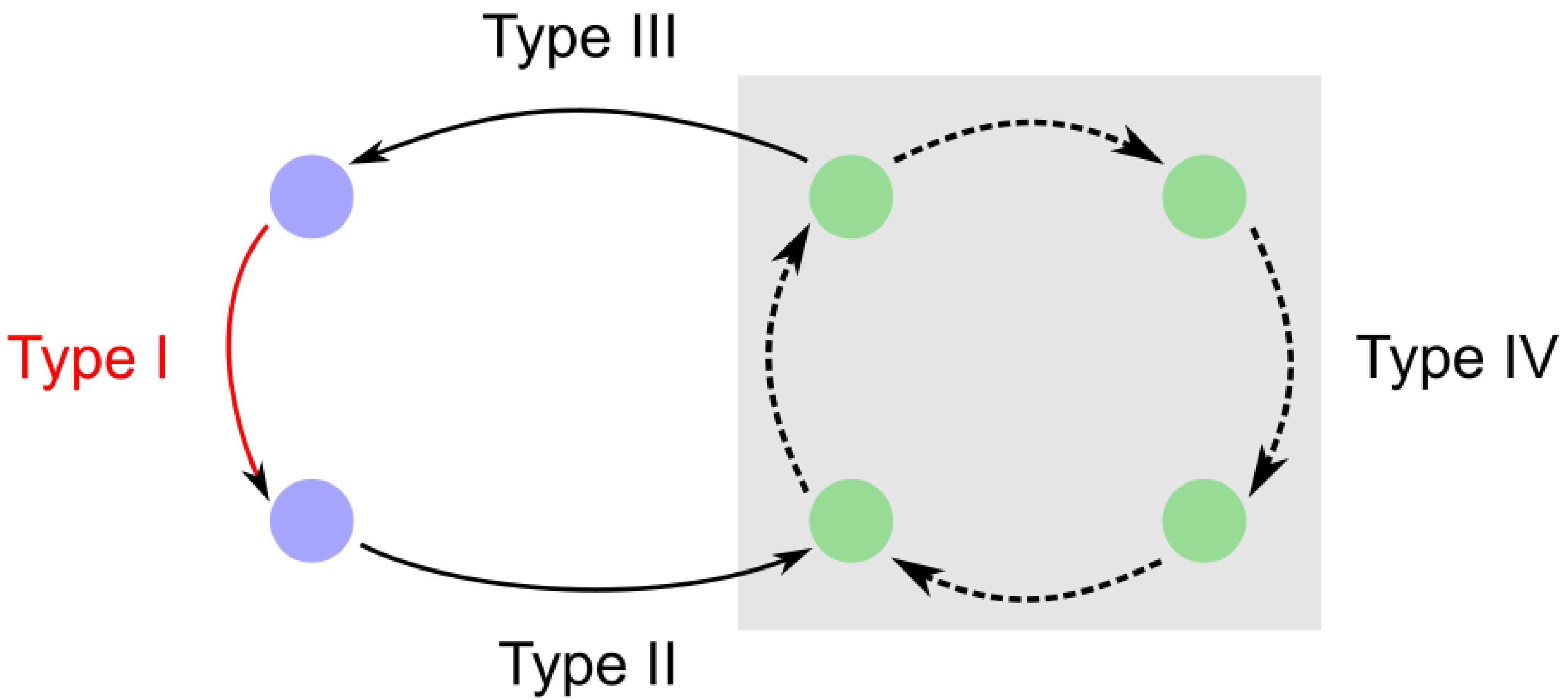
Does envy-cycle elimination algorithm terminate?

After resolving any envy cycle, the total number of edges in the envy graph strictly decreases.



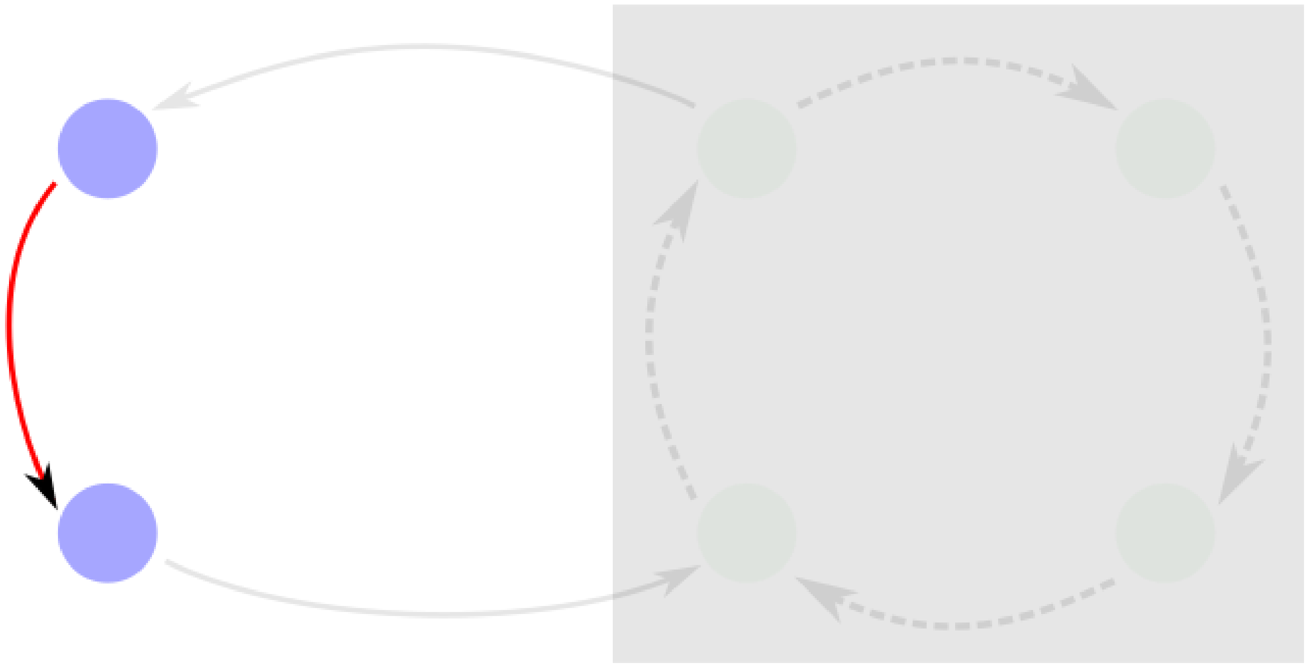
Does envy-cycle elimination algorithm terminate?

After resolving any envy cycle, the total number of edges in the envy graph strictly decreases.



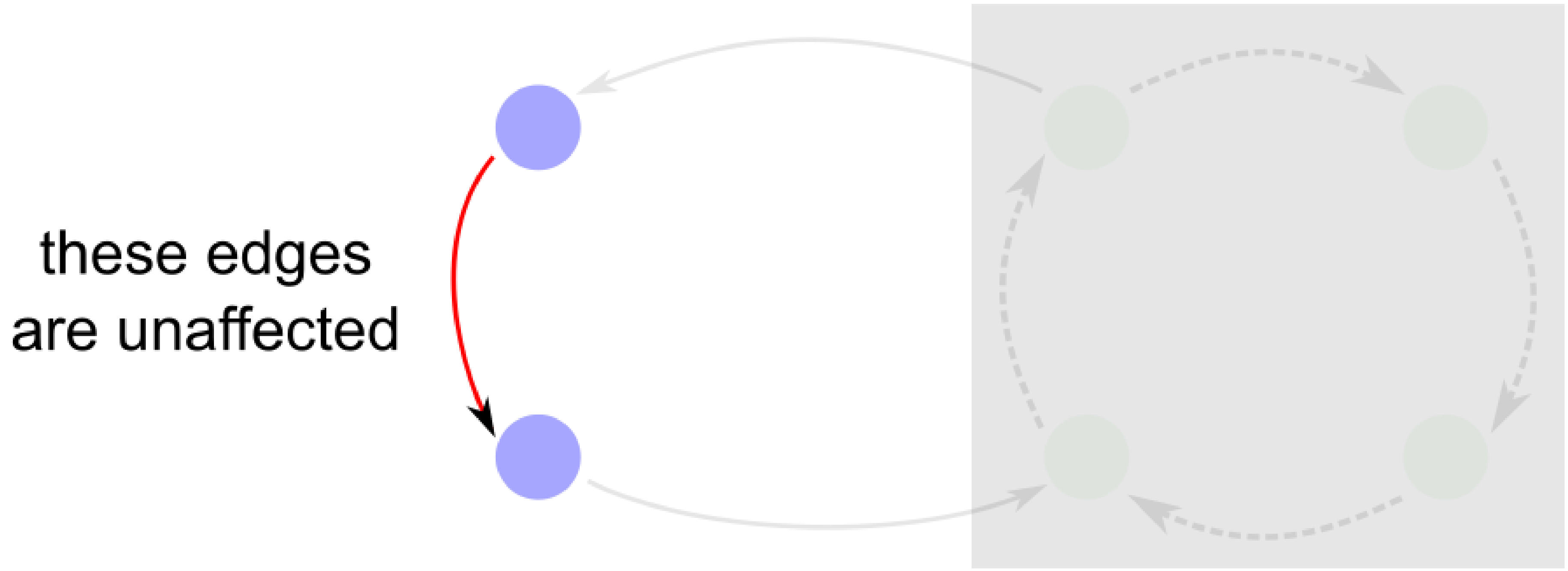
Does envy-cycle elimination algorithm terminate?

After resolving any envy cycle, the total number of edges in the envy graph strictly decreases.



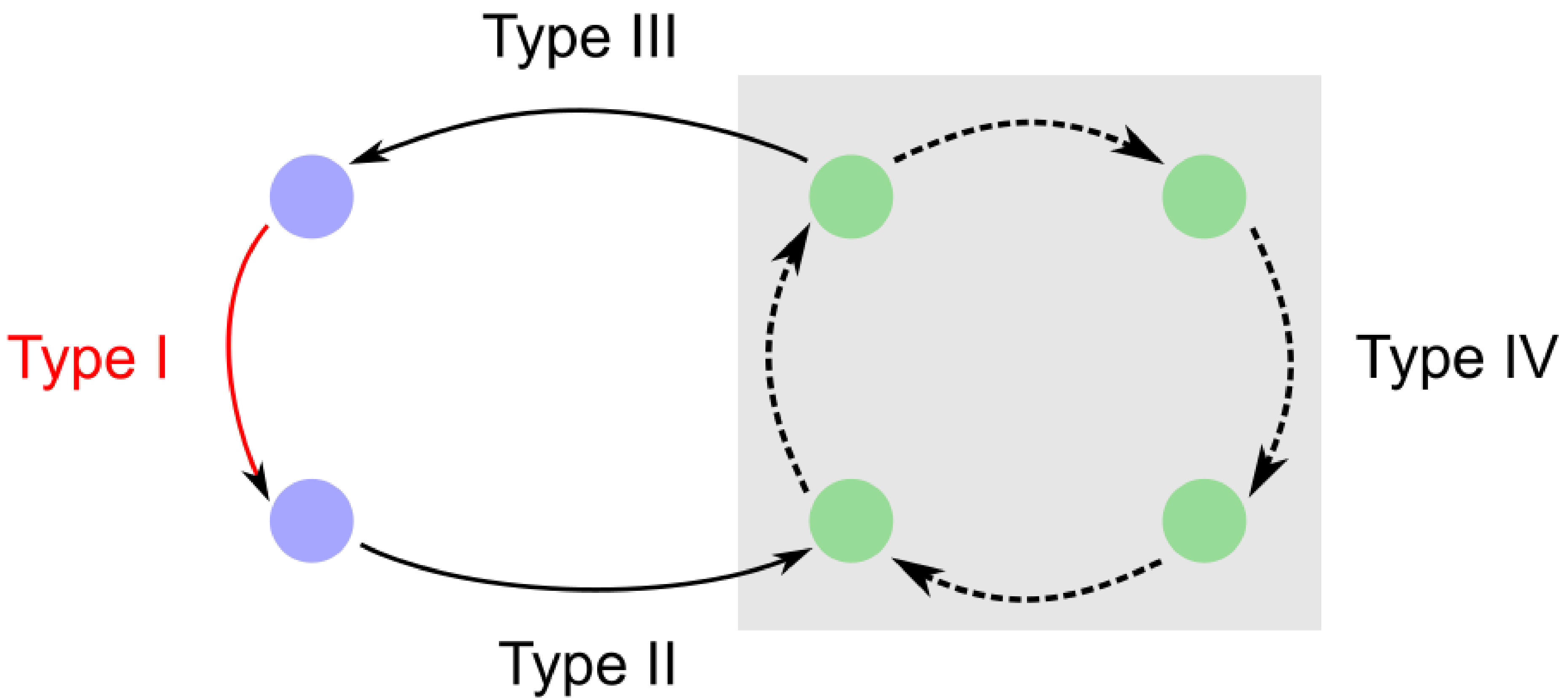
Does envy-cycle elimination algorithm terminate?

After resolving any envy cycle, the total number of edges in the envy graph strictly decreases.



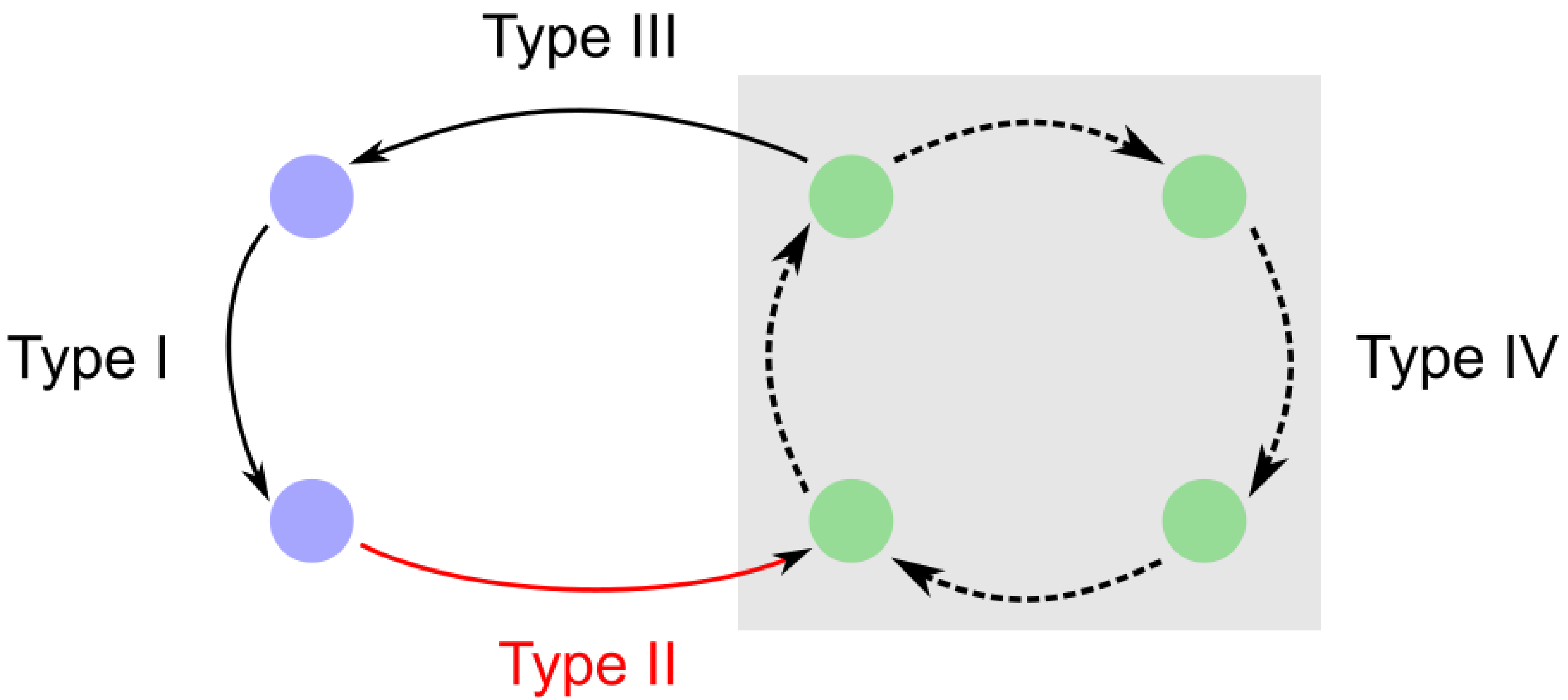
Does envy-cycle elimination algorithm terminate?

After resolving any envy cycle, the total number of edges in the envy graph strictly decreases.



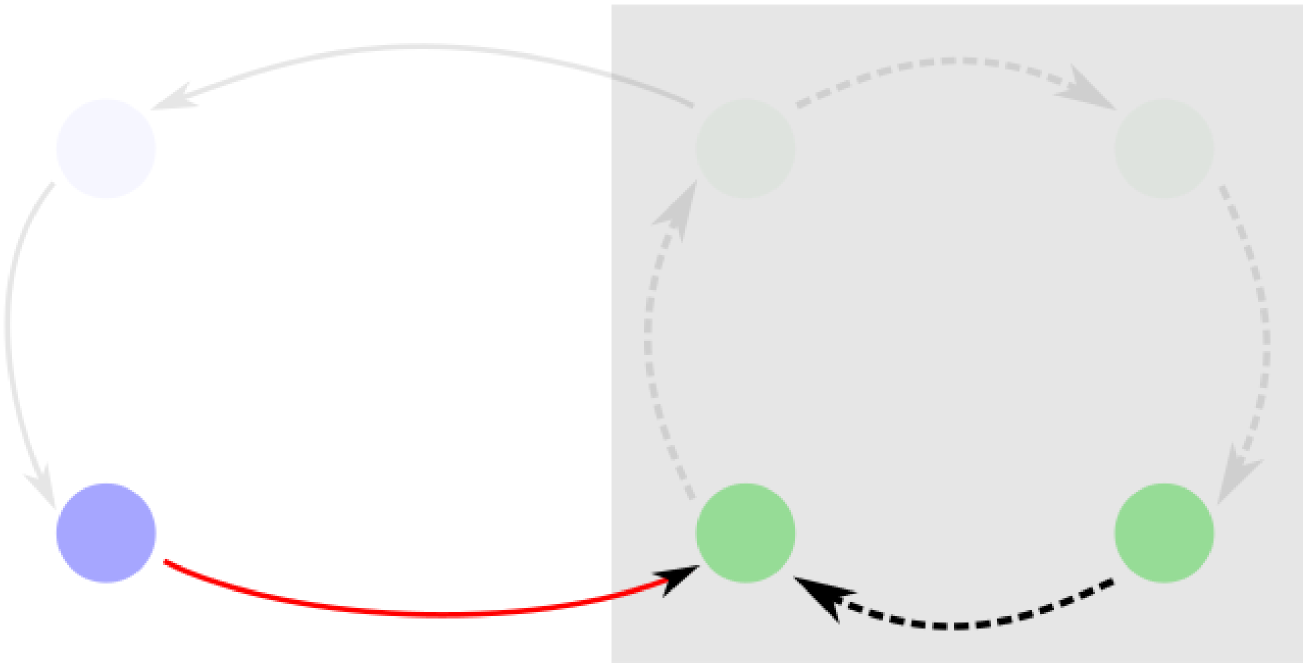
Does envy-cycle elimination algorithm terminate?

After resolving any envy cycle, the total number of edges in the envy graph strictly decreases.



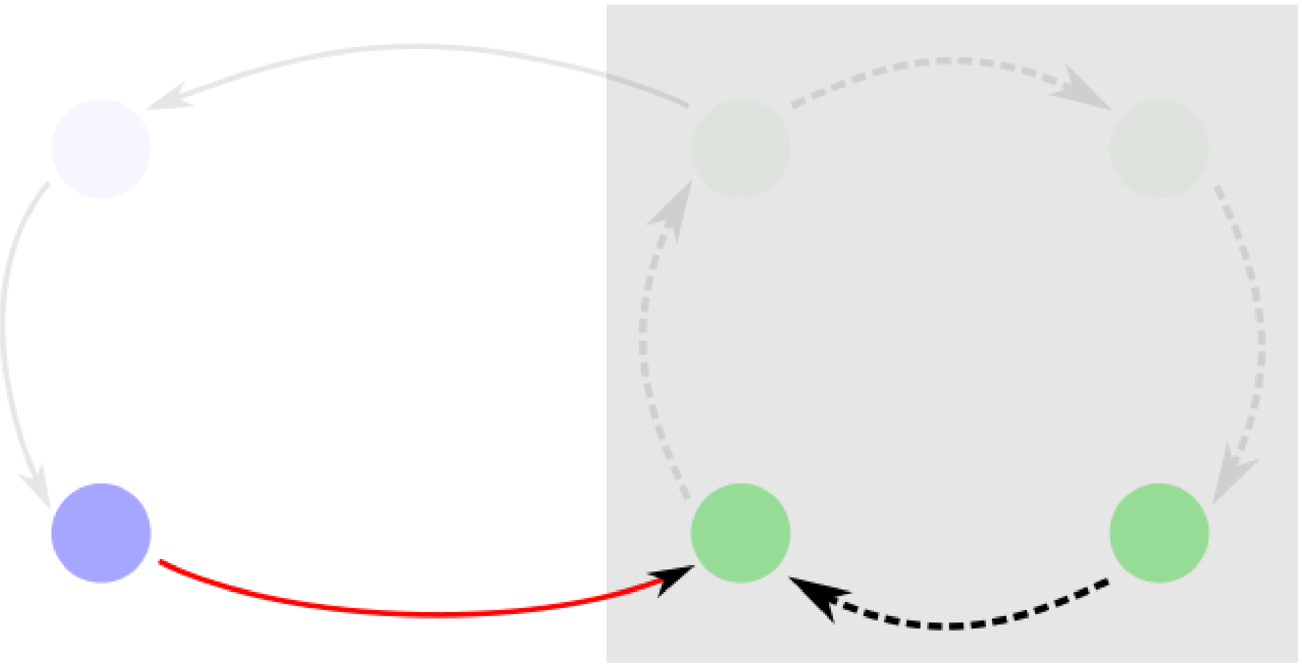
Does envy-cycle elimination algorithm terminate?

After resolving any envy cycle, the total number of edges in the envy graph strictly decreases.



Does envy-cycle elimination algorithm terminate?

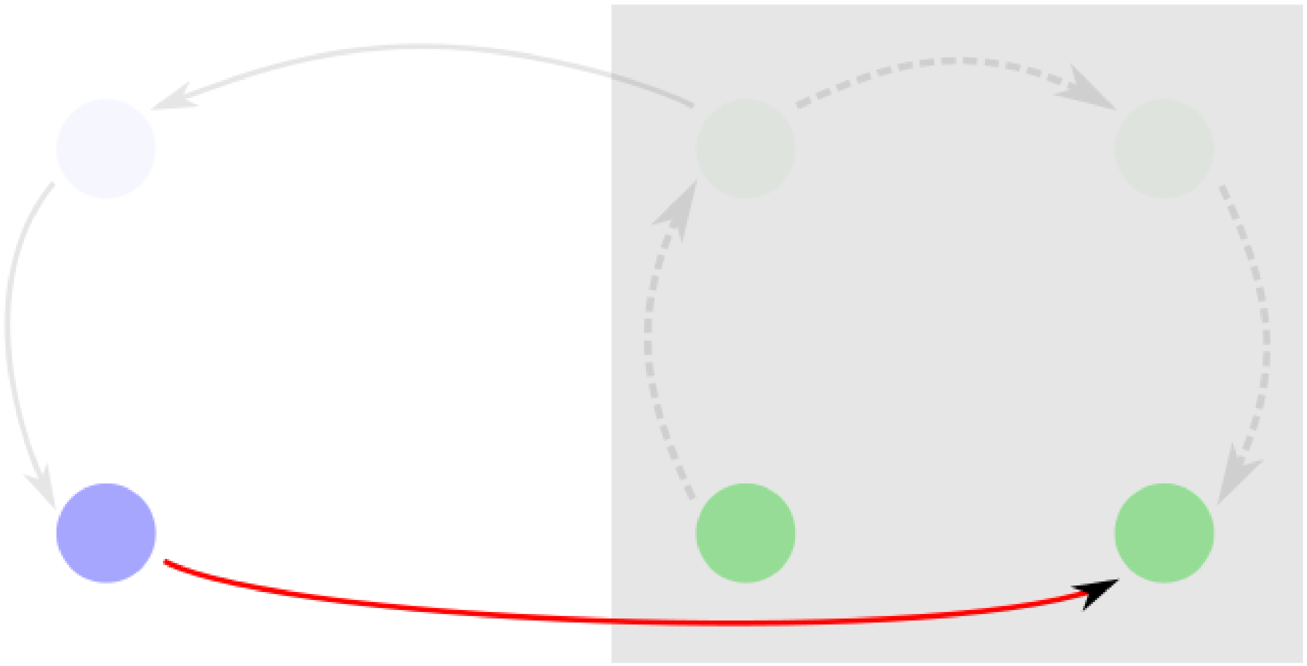
After resolving any envy cycle, the total number of edges in the envy graph strictly decreases.



these edges are shifted

Does envy-cycle elimination algorithm terminate?

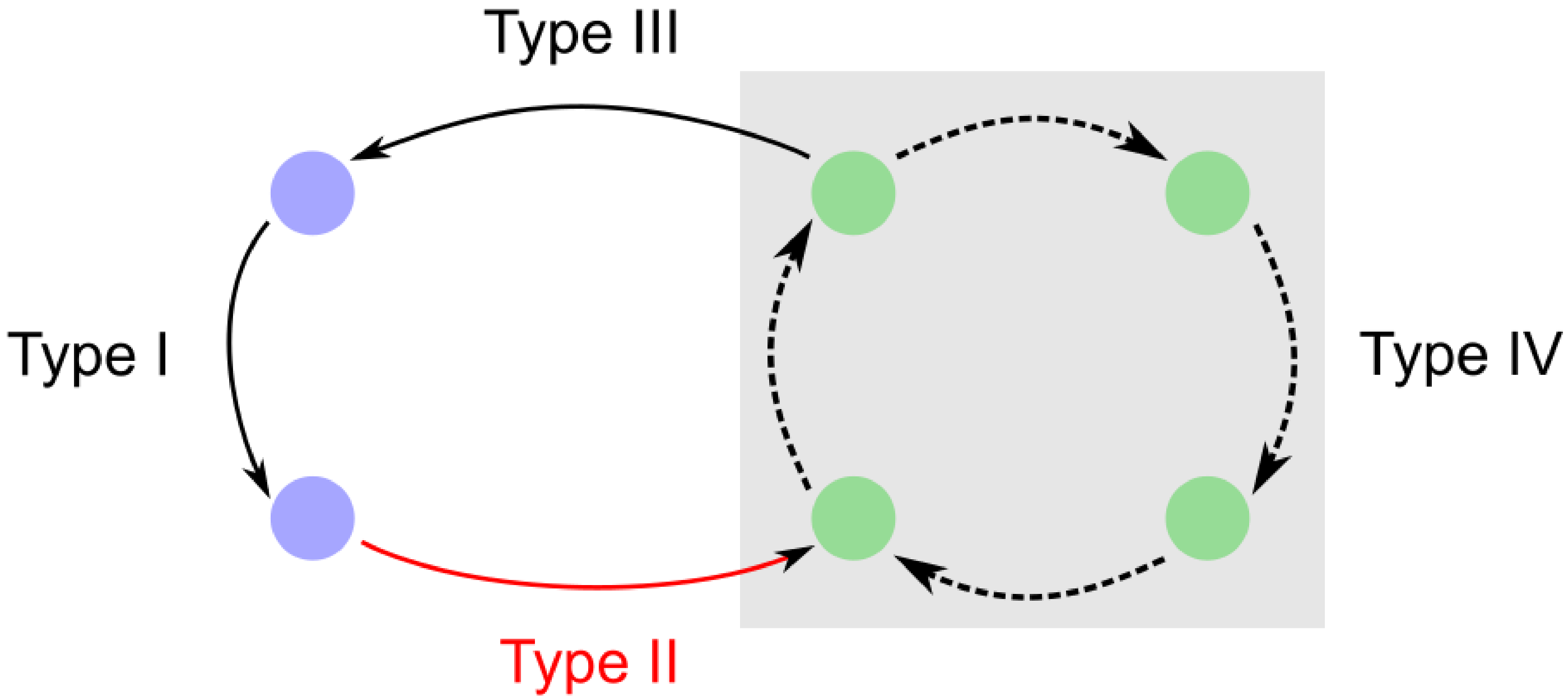
After resolving any envy cycle, the total number of edges in the envy graph strictly decreases.



these edges are shifted

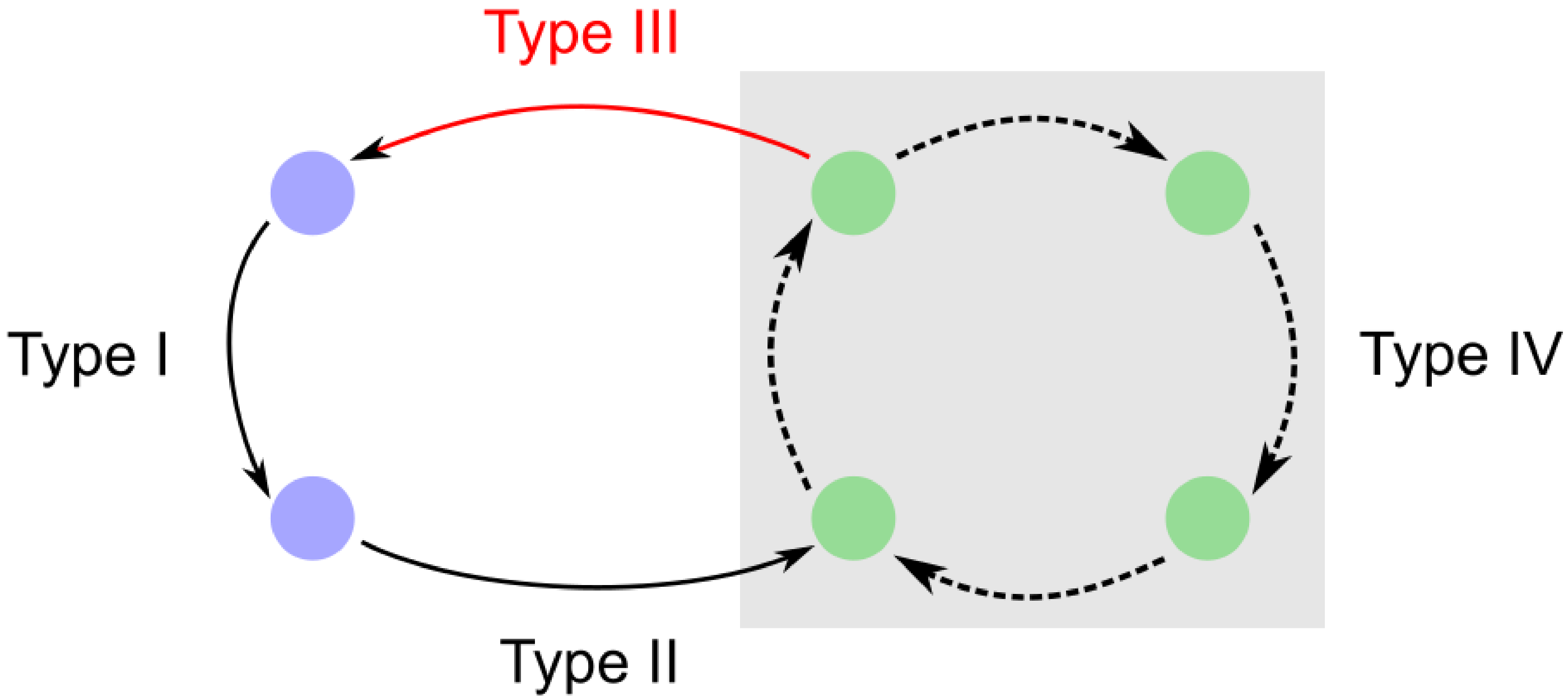
Does envy-cycle elimination algorithm terminate?

After resolving any envy cycle, the total number of edges in the envy graph strictly decreases.



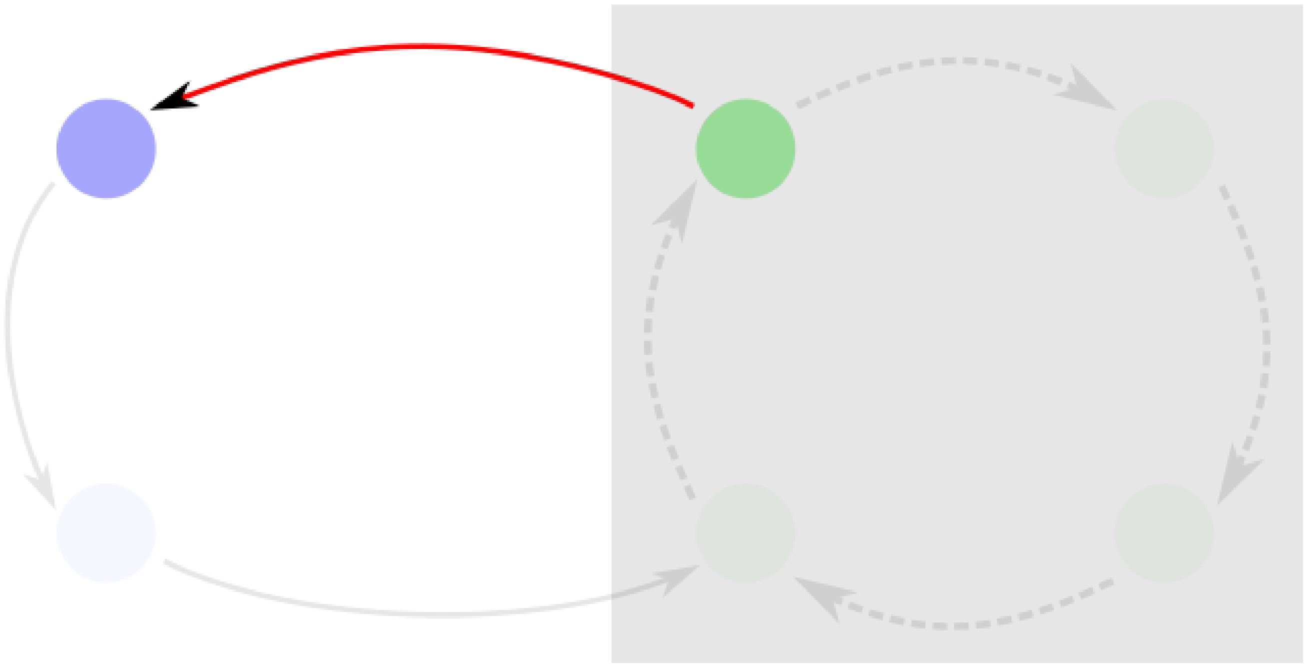
Does envy-cycle elimination algorithm terminate?

After resolving any envy cycle, the total number of edges in the envy graph strictly decreases.



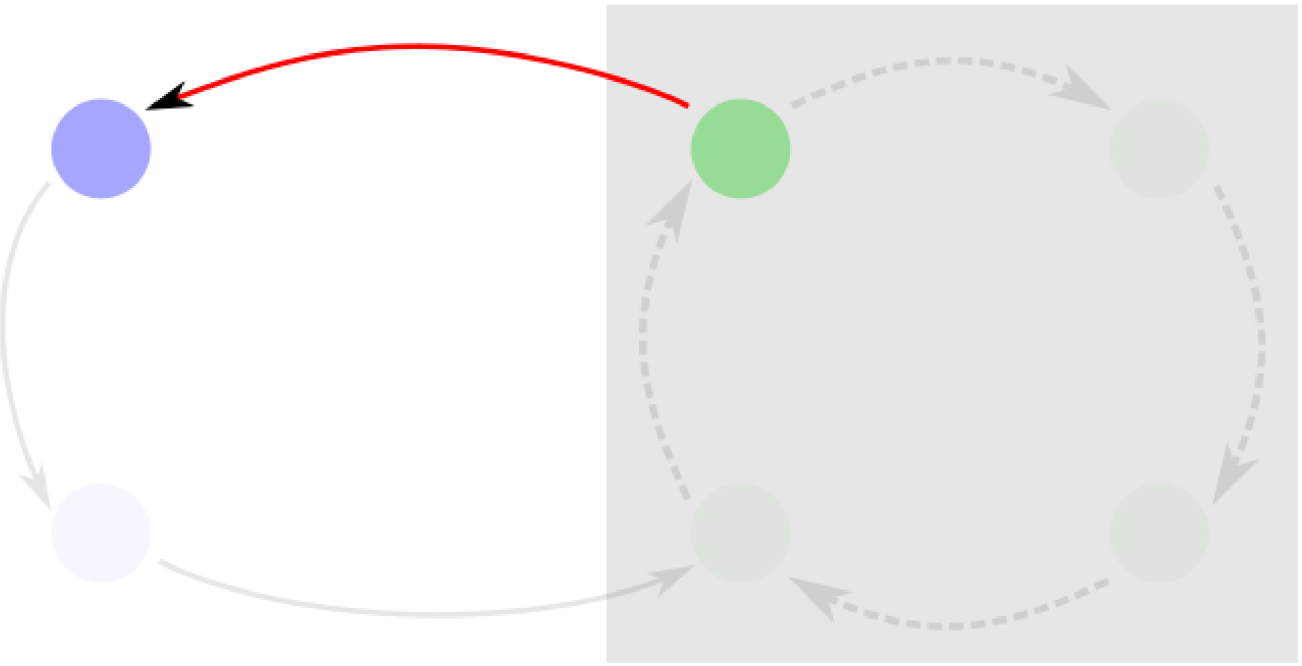
Does envy-cycle elimination algorithm terminate?

After resolving any envy cycle, the total number of edges in the envy graph strictly decreases.



Does envy-cycle elimination algorithm terminate?

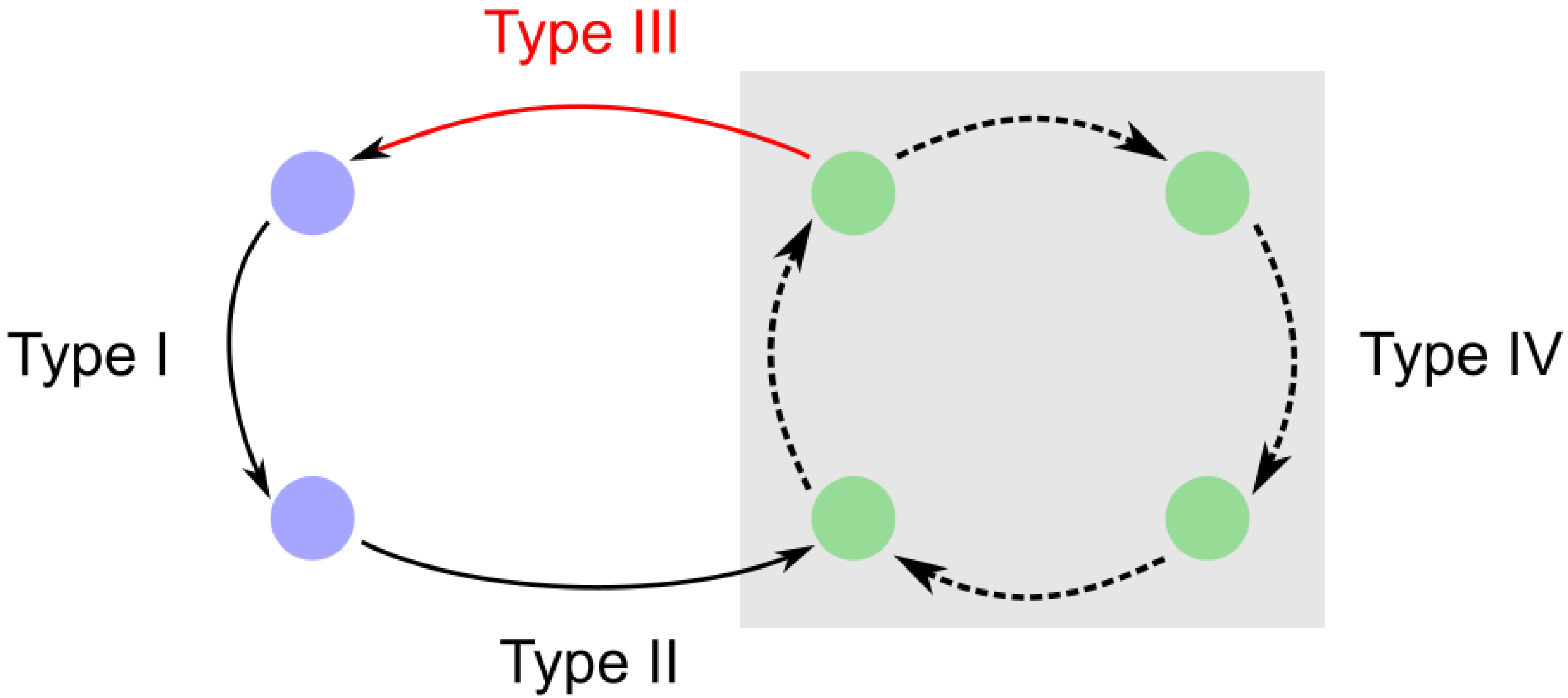
After resolving any envy cycle, the total number of edges in the envy graph strictly decreases.



these edges can either stay or disappear (no new such edges are added)

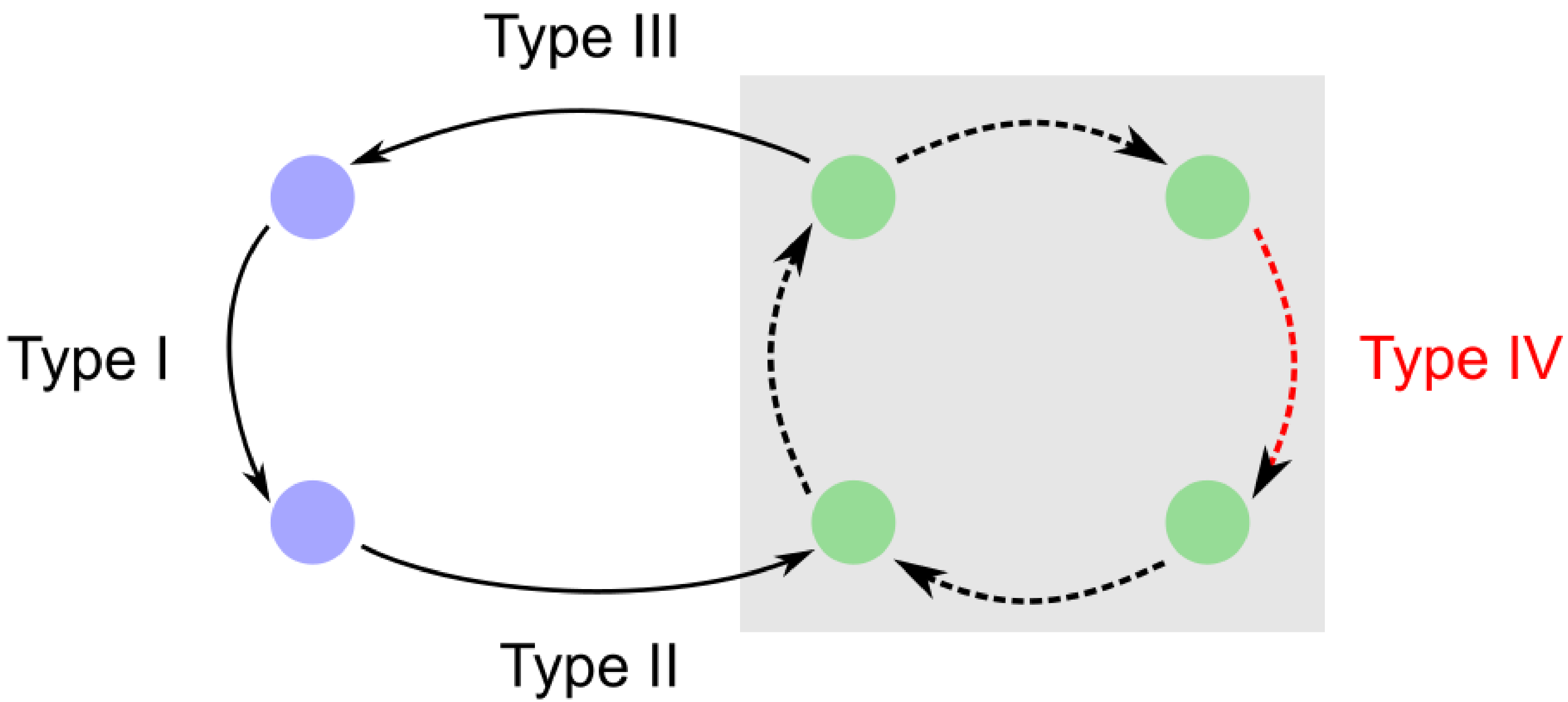
Does envy-cycle elimination algorithm terminate?

After resolving any envy cycle, the total number of edges in the envy graph strictly decreases.



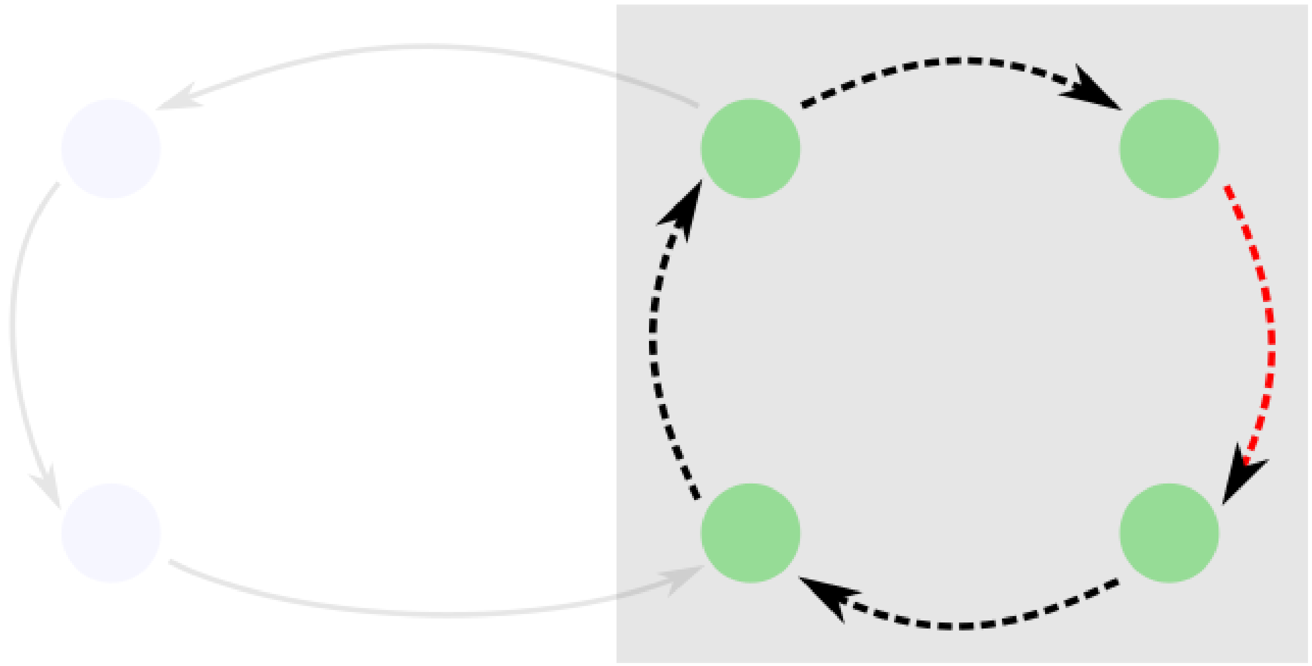
Does envy-cycle elimination algorithm terminate?

After resolving any envy cycle, the total number of edges in the envy graph strictly decreases.



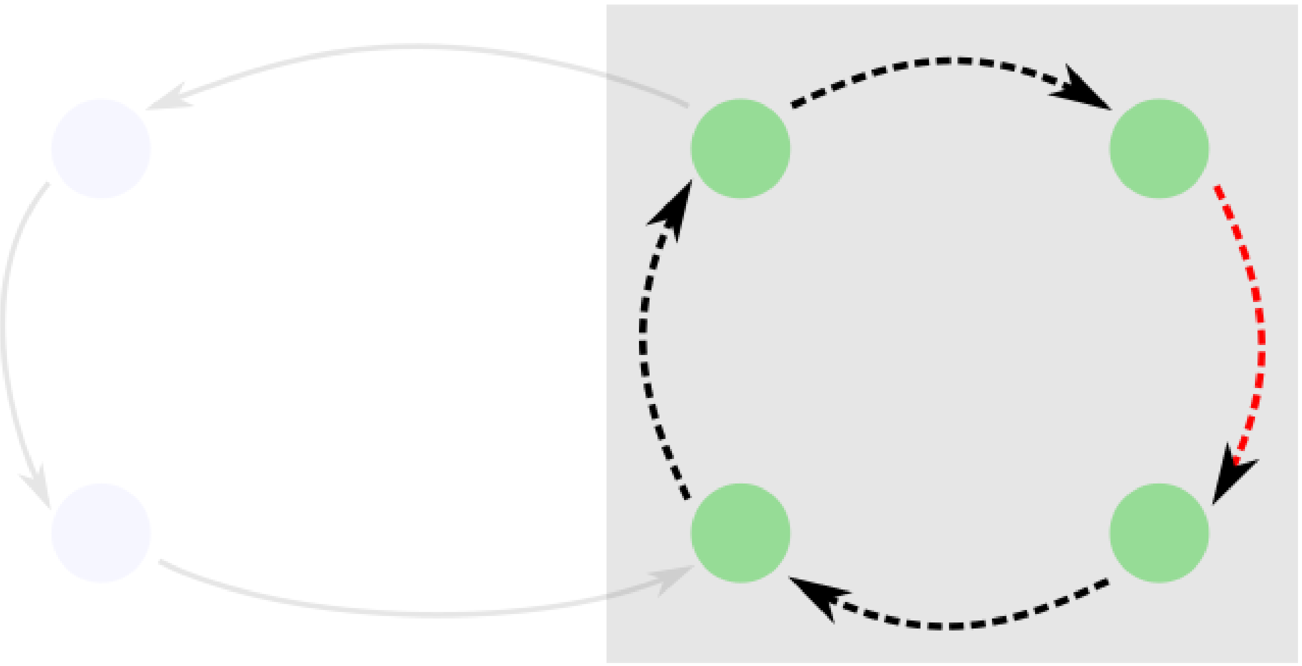
Does envy-cycle elimination algorithm terminate?

After resolving any envy cycle, the total number of edges in the envy graph strictly decreases.



Does envy-cycle elimination algorithm terminate?

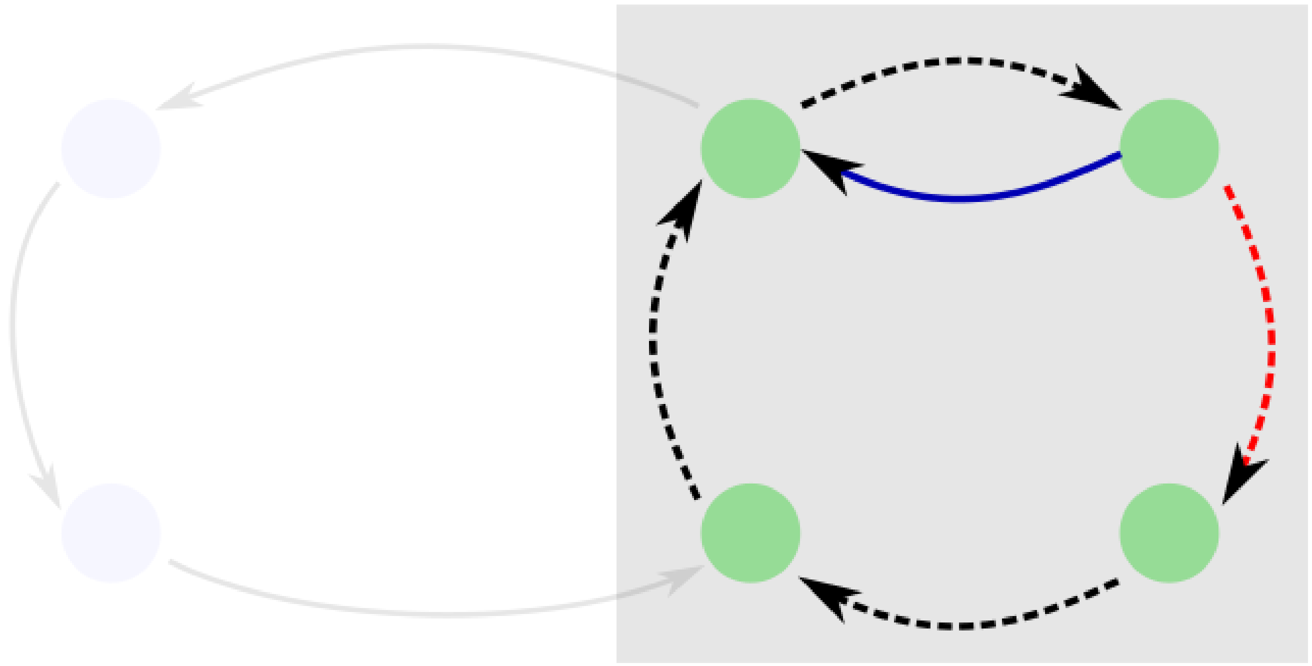
After resolving any envy cycle, the total number of edges in the envy graph strictly decreases.



For any agent in the envy cycle, its envy edge to its neighbor must disappear and any other envy edges can either get shifted or should disappear.

Does envy-cycle elimination algorithm terminate?

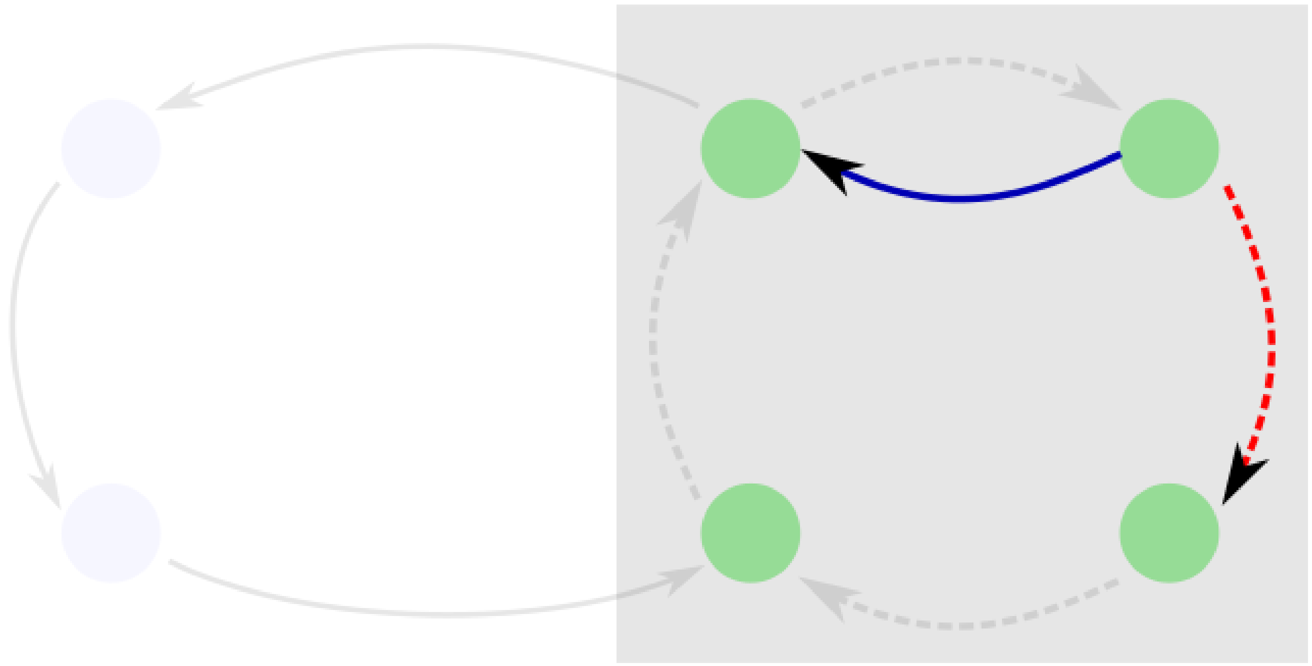
After resolving any envy cycle, the total number of edges in the envy graph strictly decreases.



For any agent in the envy cycle, its envy edge to its neighbor must disappear and any other envy edges can either get shifted or should disappear.

Does envy-cycle elimination algorithm terminate?

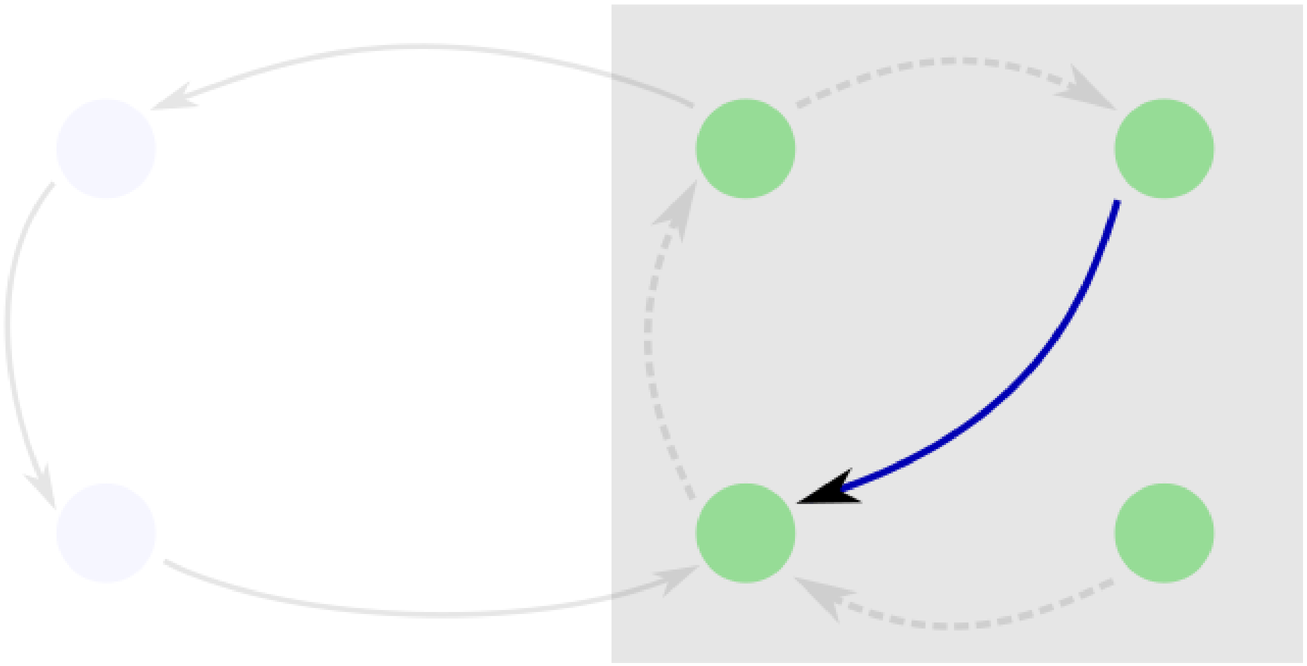
After resolving any envy cycle, the total number of edges in the envy graph strictly decreases.



For any agent in the envy cycle, its envy edge to its neighbor must disappear and any other envy edges can either get shifted or should disappear.

Does envy-cycle elimination algorithm terminate?

After resolving any envy cycle, the total number of edges in the envy graph strictly decreases.



For any agent in the envy cycle, its envy edge to its neighbor must disappear and any other envy edges can either get shifted or should disappear.

Envy-cycle elimination algorithm

[Lipton, Markakis, Mossel, and Saberi, *EC* 2004]

1 Does the algorithm terminate?

2 Does the algorithm terminate in polynomial time?

3 Does the allocation returned by the algorithm satisfy EF1?

Envy-cycle elimination algorithm

[Lipton, Markakis, Mossel, and Saberi, *EC* 2004]

1 Does the algorithm terminate?

2 Does the algorithm terminate in polynomial time?

3 Does the allocation returned by the algorithm satisfy EF1?

Envy-cycle elimination algorithm

[Lipton, Markakis, Mossel, and Saberi, *EC* 2004]

1 Does the algorithm terminate?

2 Does the algorithm terminate in polynomial time?

3 Does the allocation returned by the algorithm satisfy EF1?

Does envy-cycle elimination algorithm return an EF1 allocation?

Does envy-cycle elimination algorithm return an EF1 allocation?

Allocation A is EF1 if for every pair of agents i, k, there exists a good $j \in A_k$ such that $v_i(A_i) \geq v_i(A_k \setminus \{j\})$.

Does envy-cycle elimination algorithm return an EF1 allocation?

We will argue that each iteration of the algorithm "preserves" EF1.

Does envy-cycle elimination algorithm return an EF1 allocation?

We will argue that each iteration of the algorithm "preserves" EF1.

> If the partial allocation at the beginning of an iteration is EF1, then the partial allocation at the end of that iteration is also EF1.

Does envy-cycle elimination algorithm return an EF1 allocation?

We will argue that each iteration of the algorithm "preserves" EF1.

While there is an unallocated good

- If the envy graph has a source vertex, assign the good to that agent.
- Otherwise, resolve envy cycles until a source vertex shows up, and then assign the good to it.

Does envy-cycle elimination algorithm return an EF1 allocation?

We will argue that each iteration of the algorithm "preserves" EF1.

While there is an unallocated good

- If the envy graph has a source vertex, assign the good to that agent.
- Otherwise, resolve envy cycles until a source vertex shows up, and then assign the good to it.

Does envy-cycle elimination algorithm return an EF1 allocation?

We will argue that each iteration of the algorithm "preserves" EF1.

While there is an unallocated good

- If the envy graph has a source vertex, assign the good to that agent.
- Otherwise, resolve envy cycles until a source vertex shows up, and then assign the good to it.

The source (say, agent s) is not envied by anyone at the start of the iteration.

$$\text{for any agent } i,\ v_i(A_i) \geq v_i(A_s)$$

Does envy-cycle elimination algorithm return an EF1 allocation?

We will argue that each iteration of the algorithm "preserves" EF1.

While there is an unallocated good

- If the envy graph has a source vertex, assign the good to that agent.
- Otherwise, resolve envy cycles until a source vertex shows up, and then assign the good to it.

The source (say, agent s) is not envied by anyone at the start of the iteration.

$$\text{for any agent } i,\ v_i(A_i) \geq v_i(A_s)$$

Suppose good g is assigned to the source agent s. Then,

$$v_i(A_i) \geq v_i(A_s \cup \{g\} \setminus \{g\})$$

which means that EF1 is preserved.

Does envy-cycle elimination algorithm return an EF1 allocation?

We will argue that each iteration of the algorithm "preserves" EF1.

While there is an unallocated good

- If the envy graph has a source vertex, assign the good to that agent.
- Otherwise, resolve envy cycles until a source vertex shows up, and then assign the good to it.

Does envy-cycle elimination algorithm return an EF1 allocation?

We will argue that each iteration of the algorithm "preserves" EF1.

While there is an unallocated good

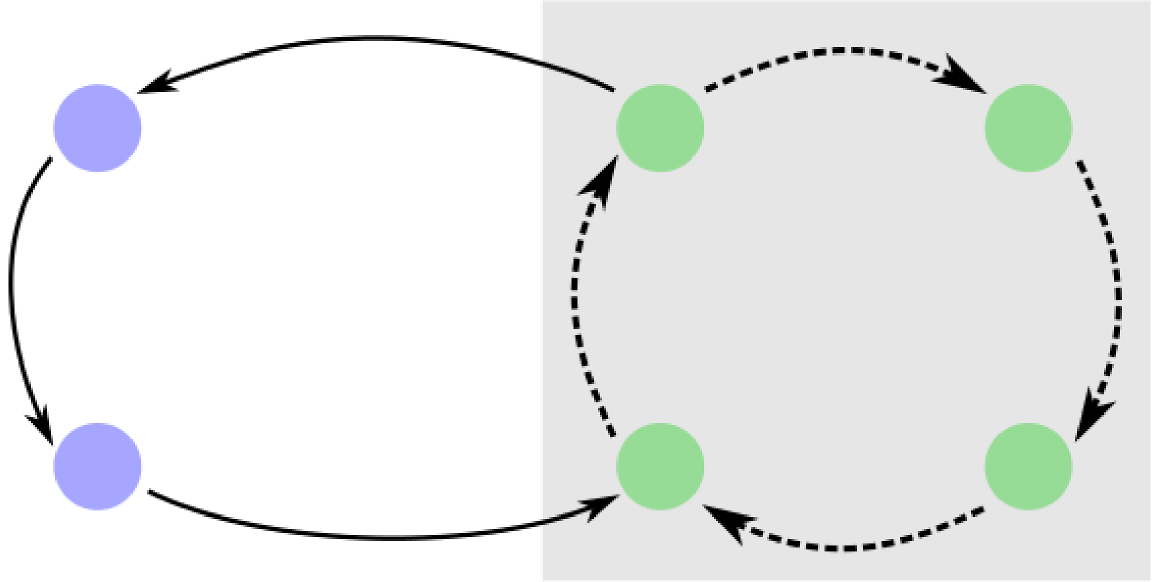
- If the envy graph has a source vertex, assign the good to that agent.
- Otherwise, resolve envy cycles until a source vertex shows up, and then assign the good to it.

Does envy-cycle elimination algorithm return an EF1 allocation?

We will argue that each iteration of the algorithm "preserves" EF1.

While there is an unallocated good

- If the envy graph has a source vertex, assign the good to that agent.
- Otherwise, resolve envy cycles until a source vertex shows up, and then assign the good to it.



Does envy-cycle elimination algorithm return an EF1 allocation?

We will argue that each iteration of the algorithm "preserves" EF1.

While there is an unallocated good

- If the envy graph has a source vertex, assign the good to that agent.
- Otherwise, resolve envy cycles until a source vertex shows up, and then assign the good to it.

Does envy-cycle elimination algorithm return an EF1 allocation?

We will argue that each iteration of the algorithm "preserves" EF1.

While there is an unallocated good

- If the envy graph has a source vertex, assign the good to that agent.
- Otherwise, resolve envy cycles until a source vertex shows up, and then assign the good to it.

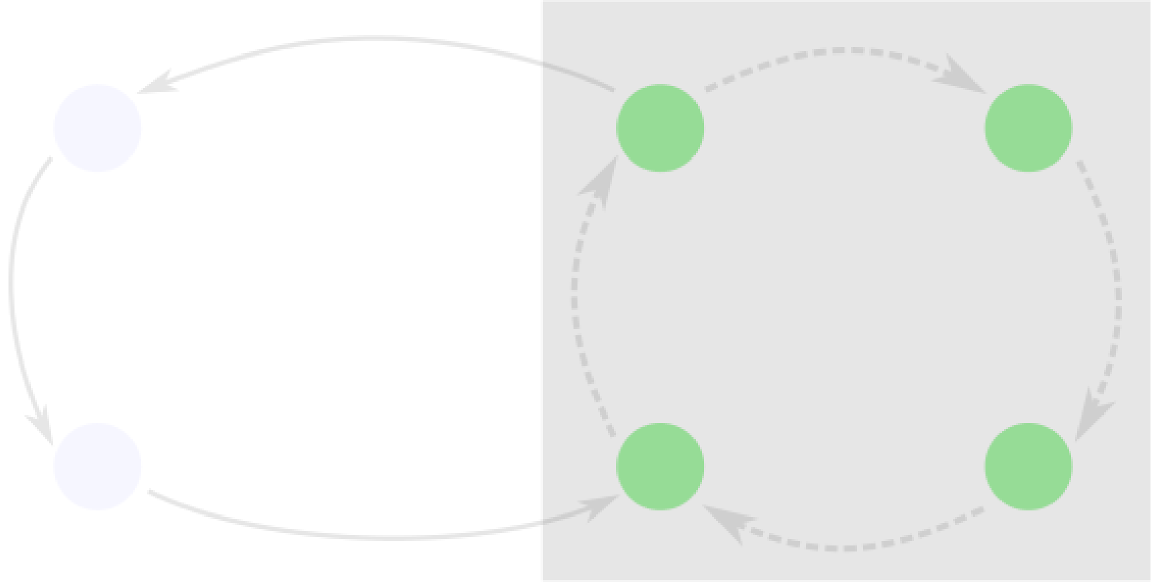
From their perspective, the bundles in the cycle are only shifted around. So, EF1 relations are the same as before.

Does envy-cycle elimination algorithm return an EF1 allocation?

We will argue that each iteration of the algorithm "preserves" EF1.

While there is an unallocated good

- If the envy graph has a source vertex, assign the good to that agent.
- Otherwise, resolve envy cycles until a source vertex shows up, and then assign the good to it.

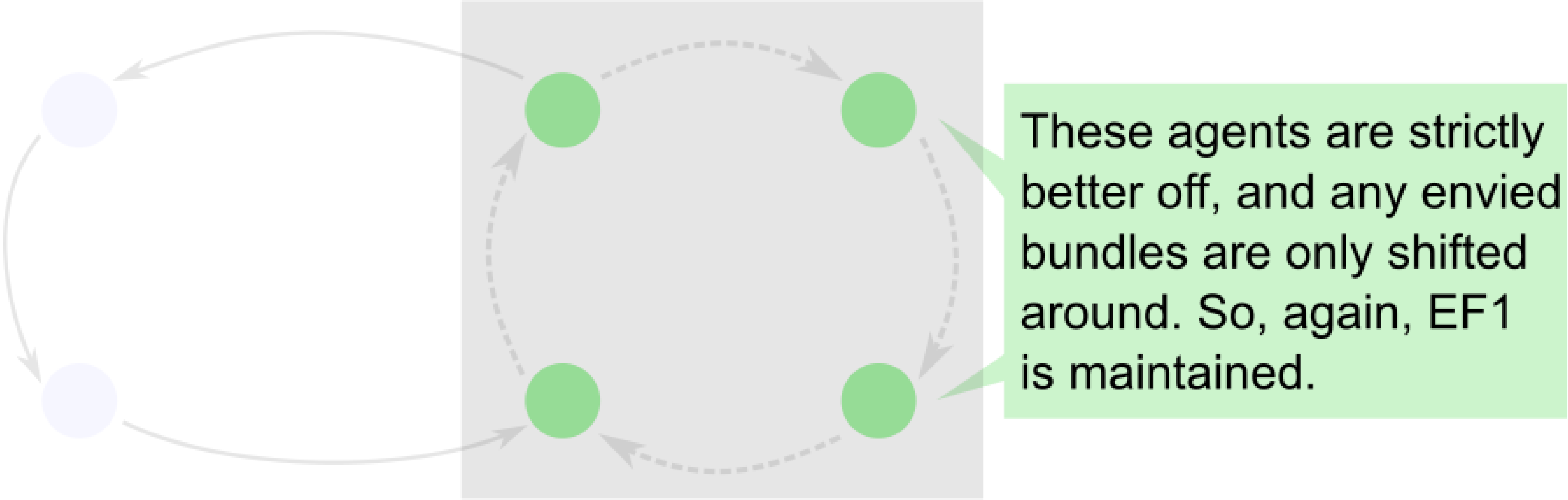


Does envy-cycle elimination algorithm return an EF1 allocation?

We will argue that each iteration of the algorithm "preserves" EF1.

While there is an unallocated good

- If the envy graph has a source vertex, assign the good to that agent.
- Otherwise, resolve envy cycles until a source vertex shows up, and then assign the good to it.



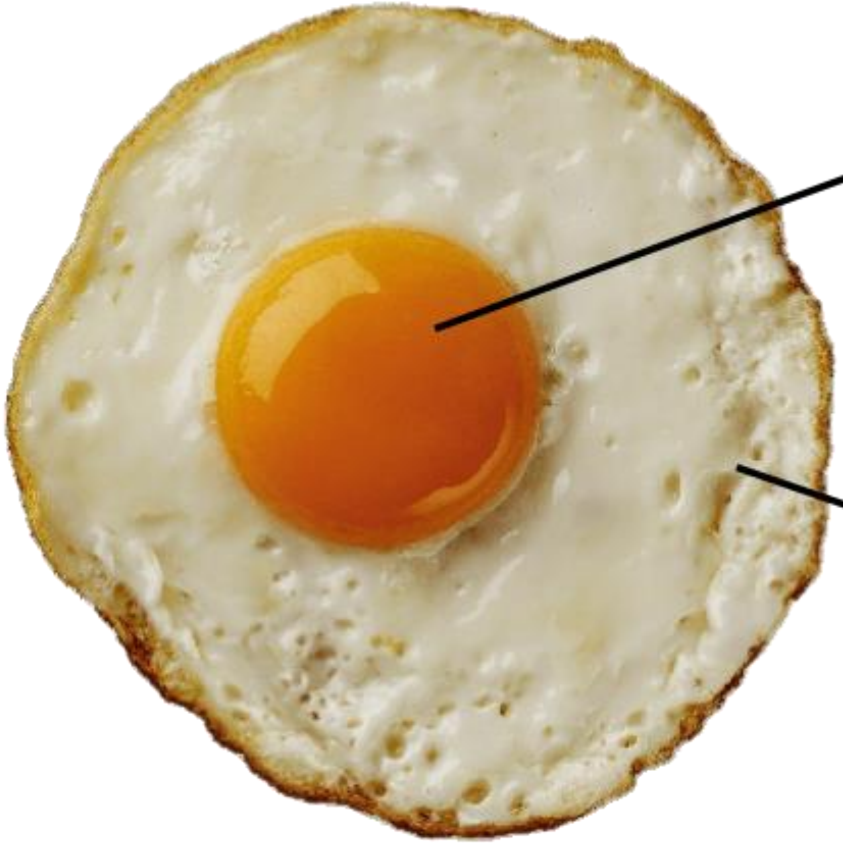
The analysis of envy-cycle elimination algorithm *did not* use additivity.

The analysis of envy-cycle elimination algorithm *did not* use additivity.

For monotone valuations, the allocation computed by the envy-cycle elimination algorithm satisfies EF1.

The analysis of envy-cycle elimination algorithm *did not* use additivity.

For monotone valuations, the allocation computed by the envy-cycle elimination algorithm satisfies EF1.



Additive: $v_i(S) = \sum_{j \in S} v_i(\{j\})$

Monotone: $S \subseteq T \Rightarrow v_i(S) \leq v_i(T)$

Reminders

Project presentations in online mode on Apr 03 (Sun).

Project reports due by Apr 10 (Sun).

Next Time

Fairness and Efficiency

References

- Envy-cycle elimination algorithm

  Richard Lipton, Evangelos Markakis, Elchanan Mossel, and Amin Saberi
  “On Approximately Fair Allocations of Indivisible Goods”
  EC 2004, pg 125-131
  https://dl.acm.org/doi/10.1145/988772.988792