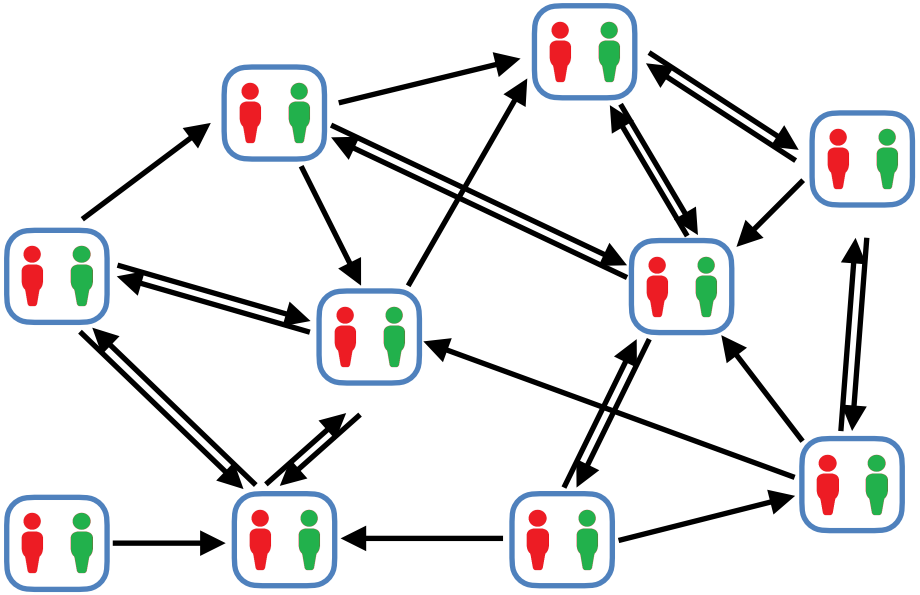
Computational Social Choice

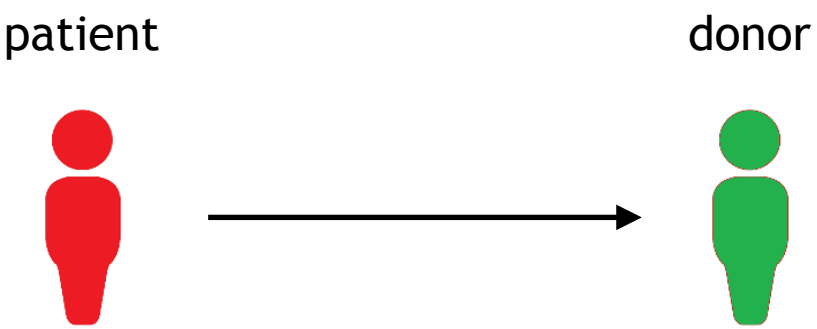
Kidney Exchange

Piotr Skowron

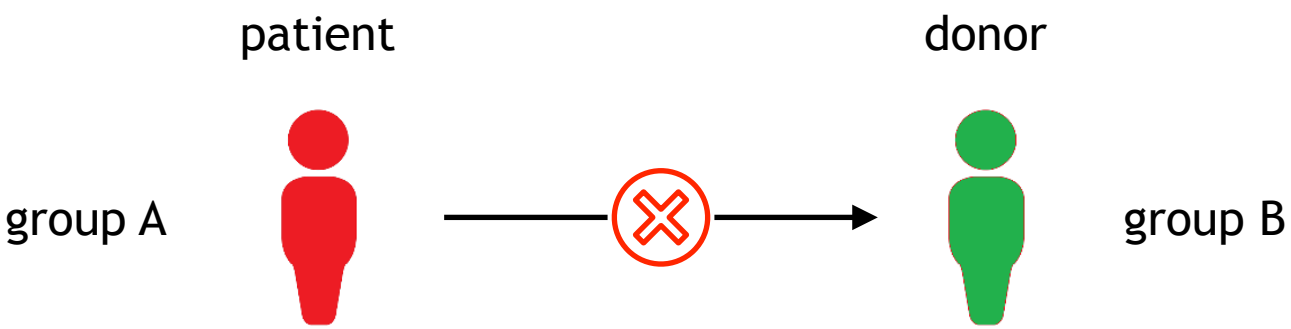
University of Warsaw



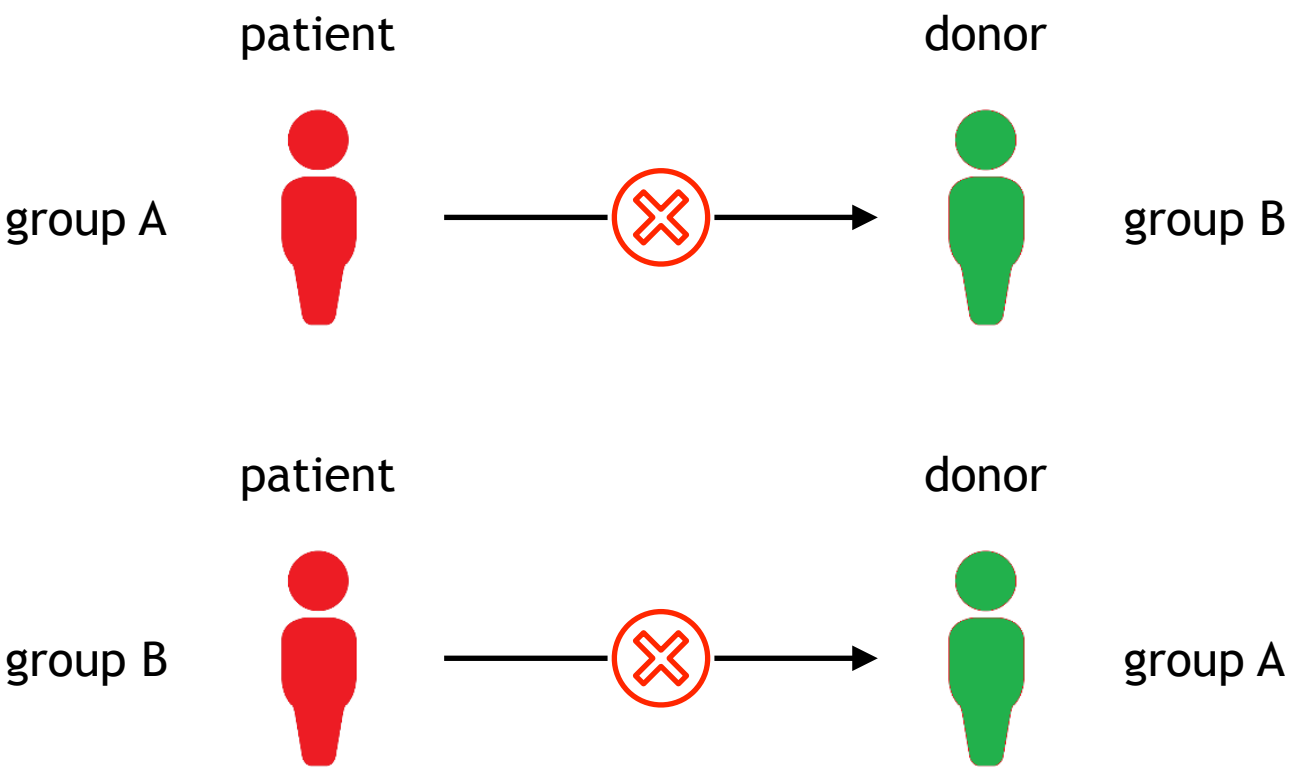
An example



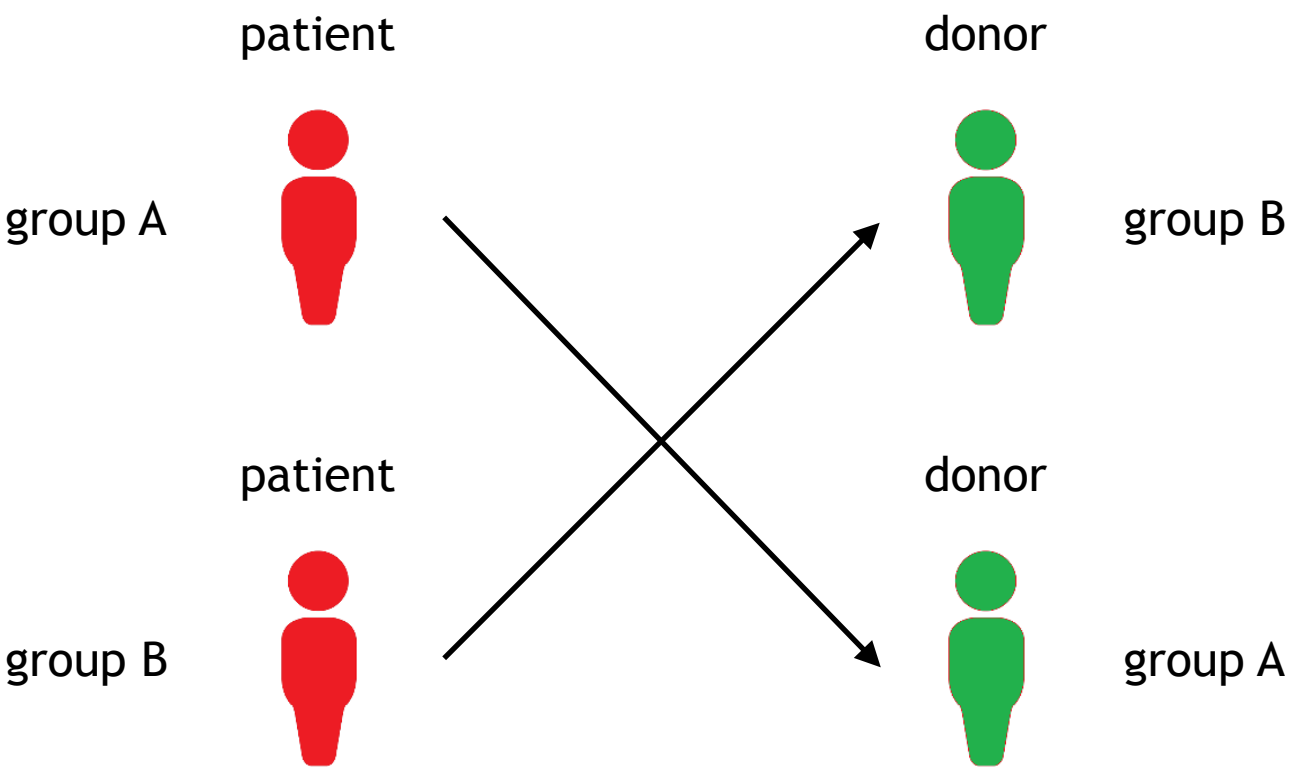
An example



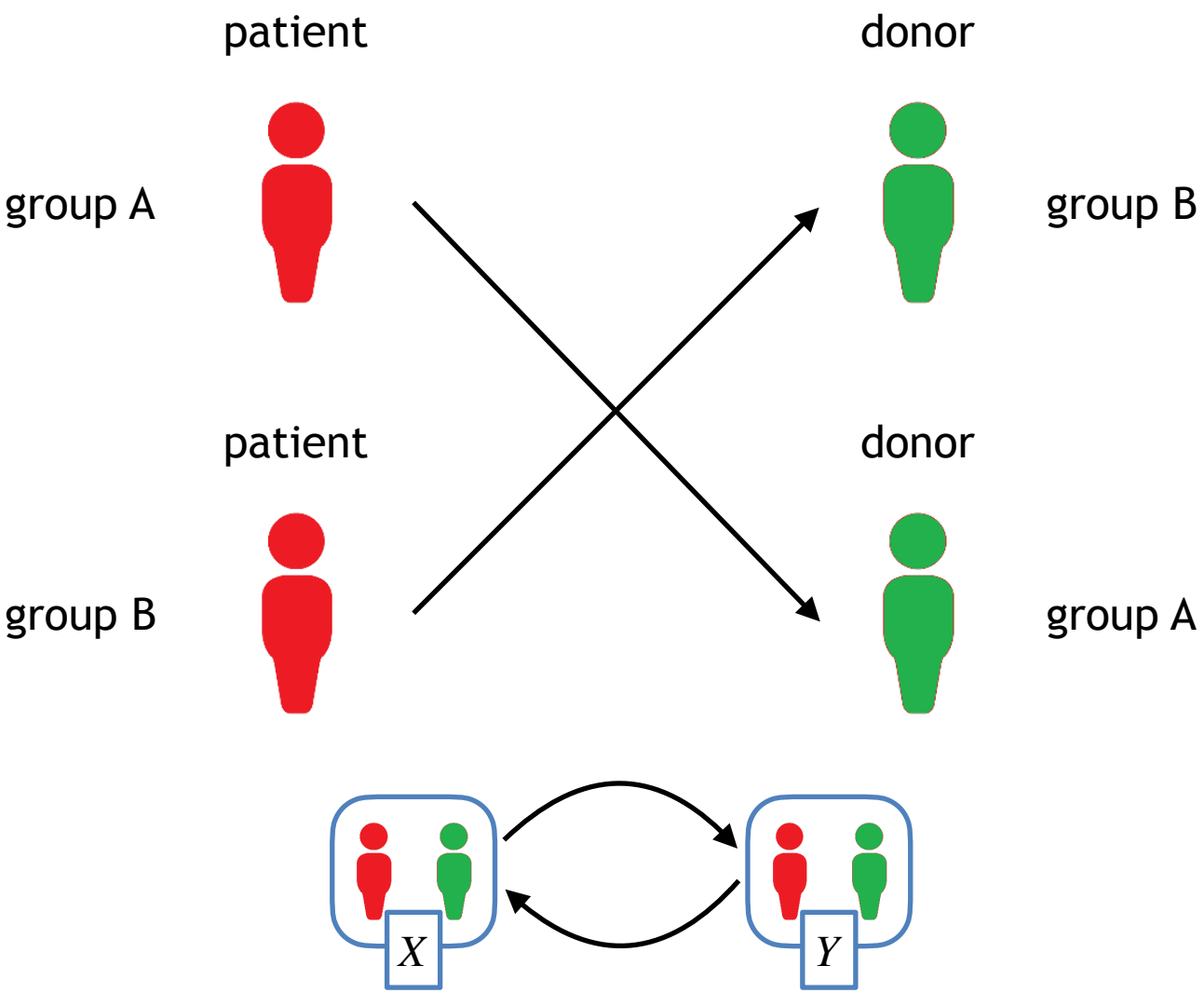
An example



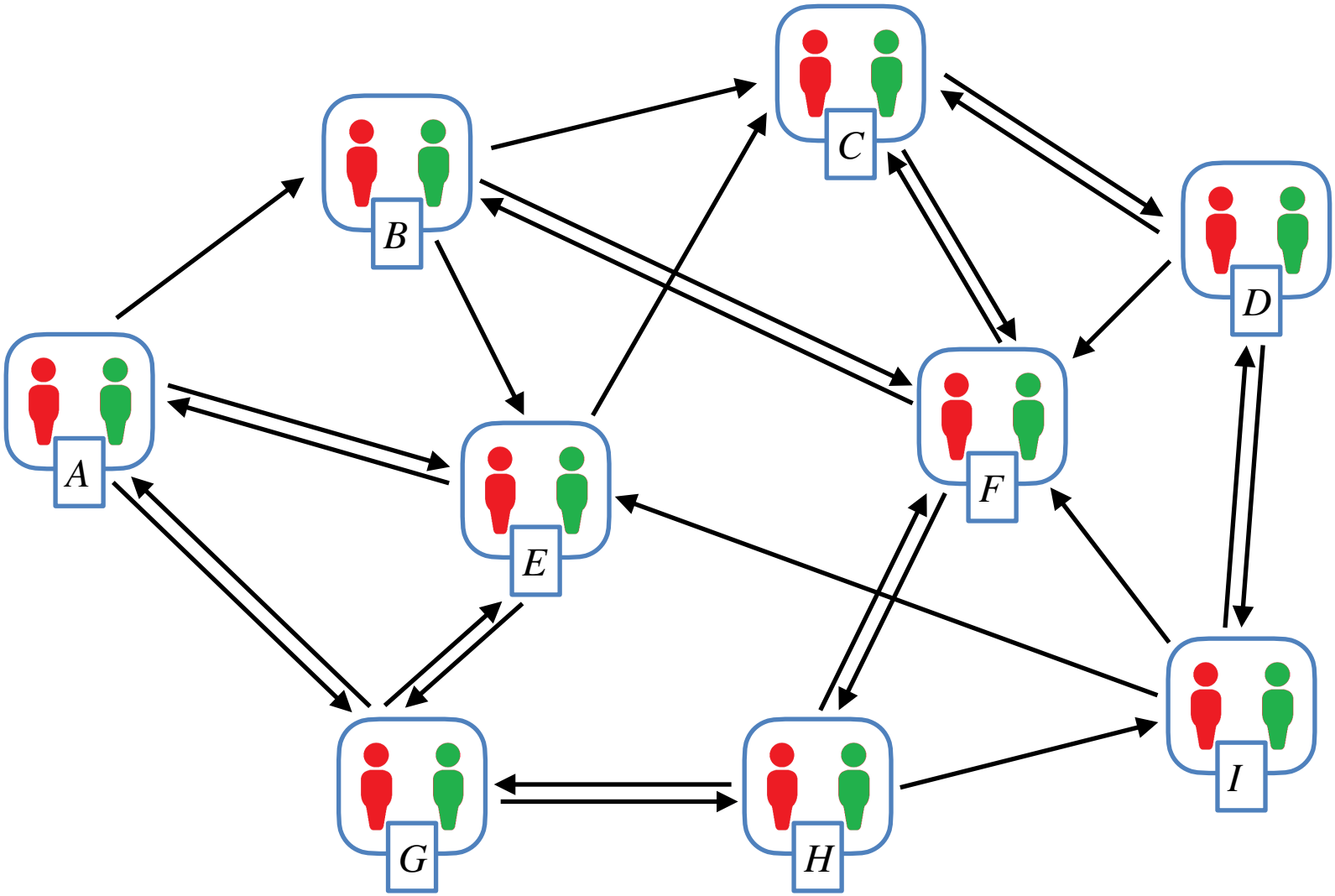
An example



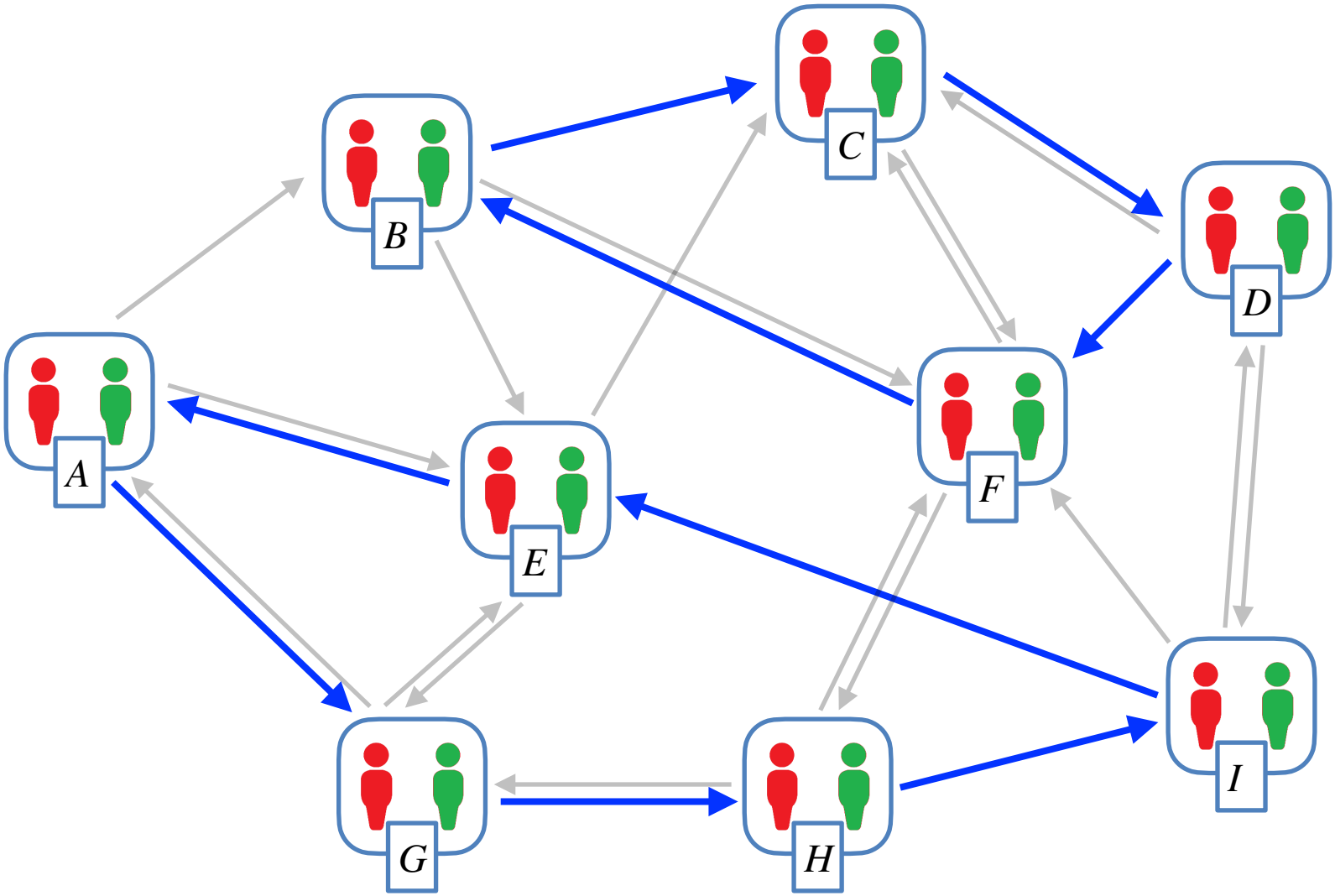
An example


patient
donor
group A
group B
patient
donor
group B
group A
X
Y

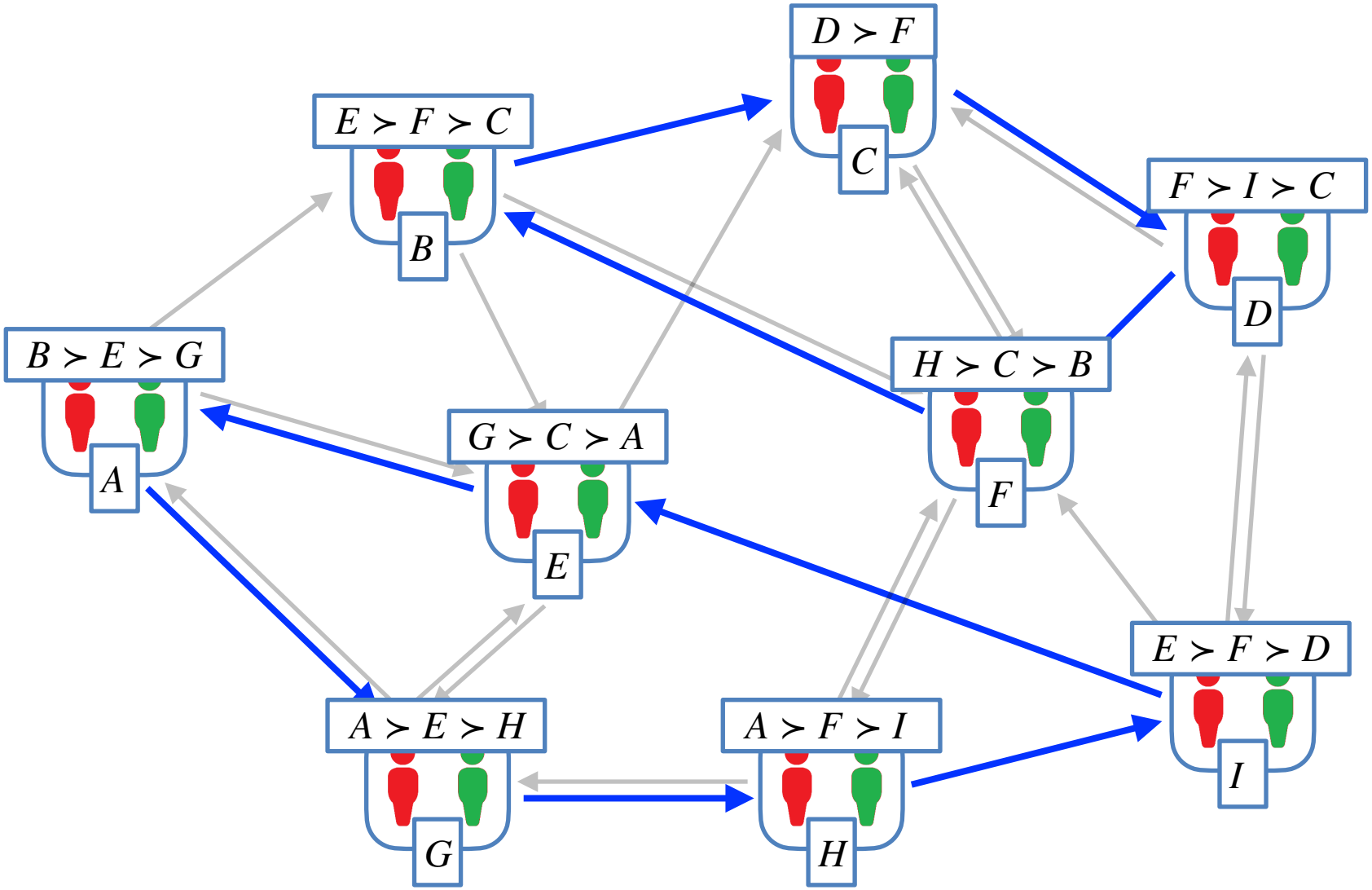
An example



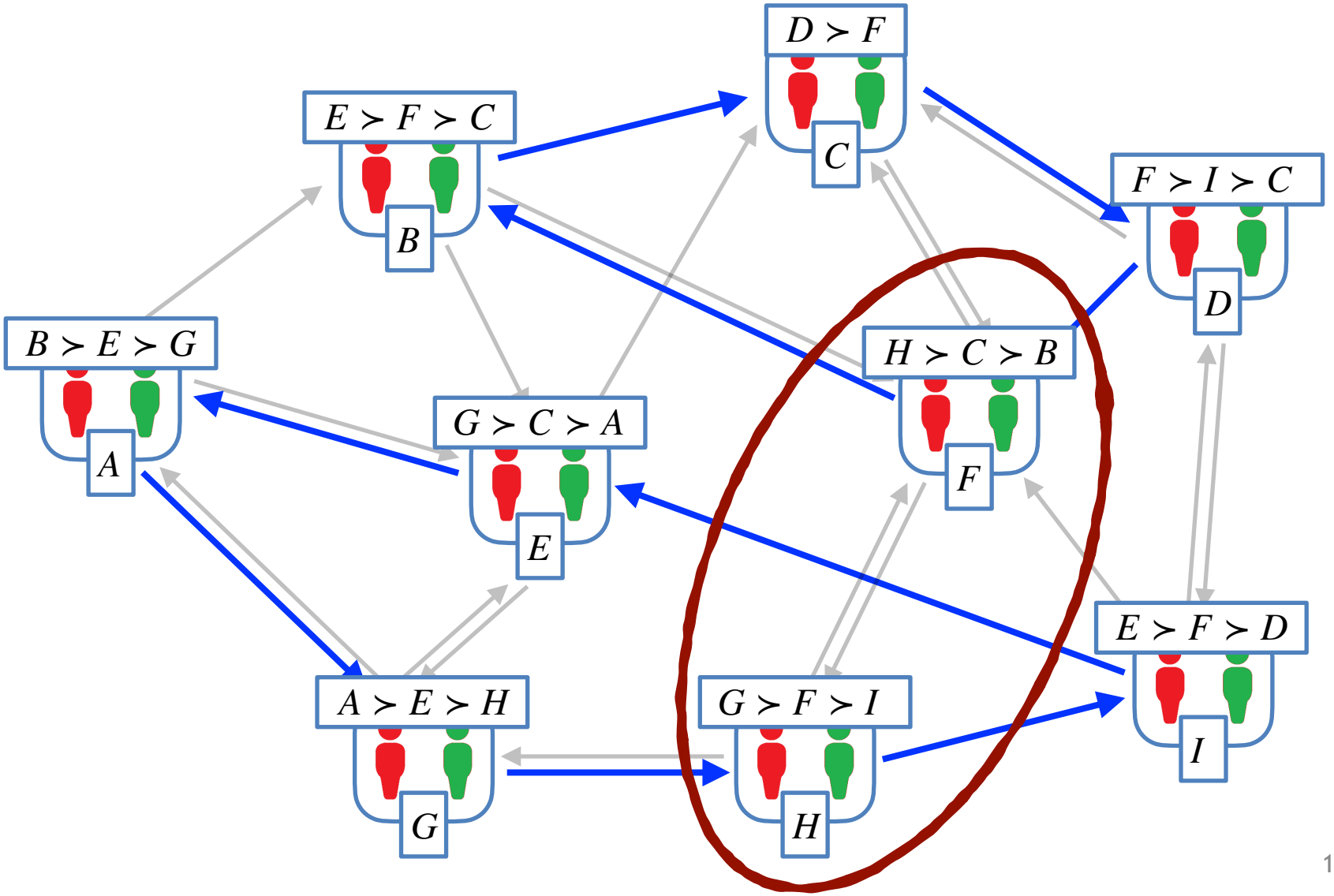
An example



An example


$D \succ F$
C
$E \succ F \succ C$
B
$F \succ I \succ C$
D
$B \succ E \succ G$
A
$H \succ C \succ B$
F
$G \succ C \succ A$
E
$E \succ F \succ D$
I
$A \succ E \succ H$
G
$A \succ F \succ I$
H

An example


$D \succ F$
C
$E \succ F \succ C$
B
$F \succ I \succ C$
D
$B \succ E \succ G$
A
$H \succ C \succ B$
F
$G \succ C \succ A$
E
$E \succ F \succ D$
I
$A \succ E \succ H$
G
$G \succ F \succ I$
H

The model

Input:

A set of pairs (donor, recipient) $C = \{p_1, p_2, \ldots, p_m\}$.
Each pair p comes with a preference relation, $\succ_p$.

Output:

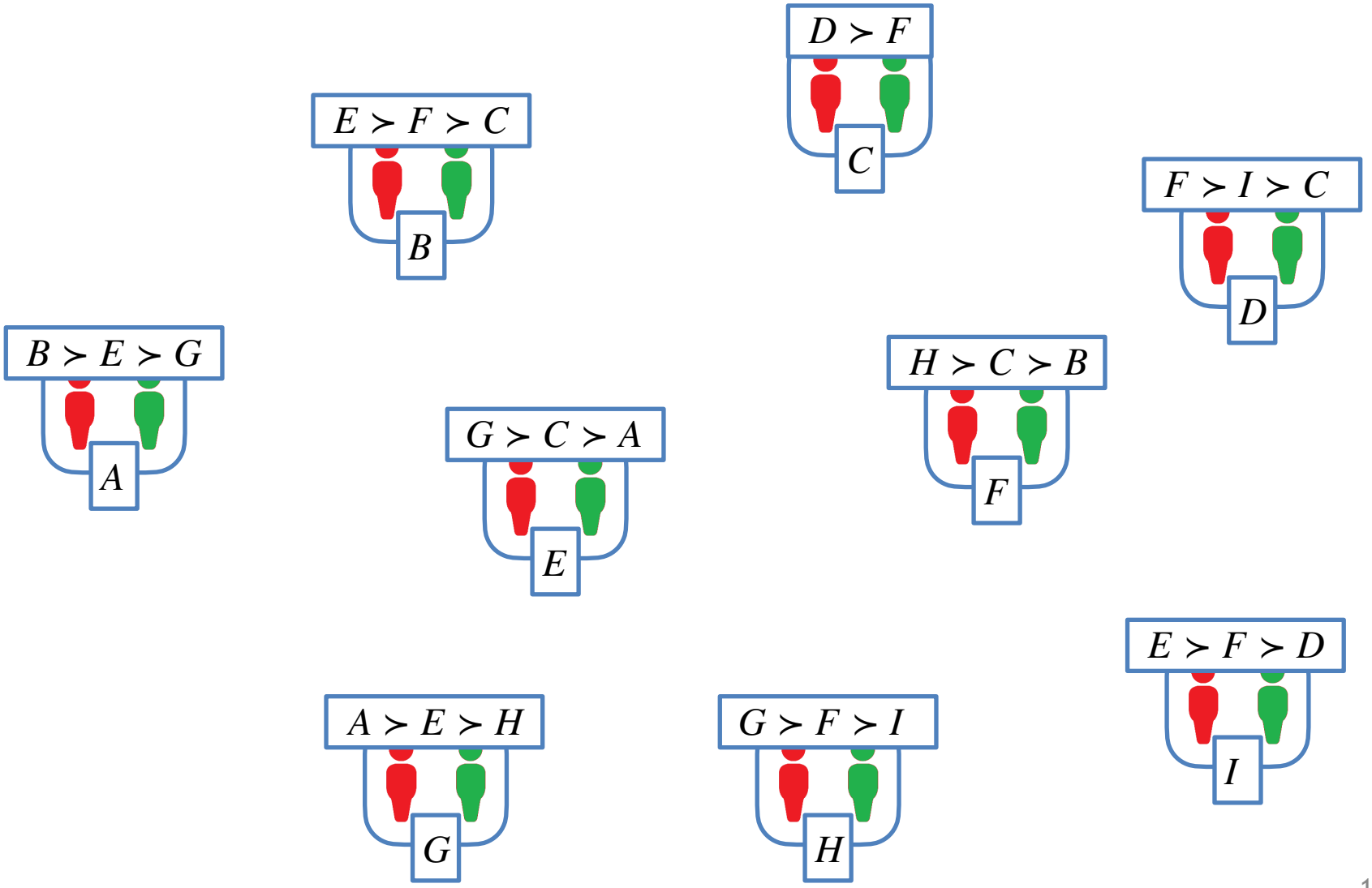
A decomposition into cycles.

The top-trading cycle

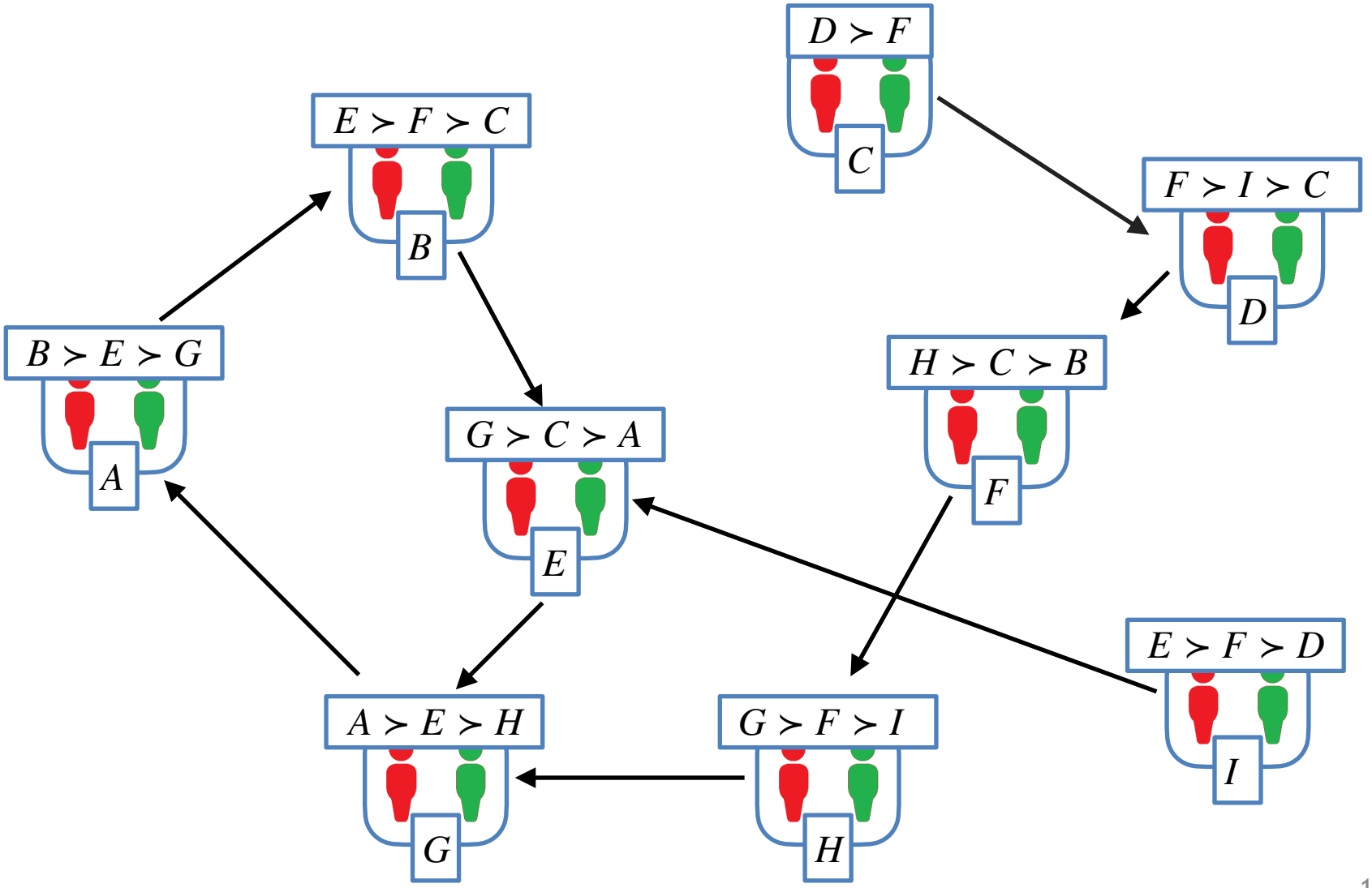
In a loop:

1. Each pair points to her most preferred pair.
2. If there is a cycle, we add it to the solution, and remove the pairs from the cycle.
3. If there is no cycle, we stop.

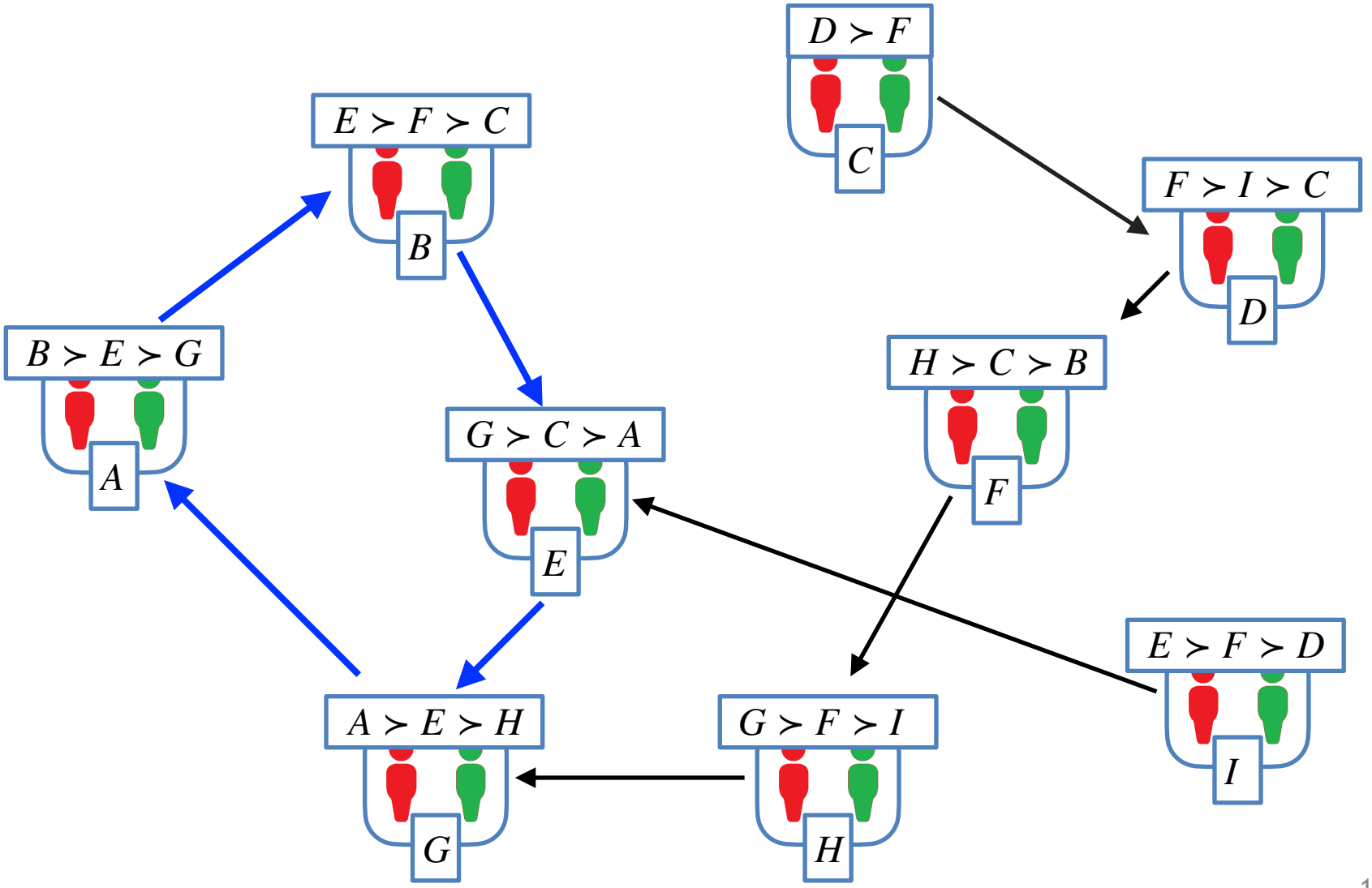
The top-trading cycle



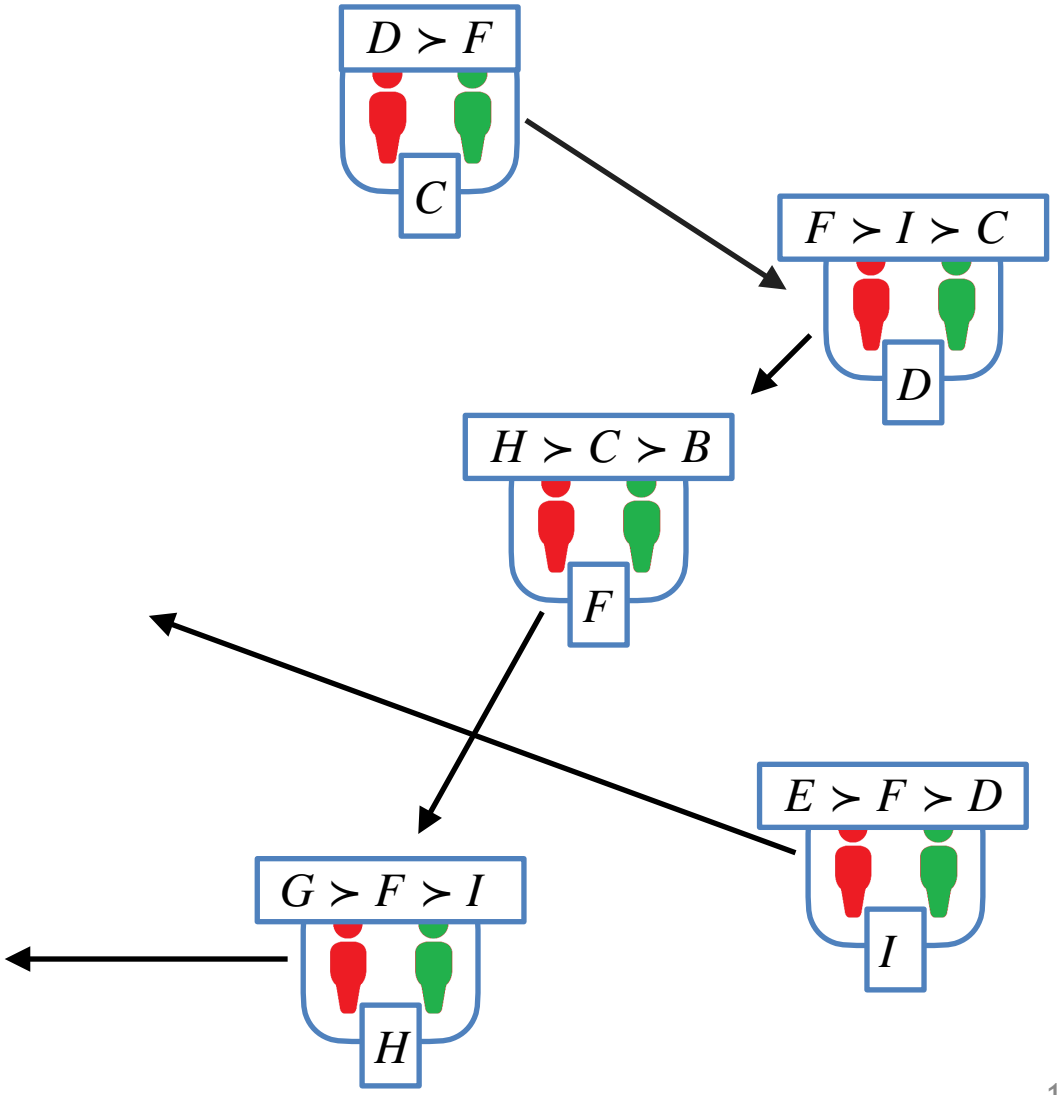
The top-trading cycle



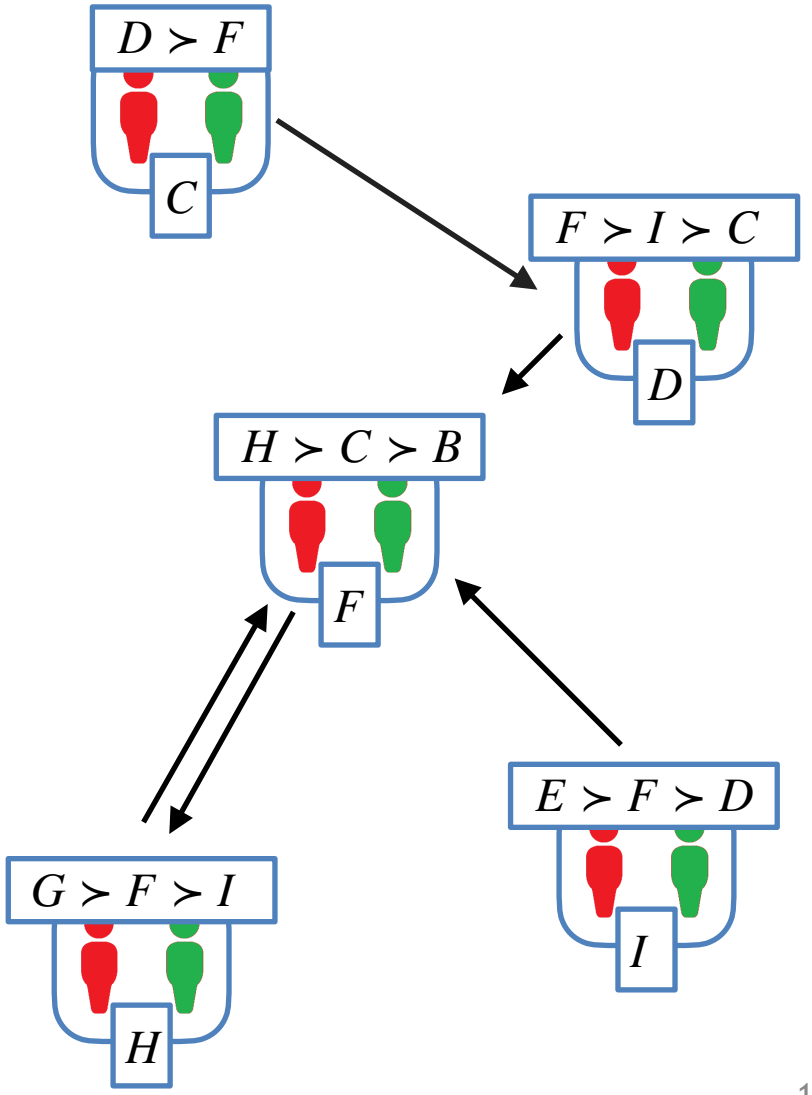
The top-trading cycle



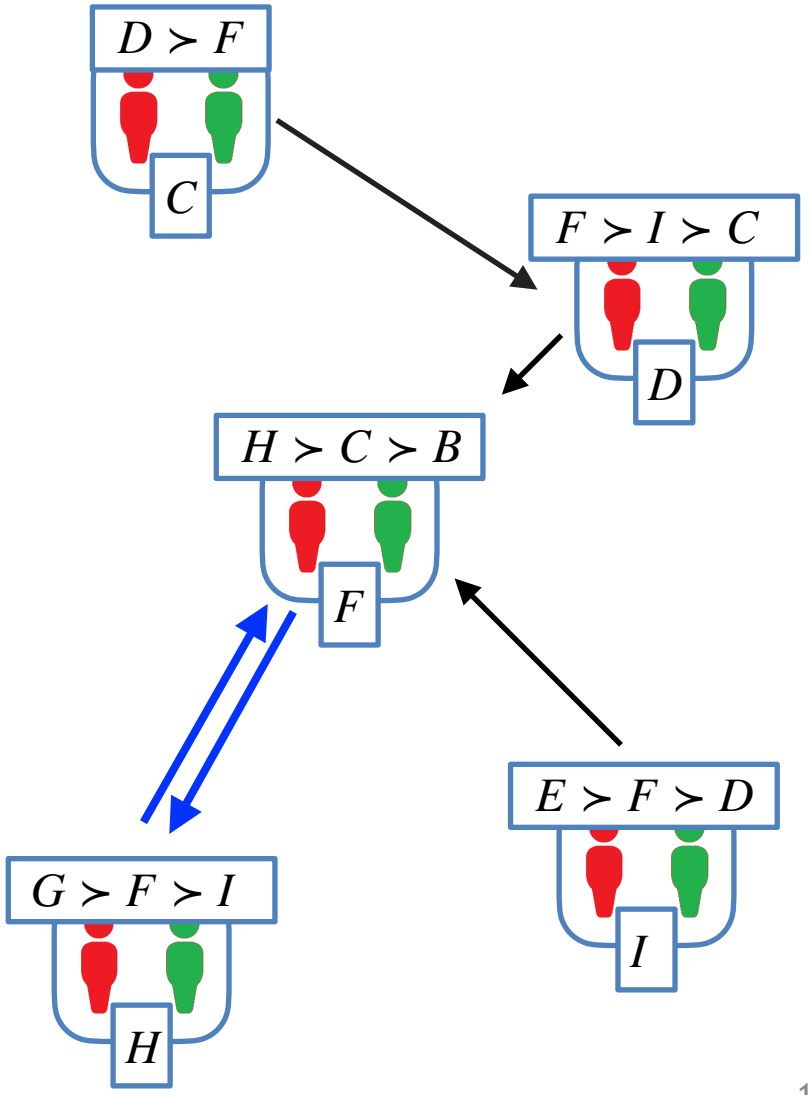
The top-trading cycle



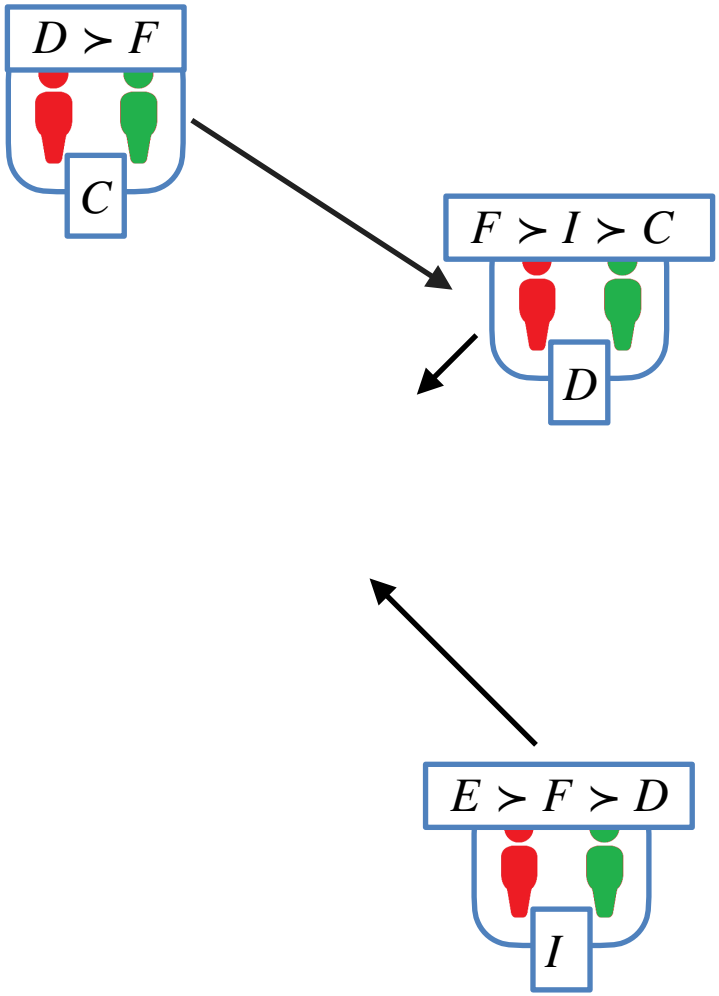
The top-trading cycle



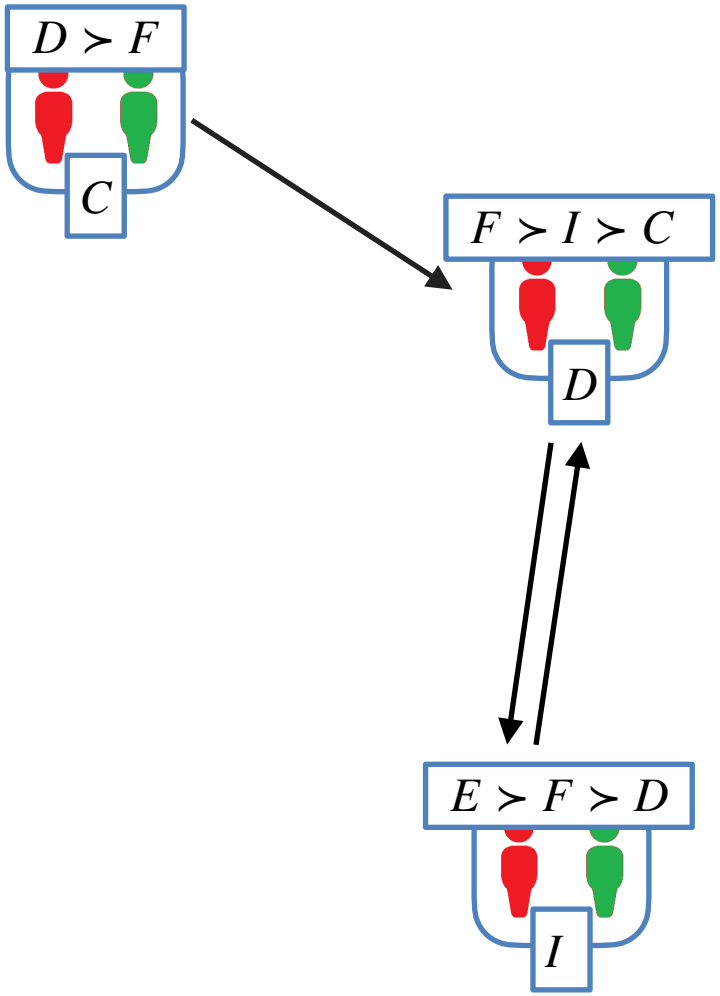
The top-trading cycle



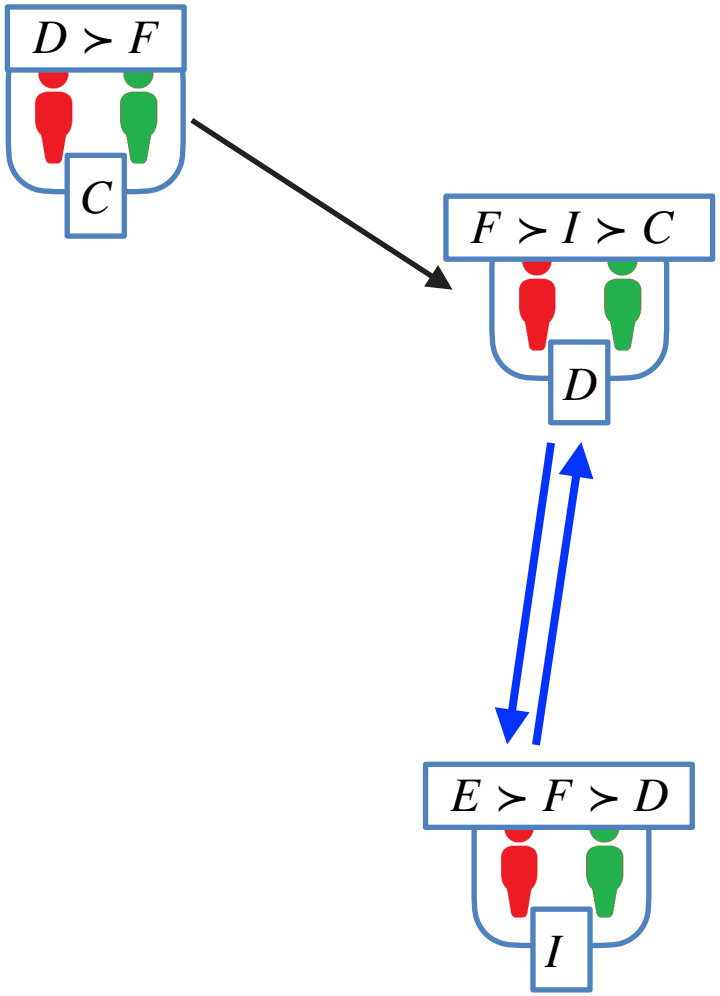
The top-trading cycle



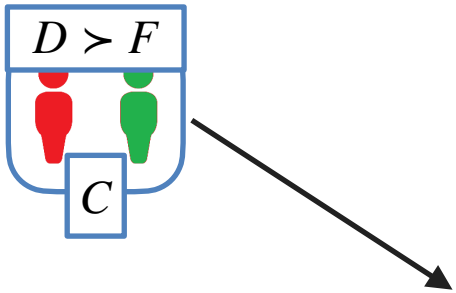
The top-trading cycle



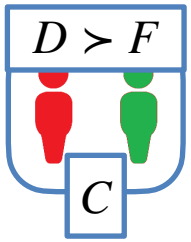
The top-trading cycle



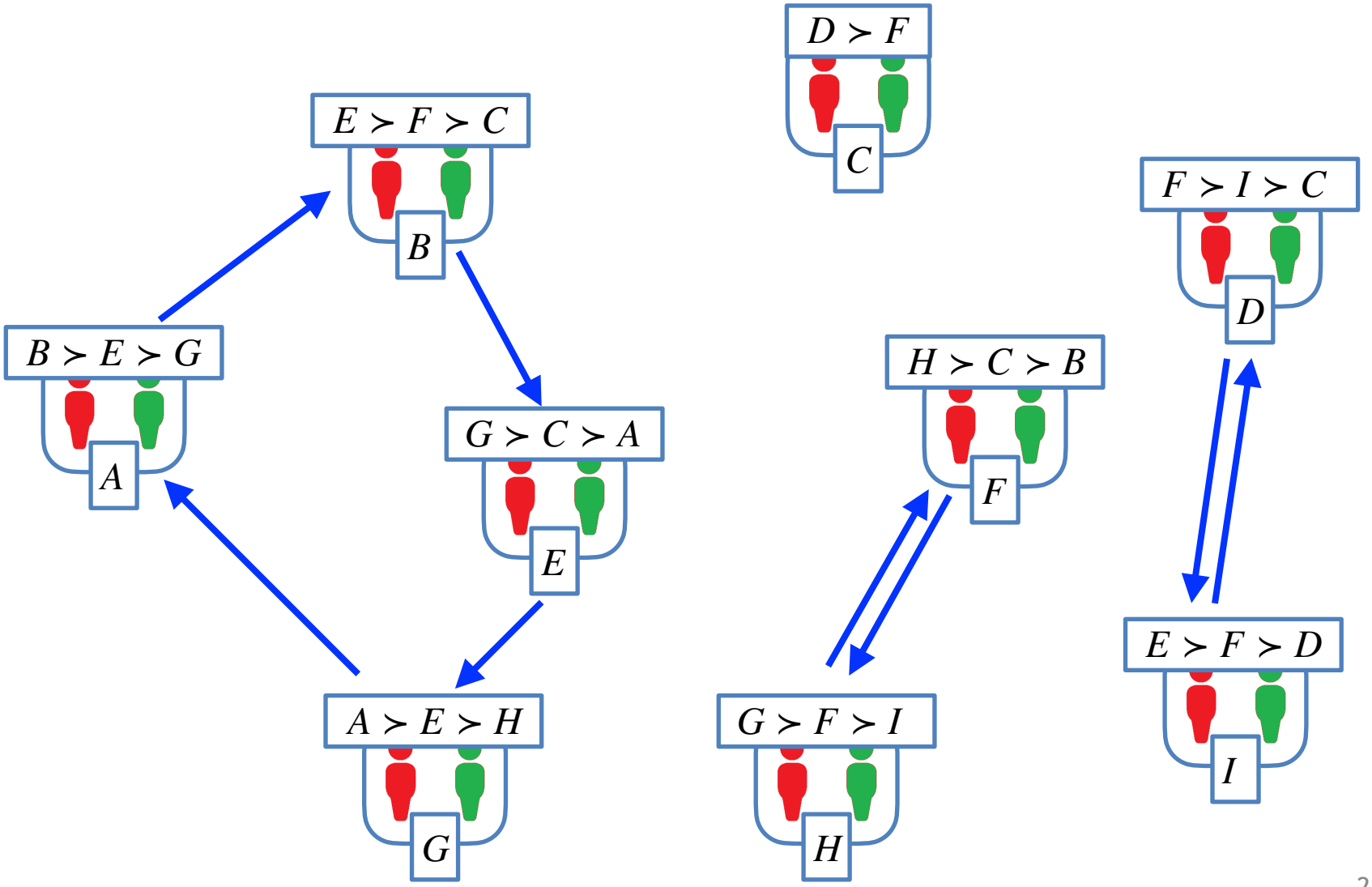
The top-trading cycle



The top-trading cycle



The top-trading cycle



The top-trading cycle

In a loop:

1. Each pair points to her most preferred pair.
2. If there is a cycle, we add it to the solution, and remove the pairs from the cycle.
3. If there is no cycle, we stop.

Theorem:
An optimal strategy for each pair is to reveal their true preference relation.

The top-trading cycle

In a loop:

1. Each pair points to her most preferred pair.
2. If there is a cycle, we add it to the solution, and remove the pairs from the cycle.
3. If there is no cycle, we stop.

Theorem:
An optimal strategy for each pair is to reveal their true preference relation.

Proof:

Consider a pair p that is matched with p'. Assume that p prefers p'' to p'.
We will show that p cannot be matched with p''.

The top-trading cycle

In a loop:
1. Each pair points to her most preferred pair.
2. If there is a cycle, we add it to the solution, and remove the pairs from the cycle.
3. If there is no cycle, we stop.

Theorem:
An optimal strategy for each pair is to reveal their true preference relation.

Proof:

Consider a pair p that is matched with p'. Assume that p prefers p'' to p'. We will show that p cannot be matched with p''.

1. Consider all time moments until p'' was removed; p could not have been removed before, since it was pointing to p'' or someone better, and got p'.

The top-trading cycle

In a loop:
1. Each pair points to her most preferred pair.
2. If there is a cycle, we add it to the solution, and remove the pairs from the cycle.
3. If there is no cycle, we stop.

Theorem:
An optimal strategy for each pair is to reveal their true preference relation.

Proof:

Consider a pair p that is matched with p'. Assume that p prefers p'' to p'. We will show that p cannot be matched with p''.

1. Consider all time moments until p'' was removed; p could not have been removed before, since it was pointing to p'' or someone better, and got p'.
2. Each voter that was removed could not point to p independently of her preferences (since, otherwise, it still would point to p when p was selected).

The top-trading cycle

In a loop:
1. Each pair points to her most preferred pair.
2. If there is a cycle, we add it to the solution, and remove the pairs from the cycle.
3. If there is no cycle, we stop.

Theorem:
An optimal strategy for each pair is to reveal their true preference relation.

Proof:

Consider a pair p that is matched with p'. Assume that p prefers p'' to p'. We will show that p cannot be matched with p''.

1. Consider all time moments until p'' was removed; p could not have been removed before, since it was pointing to p'' or someone better, and got p'.
2. Each voter that was removed could not point to p independently of her preferences (since, otherwise, it still would point to p when p was selected).
3. In such time moments, no voter would point to p, and thus, they would be removed (including p'') independently of what p reports.

The top-trading cycle

In a loop:

1. Each pair points to her most preferred pair.
2. If there is a cycle, we add it to the solution, and remove the pairs from the cycle.
3. If there is no cycle, we stop.

Theorem:
An optimal strategy for each pair is to reveal their true preference relation.

Definition:
A solution is in the ***core***, if there exists no group of pairs S that could perform trading on their own in a way that each member of S would get at least as good matched partner as in the solution, and at least one pair from S would get a strictly better one.

The top-trading cycle

In a loop:

1. Each pair points to her most preferred pair.
2. If there is a cycle, we add it to the solution, and remove the pairs from the cycle.
3. If there is no cycle, we stop.

Theorem:
An optimal strategy for each pair is to reveal their true preference relation.

Definition:
A solution is in the ***core***, if there exists no group of pairs S that could perform trading on their own in a way that each member of S would get at least as good matched partner as in the solution, and at least one pair from S would get a strictly better one.

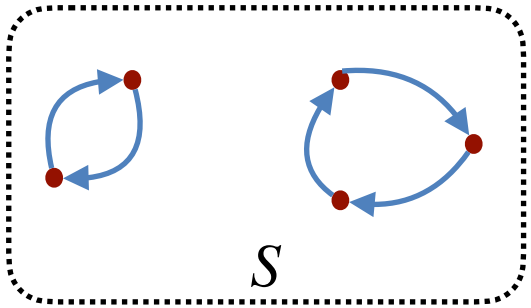
Theorem:
Solutions returned by the top-trading cycle are in the core.

The top-trading cycle

Theorem:
Solutions returned by the top-trading cycle are in the core.

Proof sketch:

Consider a minimal set S that witnesses the violation of the core.

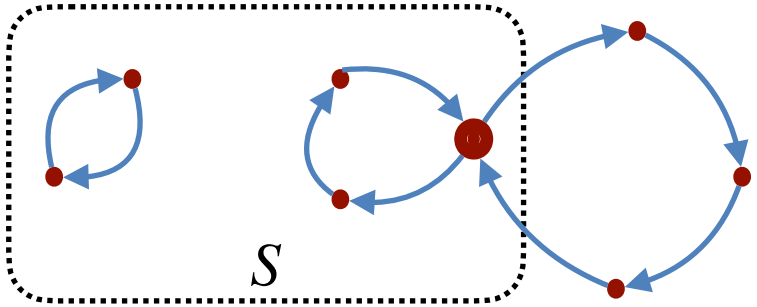


The top-trading cycle

Theorem:
Solutions returned by the top-trading cycle are in the core.

Proof sketch:

Consider a minimal set S that witnesses the violation of the core.



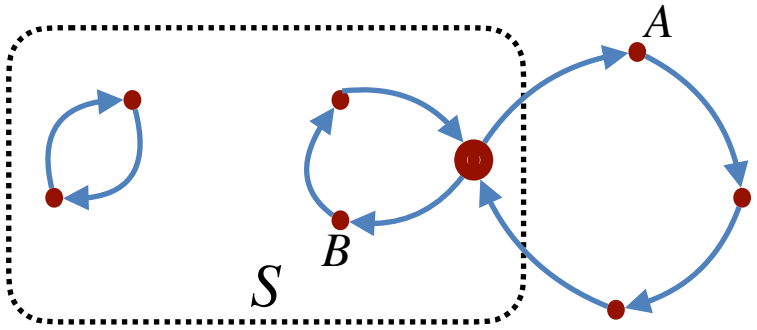
Take the voter who is in the first cycle.

The top-trading cycle

Theorem:
Solutions returned by the top-trading cycle are in the core.

Proof sketch:

Consider a minimal set S that witnesses the violation of the core.



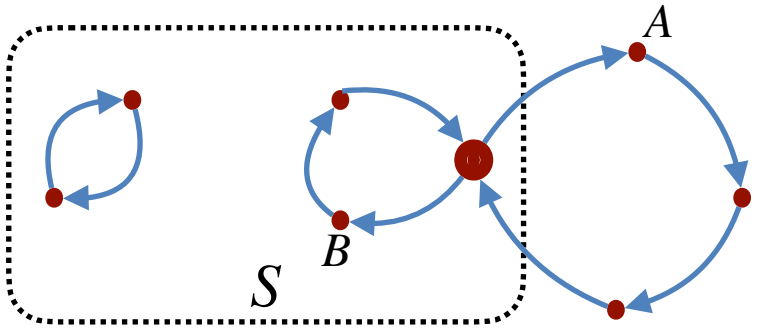
Take the voter who is in the first cycle. Since she prefers B to A, B must have been eliminated before A (otherwise she would point to B).

The top-trading cycle

Theorem:
Solutions returned by the top-trading cycle are in the core.

Proof sketch:

Consider a minimal set S that witnesses the violation of the core.



Take the voter who is in the first cycle. Since she prefers B to A, B must have been eliminated before A (otherwise she would point to B).

This is a contradiction since A was selected at the same time as the voter that we consider.

Practice vs Theory

Problem:
Clearing a cycle of size ℓ requires 2ℓ operating theatres and 2ℓ surgical teams available at the same time.

Practice vs Theory

Problem:
Clearing a cycle of size ℓ requires 2ℓ operating theatres and 2ℓ surgical teams available at the same time.

Solution:
Use only small cycles.

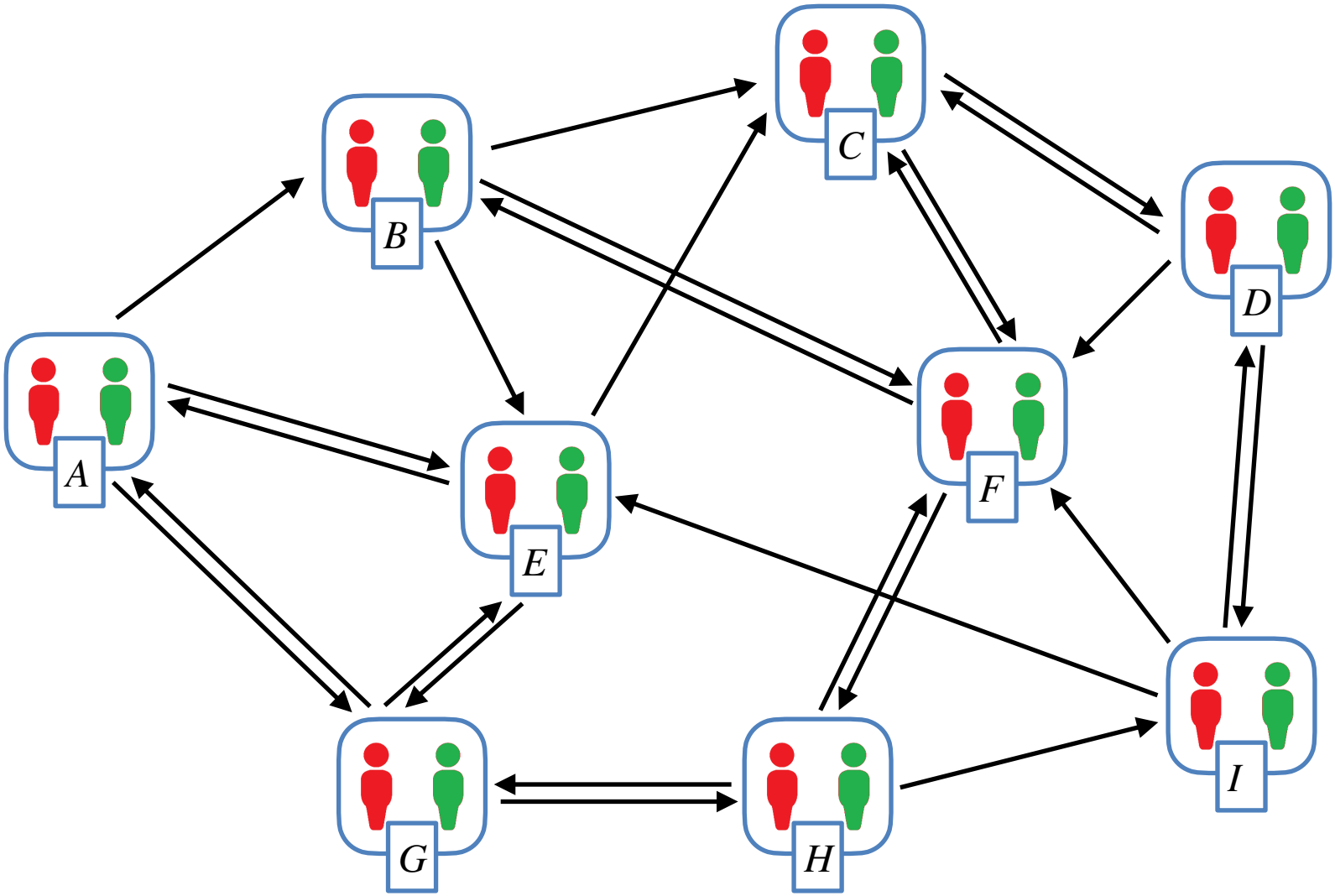
Practice vs Theory

Problem:
Clearing a cycle of size ℓ requires 2ℓ operating theatres and 2ℓ surgical teams available at the same time.

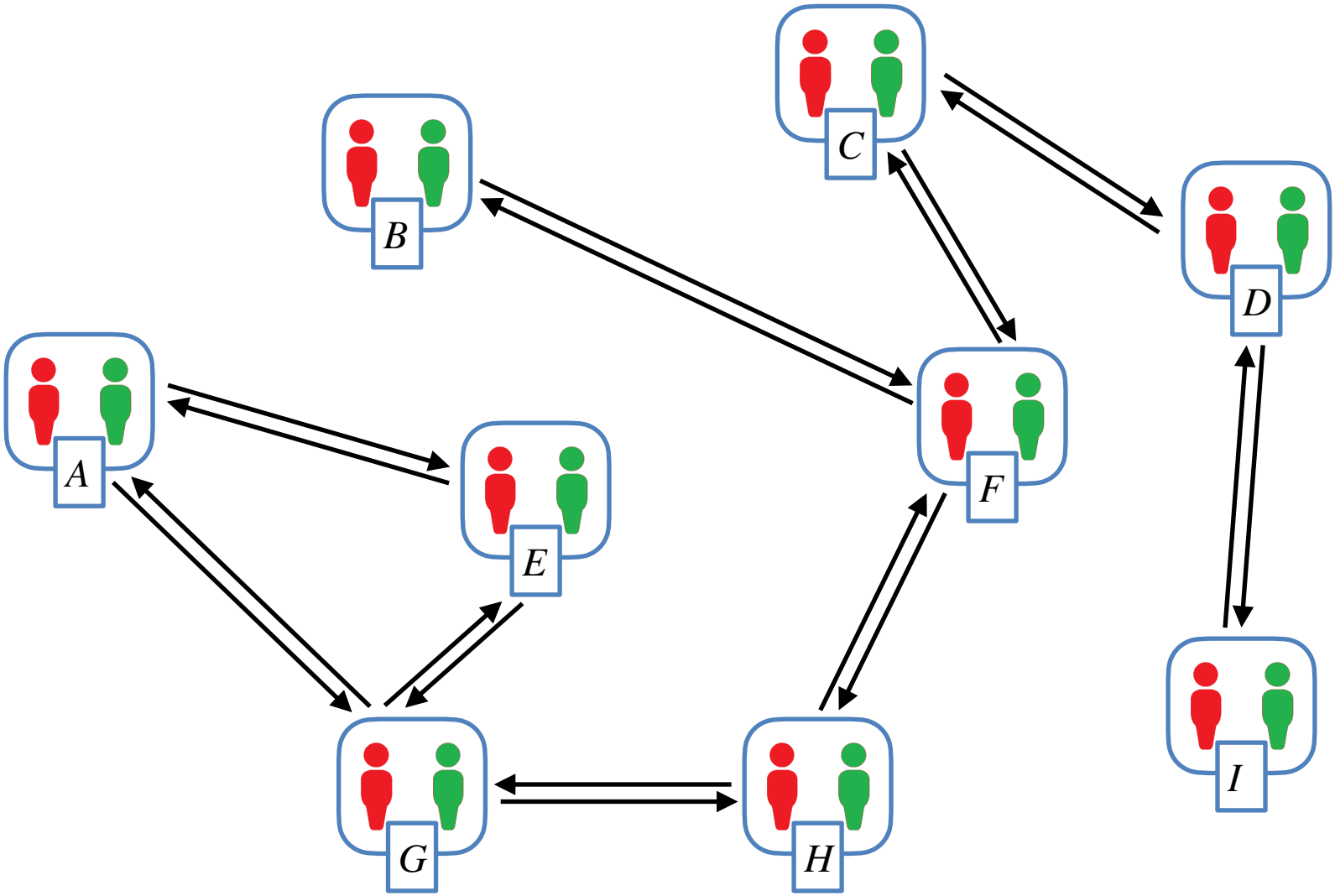
Solution:
Use only small cycles.

Matching Theory!

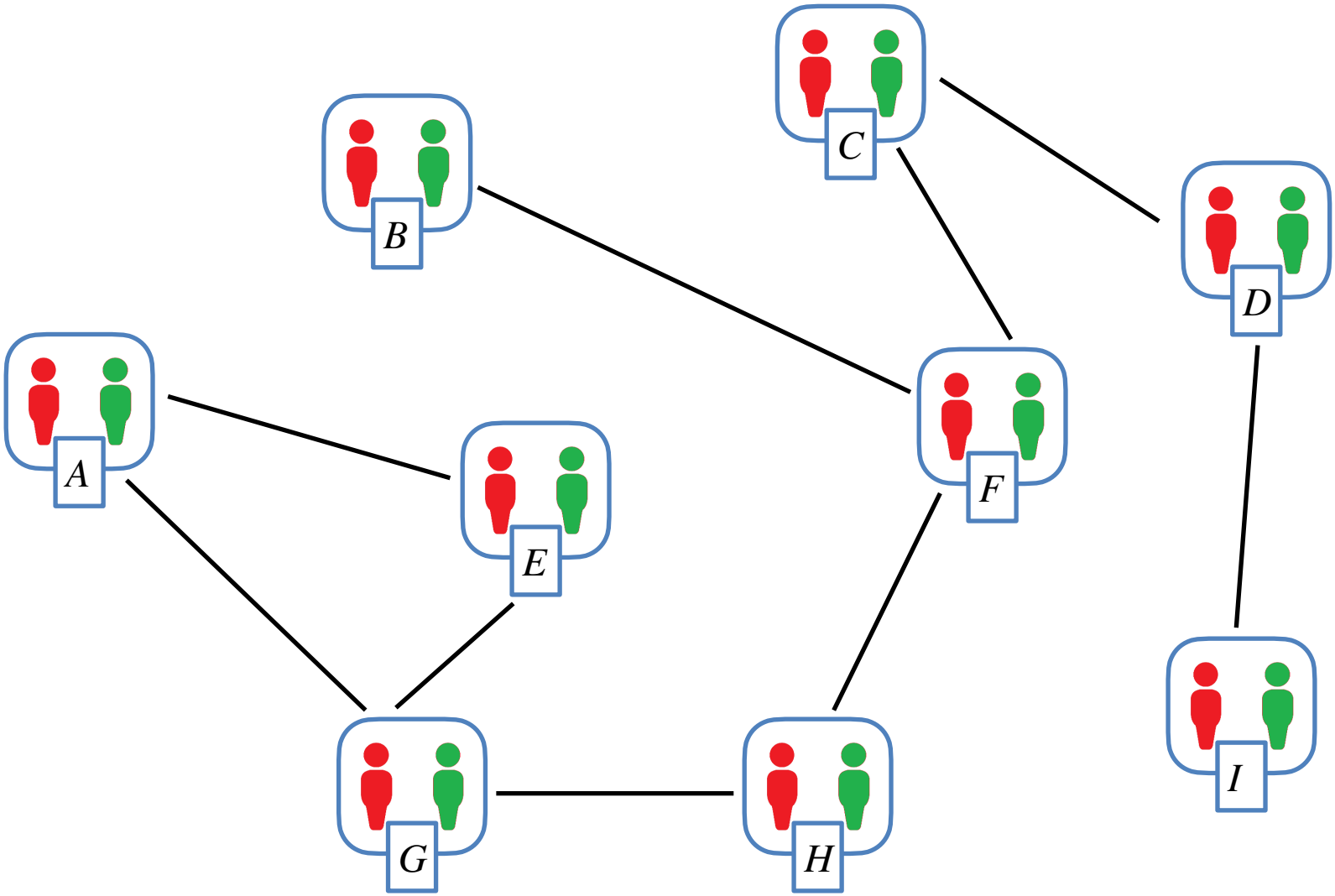
Back to the example



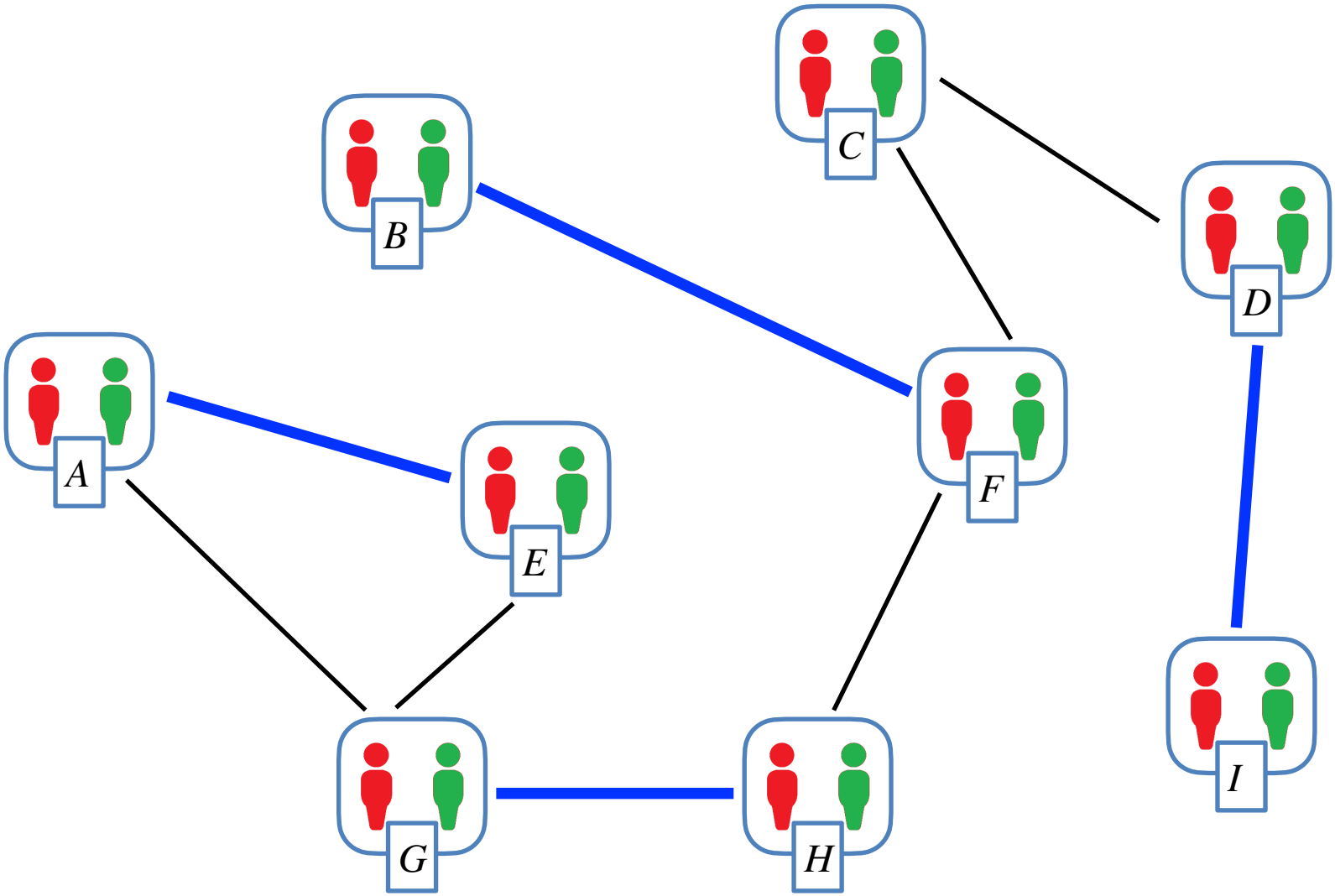
Back to the example



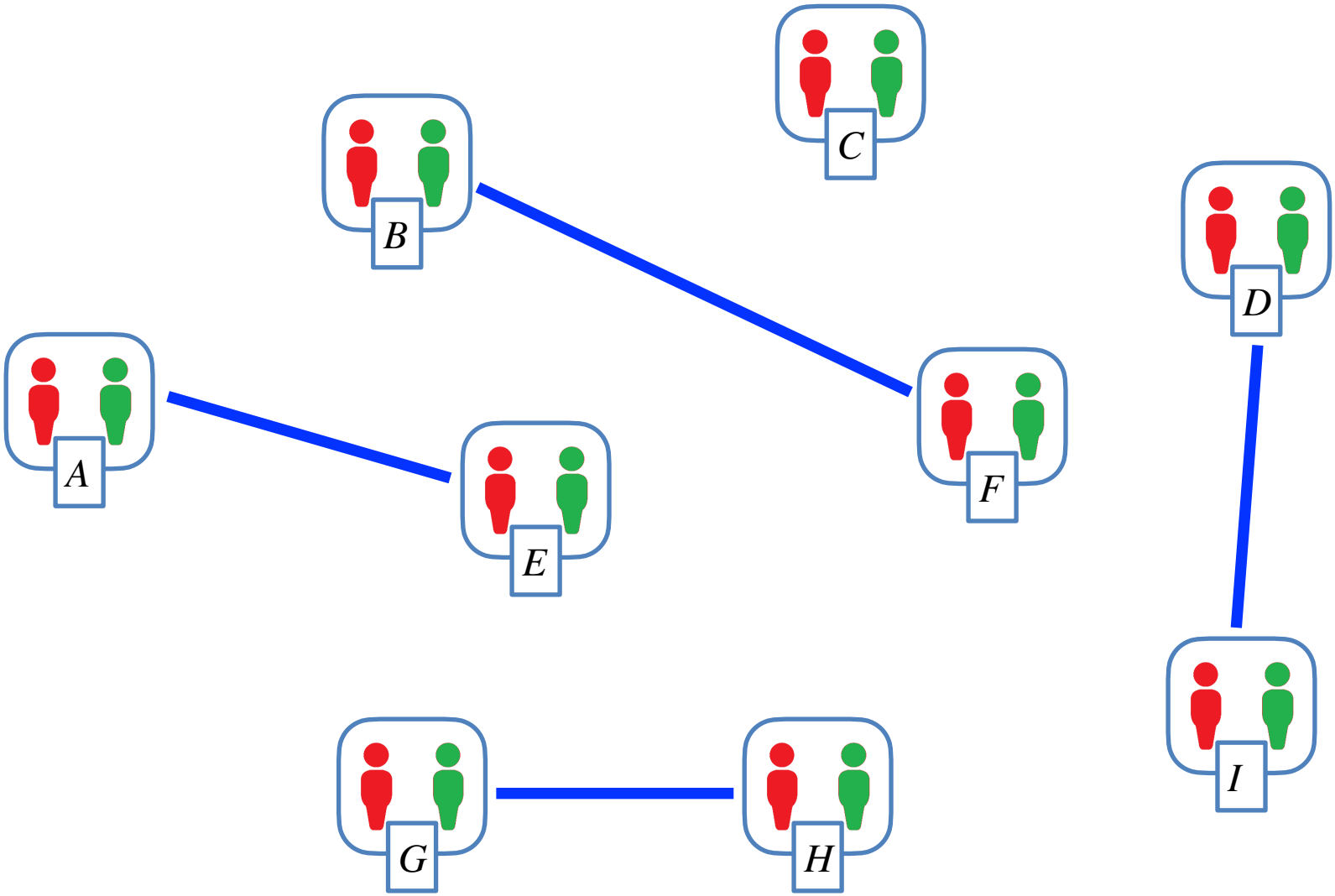
Back to the example



Back to the example



Back to the example



Practice vs Theory

Problem:
Clearing a cycle of size ℓ requires 2ℓ operating theatres and 2ℓ surgical teams available at the same time.

Solution:
Use only small cycles.

Matching Theory!

Properties:
Maximum-cardinality matchings can be found in polynomial time. How to make it incentive compatible? (Using **max-weight-matching.**)

Blossom Algorithm

Find matching with a minimal cost.

We assume perfect matching exists!

What if not?

Blossom Algorithm

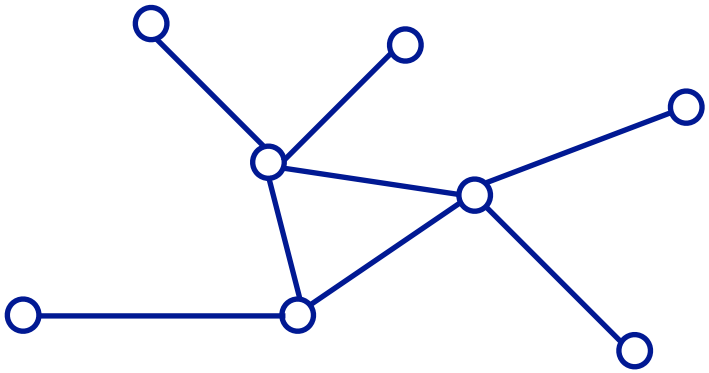
For a subset S let $\delta(S)$ denote the boundary of S that is:

$$\delta(S) = \{(u, v) \in E \colon U \in S, v \notin S\}$$

Blossom Algorithm

For a subset S let $\delta(S)$ denote the boundary of S that is:

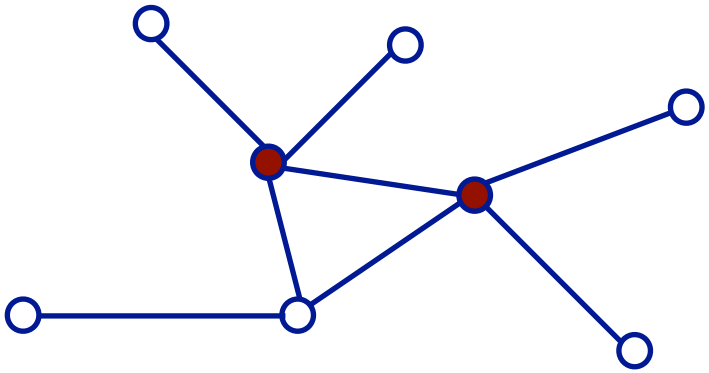
$$\delta(S) = \{(u, v) \in E\colon U \in S, v \notin S\}$$



Blossom Algorithm

For a subset S let $\delta(S)$ denote the boundary of S that is:

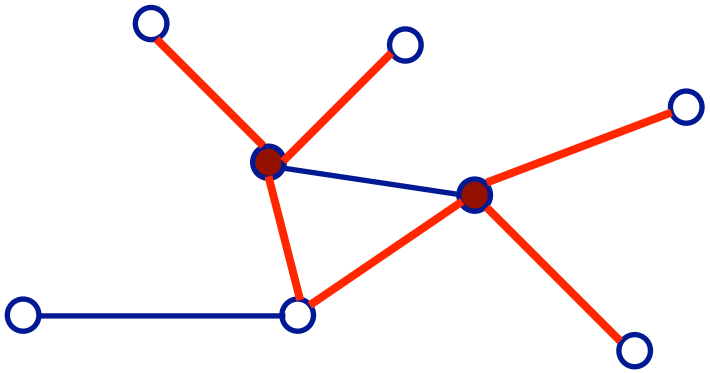
$$\delta(S) = \{(u, v) \in E\colon U \in S, v \notin S\}$$



Blossom Algorithm

For a subset S let $\delta(S)$ denote the boundary of S that is:

$$\delta(S) = \{(u, v) \in E\colon U \in S, v \notin S\}$$



Blossom Algorithm

For a subset S let $\delta(S)$ denote the boundary of S that is:

$$\delta(S) = \{(u, v) \in E\colon U \in S, v \notin S\}$$

O: all subsets of V of odd cardinality ≥ 3

Blossom Algorithm

For a subset S let $\delta(S)$ denote the boundary of S that is:

$$\delta(S) = \{(u, v) \in E\colon U \in S, v \notin S\}$$

O: all subsets of V of odd cardinality ≥ 3

Primal program

Minimize: $\sum_{e \in E} c_e \cdot x_e$

cost of an edge

Binary variable
(1 if edge is selected and 0 otherwise)

Blossom Algorithm

For a subset S let $\delta(S)$ denote the boundary of S that is:

$$\delta(S) = \{(u, v) \in E\colon\ U \in S, v \notin S\}$$

O: all subsets of V of odd cardinality ≥ 3

Primal program

Minimize: $\sum_{e \in E} c_e \cdot x_e$

subject to: $\sum_{e \in \delta(v)} x_e = 1 \qquad v \in V$

$$x_e \geq 0 \qquad e \in E$$

Blossom Algorithm

For a subset S let $\delta(S)$ denote the boundary of S that is:

$$\delta(S) = \{(u, v) \in E\colon U \in S, v \notin S\}$$

O: all subsets of V of odd cardinality ≥ 3

Primal program

Minimize: $\sum_{e \in E} c_e \cdot x_e$

subject to: $\sum_{e \in \delta(v)} x_e = 1 \qquad v \in V$

$$x_e \geq 0 \qquad e \in E$$

$$\sum_{e \in \delta(S)} x_e \geq 1 \qquad S \in O$$

Blossom Algorithm

For a subset S let $\delta(S)$ denote the boundary of S that is:

$$\delta(S) = \{(u,v) \in E\colon U \in S, v \notin S\}$$

O: all subsets of V of odd cardinality ≥ 3

Primal program

Minimize: $\sum_{e \in E} c_e \cdot x_e$

subject to: $\sum_{e \in \delta(v)} x_e = 1 \quad v \in V$

$x_e \geq 0 \quad e \in E$

$\sum_{e \in \delta(S)} x_e \geq 1 \quad S \in O$

Dual program

Maximize: $\sum_{v \in V} y_v + \sum_{S \in O} y_S$

subject to: $\text{slack}(e) \geq 0 \quad e \in E$

$y_S \geq 0 \quad S \in O$

$\text{slack}(e) = c_e - y_u - y_v - \sum_{e \in \delta(S)} y_S$

Blossom Algorithm

For a subset S let $\delta(S)$ denote the boundary of S that is:

$$\delta(S) = \{(u, v) \in E\colon U \in S, v \notin S\}$$

O: all subsets of V of odd cardinality ≥ 3

Primal program

Minimize: $\sum_{e \in E} c_e \cdot x_e$

subject to: $\sum_{e \in \delta(v)} x_e = 1 \quad v \in V$

$x_e \geq 0 \quad e \in E$

Dual program

Maximize: $\sum_{v \in V} y_v$

subject to: $\text{slack}(e) \geq 0 \quad e \in E$

$\text{slack}(e) = c_e - y_u - y_v$

Blossom Algorithm

Primal program

Minimize: $\sum_{e \in E} c_e \cdot x_e$

subject to: $\sum_{e \in \delta(v)} x_e = 1 \qquad v \in V$

$x_e \geq 0 \qquad e \in E$

Dual program

Maximize: $\sum_{v \in V} y_v$

subject to: $\text{slack}(e) \geq 0 \qquad e \in E$

$\text{slack}(e) = c_e - y_u - y_v$

Blossom Algorithm

Primal program

Minimize: $\sum_{e \in E} c_e \cdot x_e$

subject to: $\sum_{e \in \delta(v)} x_e = 1 \qquad v \in V$

$x_e \geq 0 \qquad e \in E$

Dual program

Maximize: $\sum_{v \in V} y_v$

subject to: $y_u + y_v \leq c_e \qquad e \in E$

Blossom Algorithm

Primal program

Minimize: $\sum_{e \in E} c_e \cdot x_e$

subject to: $\sum_{e \in \delta(v)} x_e = 1 \qquad v \in V$

$x_e \geq 0 \qquad e \in E$

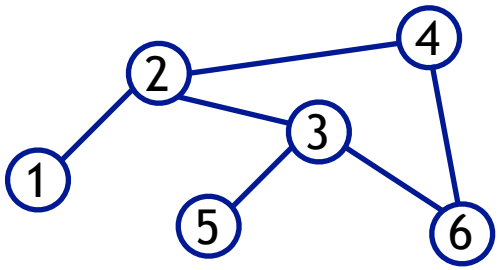
Dual program

Maximize: $\sum_{v \in V} y_v$

subject to: $y_u + y_v \leq c_e \qquad e \in E$

Intuition: split the cost of edges on the vertices.

Blossom Algorithm



Primal program

Minimize: $\sum_{e \in E} c_e \cdot x_e$

subject to: $\sum_{e \in \delta(v)} x_e = 1 \qquad v \in V$

$x_e \geq 0 \qquad e \in E$

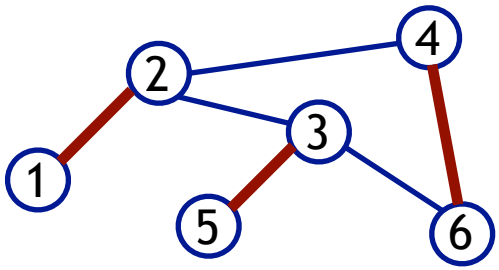
Dual program

Maximize: $\sum_{v \in V} y_v$

subject to: $y_u + y_v \leq c_e \qquad e \in E$

Intuition: split the cost of edges on the vertices.

Blossom Algorithm



$$c_{1,2} + c_{3,5} + c_{4,6}$$

Primal program

Minimize: $\sum_{e \in E} c_e \cdot x_e$

subject to: $\sum_{e \in \delta(v)} x_e = 1 \qquad v \in V$

$x_e \geq 0 \qquad e \in E$

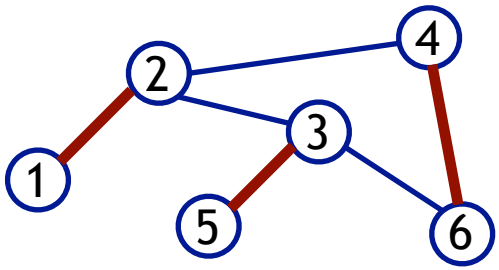
Dual program

Maximize: $\sum_{v \in V} y_v$

subject to: $y_u + y_v \leq c_e \qquad e \in E$

Intuition: split the cost of edges on the vertices.

Blossom Algorithm



$$c_{1,2} + c_{3,5} + c_{4,6} \leq y_1 + y_2 + y_3 + y_5 + y_4 + y_6$$

Primal program

Minimize: $\sum_{e \in E} c_e \cdot x_e$

subject to: $\sum_{e \in \delta(v)} x_e = 1 \qquad v \in V$

$x_e \geq 0 \qquad e \in E$

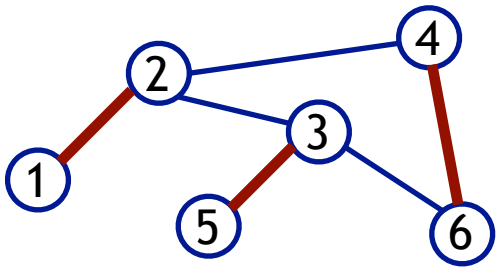
Dual program

Maximize: $\sum_{v \in V} y_v$

subject to: $y_u + y_v \leq c_e \qquad e \in E$

Intuition: split the cost of edges on the vertices.

Blossom Algorithm



$$c_{1,2} + c_{3,5} + c_{4,6} \leq y_1 + y_2 + y_3 + y_5 + y_4 + y_6$$

Primal program

Minimize: $\sum_{e \in E} c_e \cdot x_e$

subject to: $\sum_{e \in \delta(v)} x_e = 1 \qquad v \in V$

$x_e \geq 0 \qquad e \in E$

Dual program

Maximize: $\sum_{v \in V} y_v$

subject to: $y_u + y_v \leq c_e \qquad e \in E$

Intuition: split the cost of edges on the vertices.

Blossom Algorithm

For a subset S let $\delta(S)$ denote the boundary of S that is:

$$\delta(S) = \{(u,v) \in E\colon U \in S, v \notin S\}$$

O: all subsets of V of odd cardinality ≥ 3

Primal program

Minimize: $\sum_{e \in E} c_e \cdot x_e$

subject to: $\sum_{e \in \delta(v)} x_e = 1 \quad v \in V$

$x_e \geq 0 \quad e \in E$

$\sum_{e \in \delta(S)} x_e \geq 1 \quad S \in O$

Dual program

Maximize: $\sum_{v \in V} y_v + \sum_{S \in O} y_S$

subject to: $\text{slack}(e) \geq 0 \quad e \in E$

$y_S \geq 0 \quad S \in O$

$\text{slack}(e) = c_e - y_u - y_v - \sum_{e \in \delta(S)} y_S$

Blossom Algorithm

For a subset S let $\delta(S)$ denote the boundary of S that is:

$$\delta(S) = \{(u,v) \in E : u \in S, v \notin S\}$$

$$\geq 3$$

Intuition: sometimes we assign the values not only to vertices but also to (A small number of) groups of vertices.

Dual program

mize: $\sum_{v \in V} y_v + \sum_{S \in O} y_S$

subject to: $\sum_{e \in \delta(v)} x_e = 1 \quad v \in V$

$x_e \geq 0 \quad e \in E$

$\sum_{e \in \delta(S)} x_e \geq 1 \quad S \in O$

subject to: $\text{slack}(e) \geq 0 \quad e \in E$

$y_S \geq 0 \quad S \in O$

$\text{slack}(e) = c_e - y_u - y_v - \sum_{e \in \delta(S)} y_S$

Blossom Algorithm

For a subset S let $\delta(S)$ denote the boundary of S that is:

$$\delta(S) = \{(u, v) \in E\colon U \in S, v \notin S\}$$

O: all subsets of V of odd cardinality ≥ 3

Dual program

Maximize: $\sum_{v \in V} y_v + \sum_{S \in O} y_S$

e is tight if $\text{slack}(e) = 0$ → subject to: $\text{slack}(e) \geq 0 \quad e \in E$

$$y_S \geq 0 \qquad S \in O$$

$$\text{slack}(e) = c_e - y_u - y_v - \sum_{e \in \delta(S)} y_S$$

Blossom Algorithm

We build iteratively a tree (or a forest).

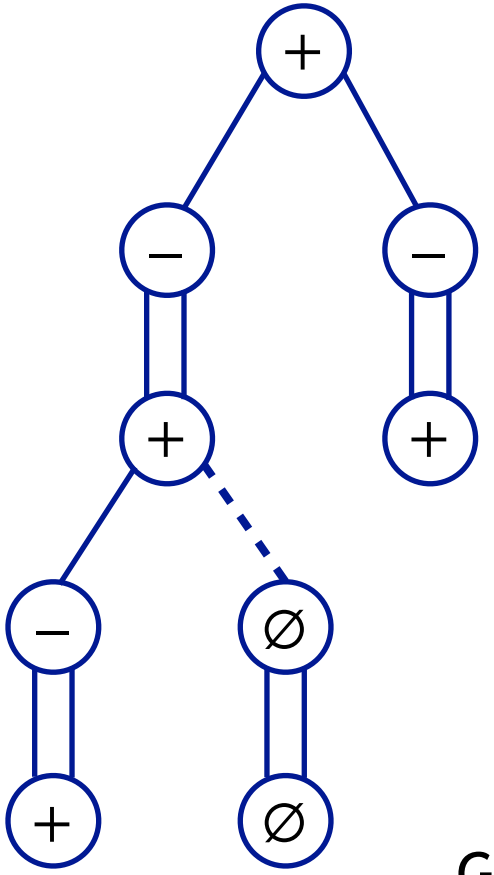
The vertices v will be labelled:

- $\varnothing$: nodes matched via a tight edge.
- $+$: an unmatched vertex being a root of a tree, or a child of a vertex labelled $-$ to which v is matched via a tight edge.
- $-$: a child of a vertex labelled $+$ to which v is connected via a tight unmatched edge.

At the beginning all unmatched vertices are $+$.

Blossom Algorithm

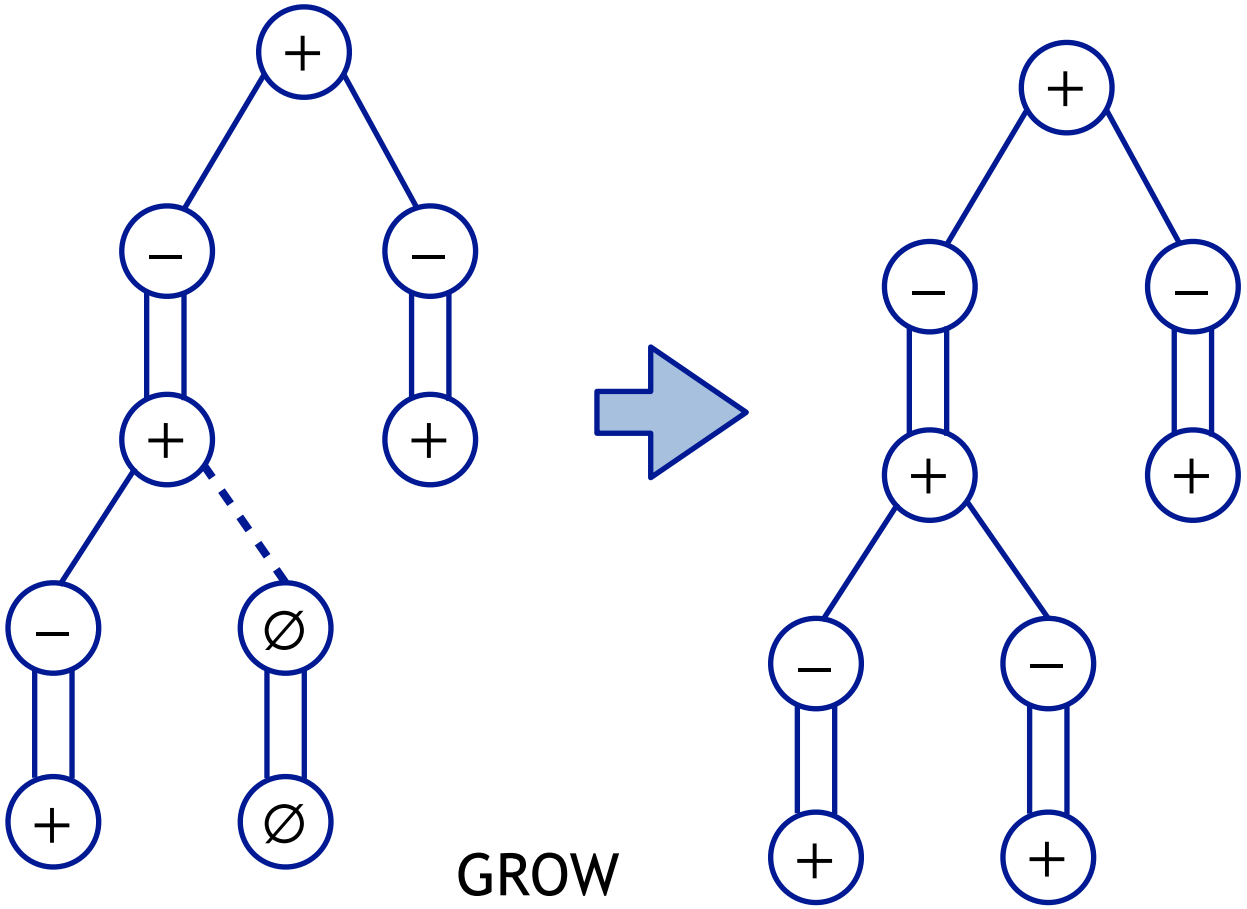
Four operations:



GROW

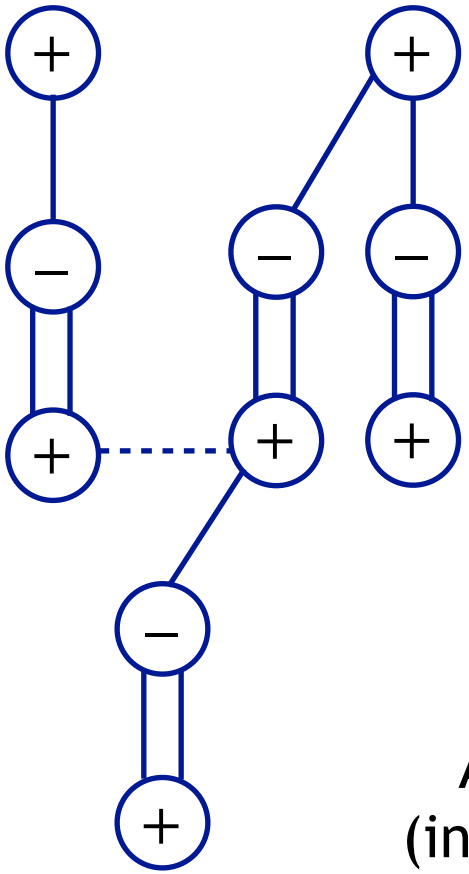
Blossom Algorithm

Four operations:



Blossom Algorithm

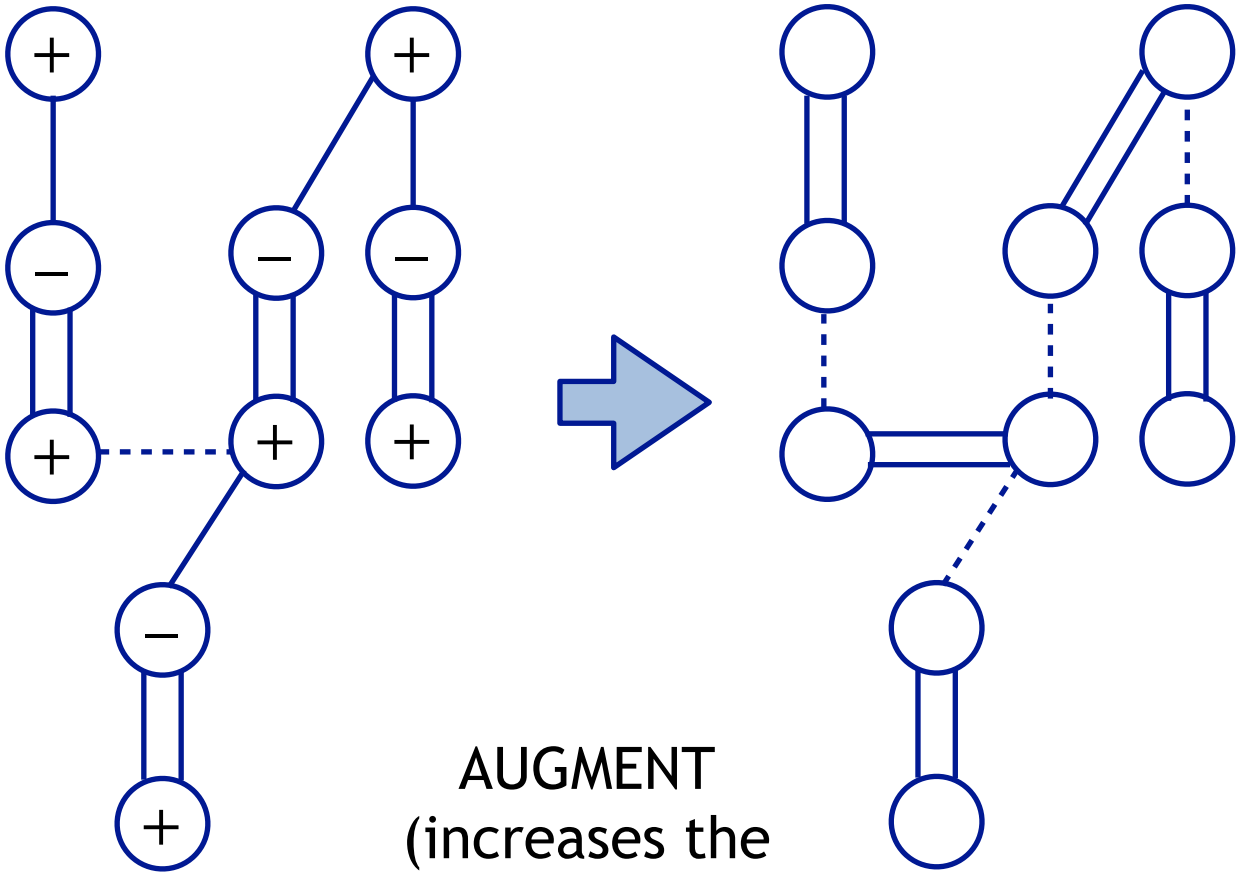
Four operations:



AUGMENT
(increases the
size of a matching)

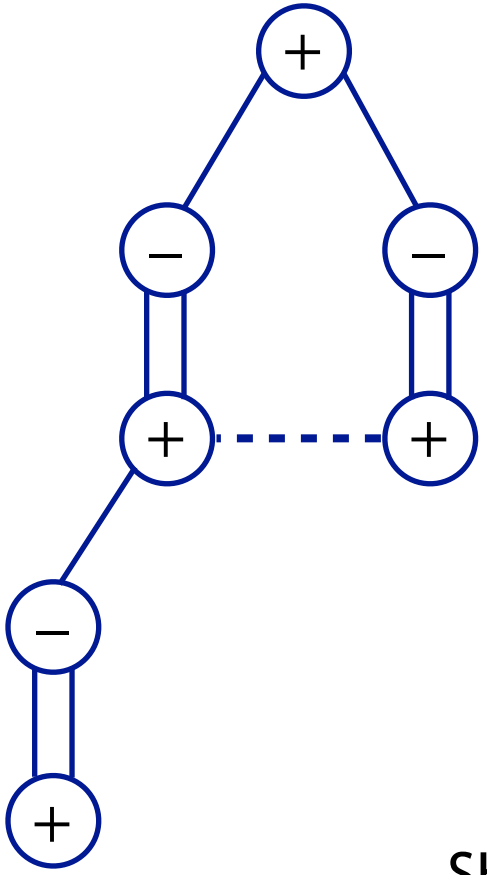
Blossom Algorithm

Four operations:



Blossom Algorithm

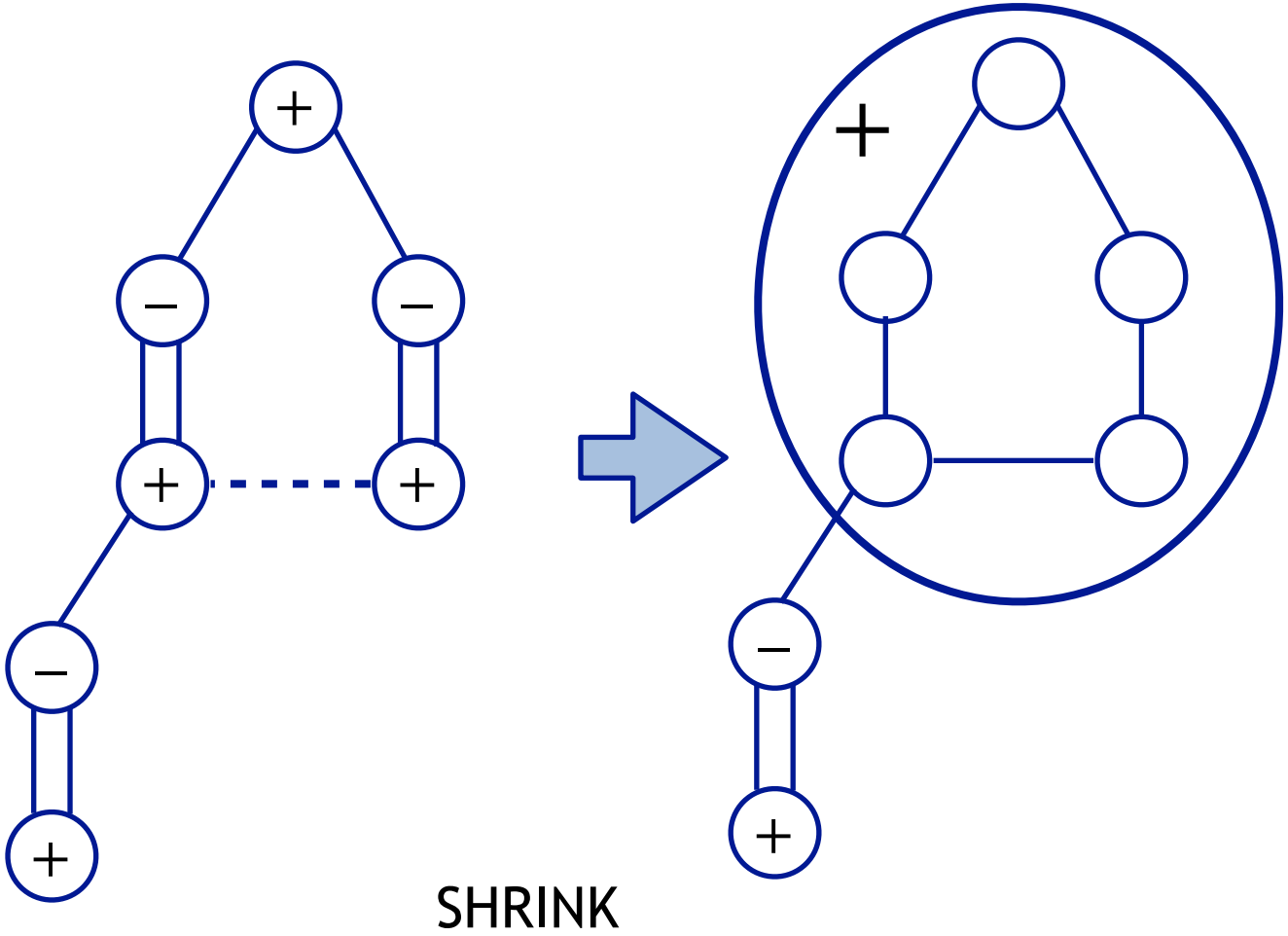
Four operations:



SHRINK

Blossom Algorithm

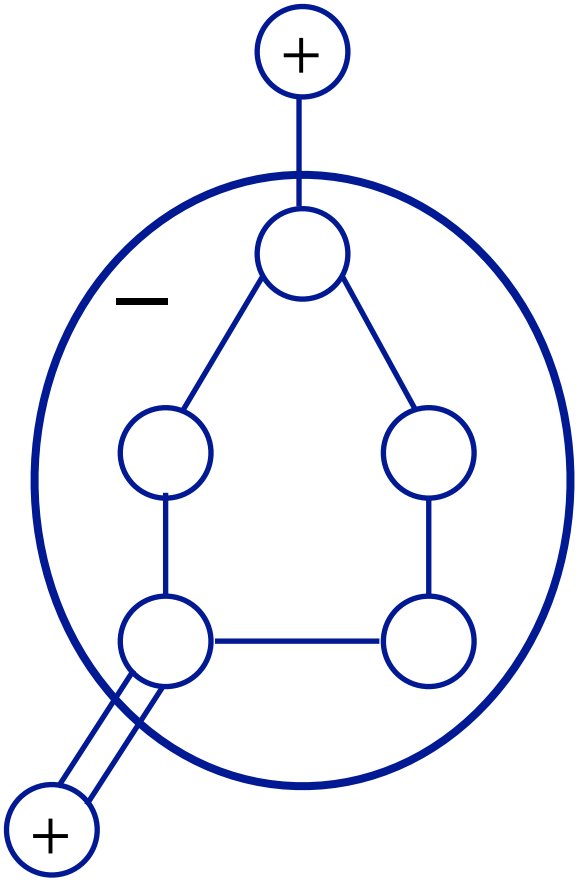
Four operations:



SHRINK

Blossom Algorithm

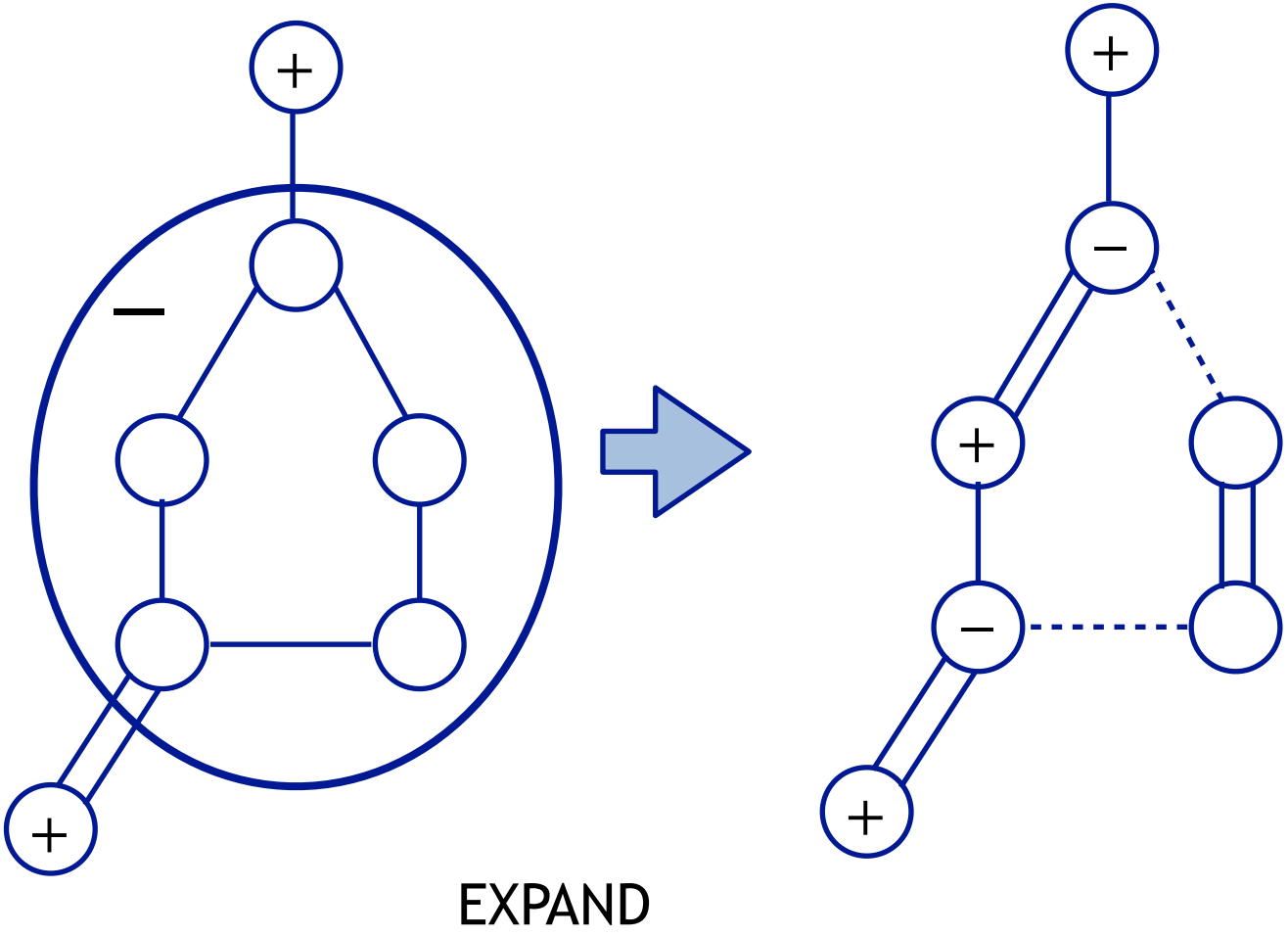
Four operations:



EXPAND

Blossom Algorithm

Four operations:



EXPAND

Blossom Algorithm

If there are no tight edges we find a value of δ_T, and:

- If v has label $-$ then $y_v := y_v - \delta_T$
- If v has label $+$ then $y_v := y_v + \delta_T$

Blossom Algorithm

If there are no tight edges we find a value of δ_T, and:

- If v has label $-$ then $y_v := y_v - \delta_T$
- If v has label $+$ then $y_v := y_v + \delta_T$

The dual objective is increased. Why?

Blossom Algorithm

If there are no tight edges we find a value of δ_T, and:

- If v has label $-$ then $y_v := y_v - \delta_T$
- If v has label $+$ then $y_v := y_v + \delta_T$

The dual objective is increased. Why?

The dual must remain feasible thus:

1. $\delta_T \leq \text{slack}(u, v)$ $\quad$ (u, v) is an $(+, \varnothing)$ and $u \in T$

Blossom Algorithm

If there are no tight edges we find a value of δ_T, and:

- If v has label $-$ then $y_v := y_v - \delta_T$
- If v has label $+$ then $y_v := y_v + \delta_T$

The dual objective is increased. Why?

The dual must remain feasible thus:

1. $\delta_T \leq \text{slack}(u, v)$ (u, v) is an $(+, \varnothing)$ and $u \in T$
2. $\delta_T \leq \text{slack}(u, v)$ (u, v) is an $(+, +)$ and $u \in T$, $v \in T'$

Blossom Algorithm

If there are no tight edges we find a value of δ_T, and:

- If v has label $-$ then $y_v := y_v - \delta_T$
- If v has label $+$ then $y_v := y_v + \delta_T$

The dual objective is increased. Why?

The dual must remain feasible thus:

1. $\delta_T \leq \text{slack}(u, v)$ (u, v) is an $(+, \varnothing)$ and $u \in T$
2. $\delta_T \leq \text{slack}(u, v)$ (u, v) is an $(+, +)$ and $u \in T$, $v \in T'$
3. $\delta_T \leq \text{slack}(u, v)/2$ (u, v) is an $(+, +)$ and $u, v \in T$

Blossom Algorithm

If there are no tight edges we find a value of δ_T, and:

- If v has label $-$ then $y_v := y_v - \delta_T$
- If v has label $+$ then $y_v := y_v + \delta_T$

The dual objective is increased. Why?

The dual must remain feasible thus:

1. $\delta_T \leq \text{slack}(u, v)$ (u, v) is an $(+, \varnothing)$ and $u \in T$
2. $\delta_T \leq \text{slack}(u, v)$ (u, v) is an $(+, +)$ and $u \in T$, $v \in T'$
3. $\delta_T \leq \text{slack}(u, v)/2$ (u, v) is an $(+, +)$ and $u, v \in T$
4. $\delta_T \leq y_v$ v is a $-$ blossom

Blossom Algorithm

If there are no tight edges we find a value of δ_T, and:

- If v has label $-$ then $y_v := y_v - \delta_T$
- If v has label $+$ then $y_v := y_v + \delta_T$

The dual objective is increased. Why?

The dual must remain feasible thus:

1. $\delta_T \leq \text{slack}(u,v)$ (u,v) is an $(+,\emptyset)$ and $u \in T$
2. $\delta_T \leq \text{slack}(u,v)$ (u,v) is an $(+,+)$ and $u \in T$, $v \in T'$
3. $\delta_T \leq \text{slack}(u,v)/2$ (u,v) is an $(+,+)$ and $u,v \in T$
4. $\delta_T \leq y_v$ v is a $-$ blossom

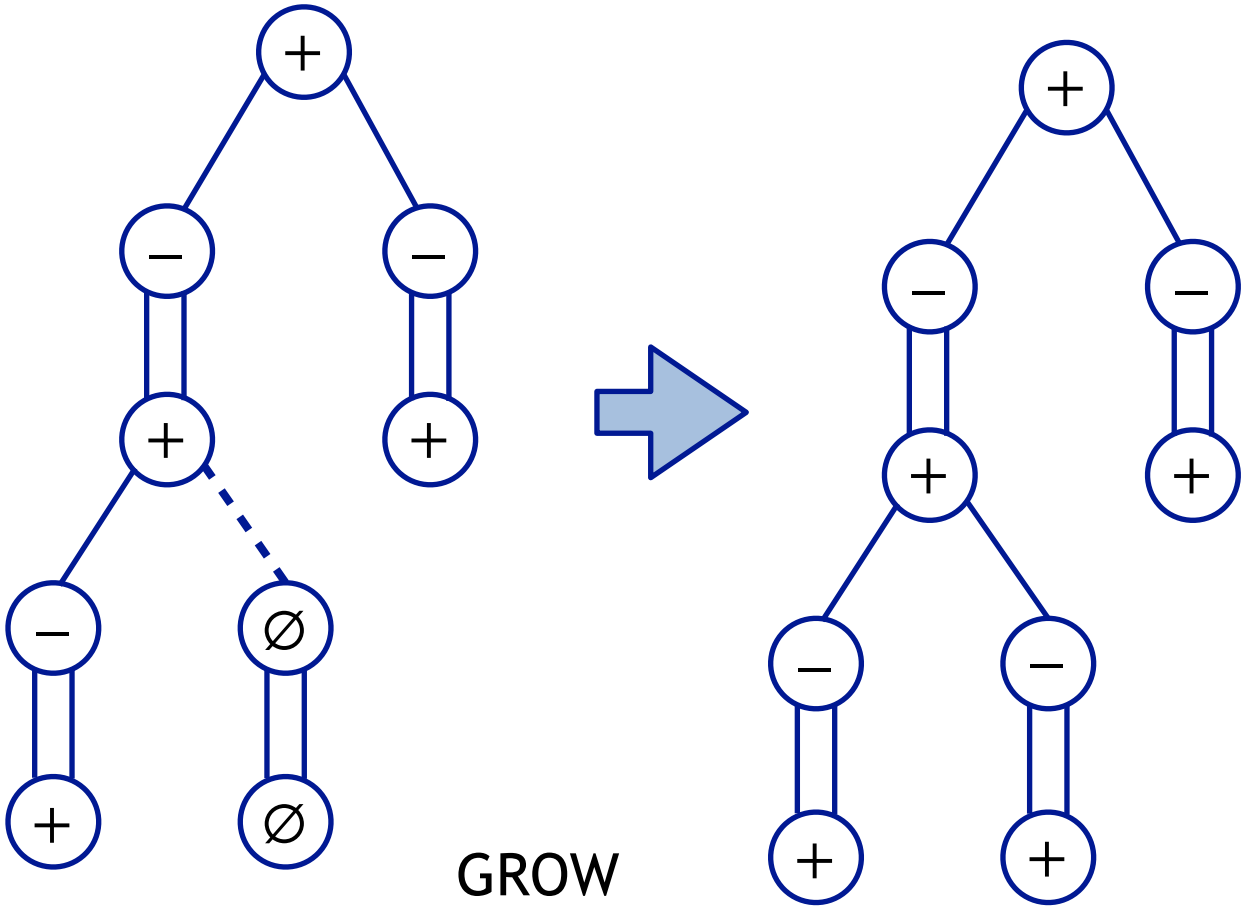
Find the first δ_T for which some constraint becomes tight!

Blossom Algorithm

1. $\delta_T \leq \text{slack}(u, v)$ (u, v) is an $(+, \varnothing)$ and $u \in T$

Blossom Algorithm

1. $\delta_T \leq \text{slack}(u, v)$ (u, v) is an $(+, \varnothing)$ and $u \in T$

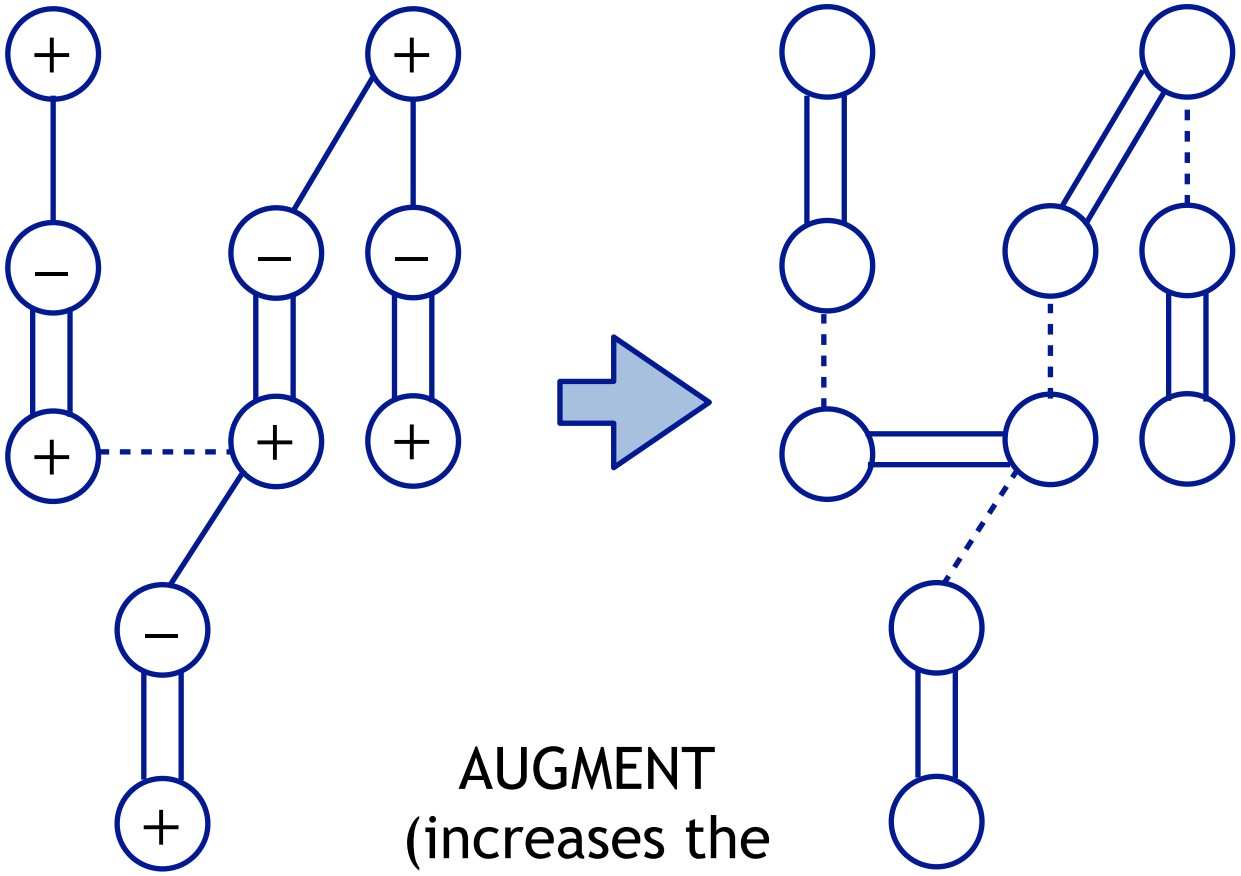


Blossom Algorithm

2. $\delta_T \leq \text{slack}(u, v)$ (u, v) is an $(+, +)$ and $u \in T$, $v \in T'$

Blossom Algorithm

2. $\delta_T \leq \text{slack}(u, v)$ (u, v) is an $(+, +)$ and $u \in T$, $v \in T'$

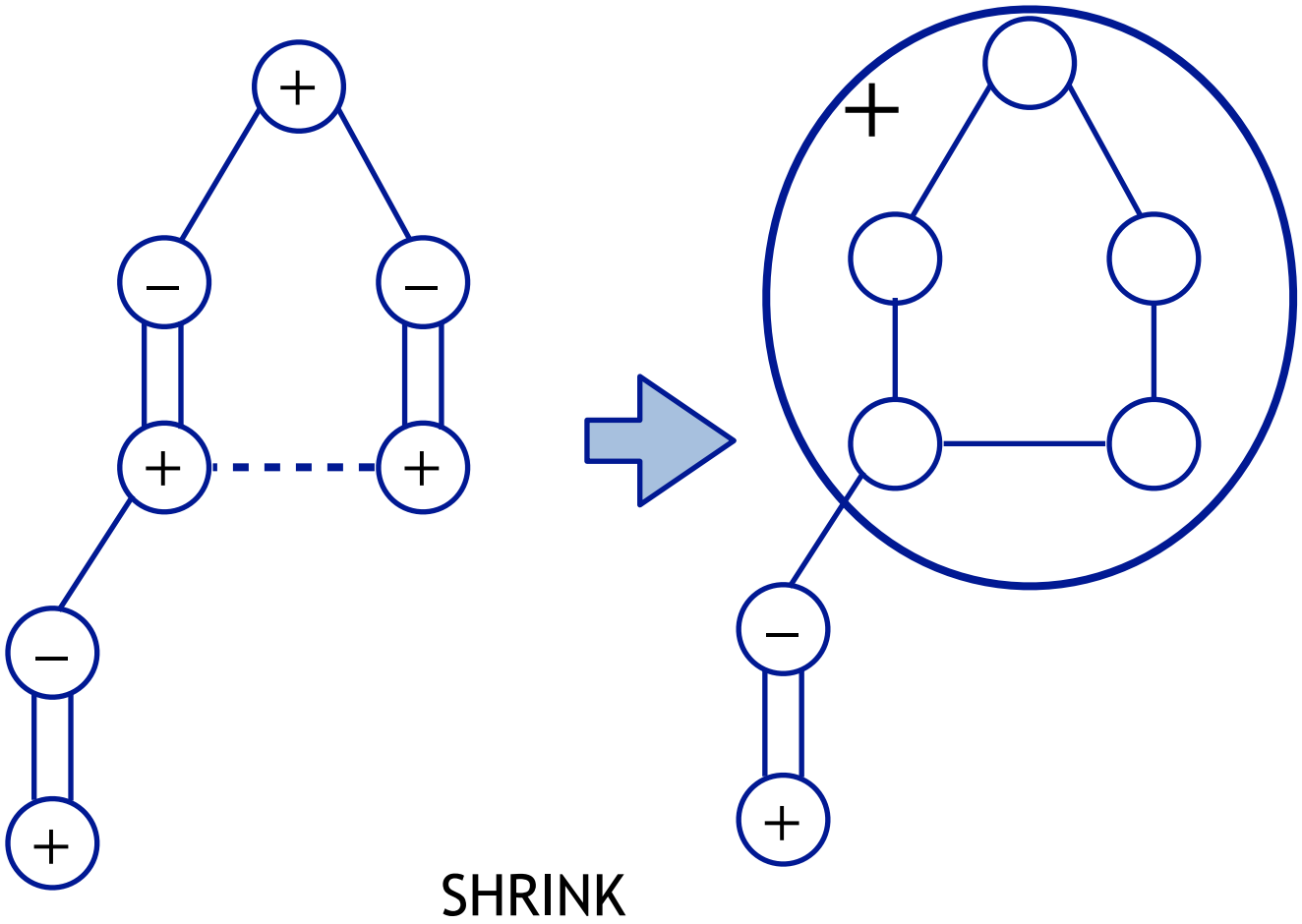


Blossom Algorithm

3. $\delta_T \leq \text{slack}(u, v)/2 \quad (u, v)$ is an $(+, +)$ and $u, v \in T$

Blossom Algorithm

3. $\delta_T \leq \text{slack}(u, v)/2$ (u, v) is an $(+, +)$ and $u, v \in T$

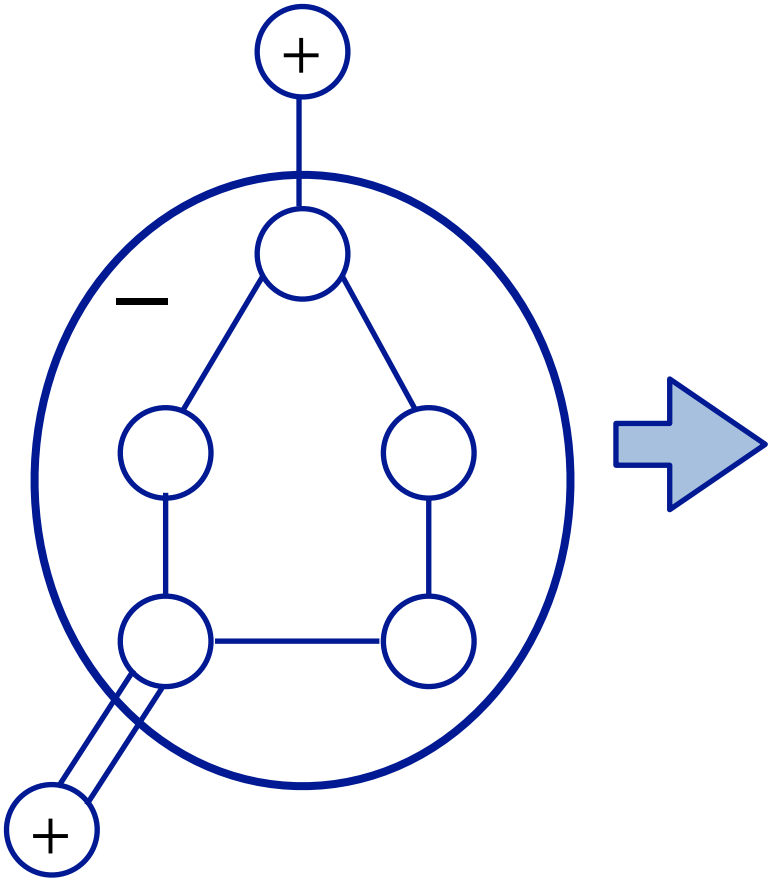


Blossom Algorithm

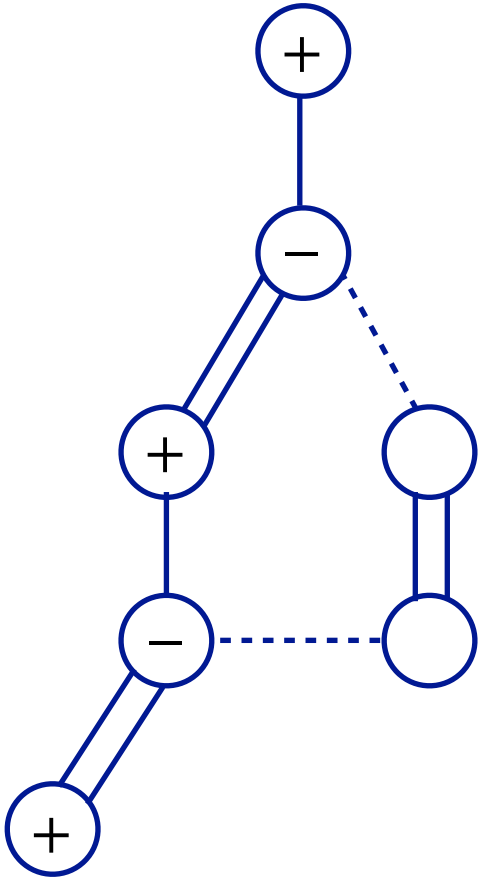
4. $\delta_T \leq y_v$ v is a $-$ blossom

Blossom Algorithm

4. $\delta_T \leq y_v$



v is a − blossom



EXPAND

Blossom Algorithm: Example

Clique Description

	1	2	3	4	5	6	7	8	9	10
1										
2	42									
3	4	68								
4	6	36	44							
5	14	26	30	2						
6	8	62	26	60	8					
7	34	50	4	42	42	52				
8	58	6	32	44	34	70	38			
9	68	52	60	30	44	6	38	24		
10	64	24	40	36	62	48	50	32	10	

Blossom Algorithm: Example

Clique Description

	1	2	3	4	5	6	7	8	9	10
1		42	4	6	14	8	34	58	68	64
2	42		68	36	26	62	50	6	52	24
3	4	68		44	30	26	4	32	60	40
4	6	36	44		2	60	42	44	30	36
5	14	26	30	2		8	42	34	44	62
6	8	62	26	60	8		52	70	6	48
7	34	50	4	42	42	52		38	38	50
8	58	6	32	44	34	70	38		24	32
9	68	52	60	30	44	6	38	24		10
10	64	24	40	36	62	48	50	32	10	

Blossom Algorithm: Example

Clique Description

y	0	0	0	0	0	0	0	0	0	0
	1	2	3	4	5	6	7	8	9	10
1		42	4	6	14	8	34	58	68	64
2	42		68	36	26	62	50	6	52	24
3	4	68		44	30	26	4	32	60	40
4	6	36	44		2	60	42	44	30	36
5	14	26	30	2		8	42	34	44	62
6	8	62	26	60	8		52	70	6	48
7	34	50	4	42	42	52		38	38	50
8	58	6	32	44	34	70	38		24	32
9	68	52	60	30	44	6	38	24		10
10	64	24	40	36	62	48	50	32	10	

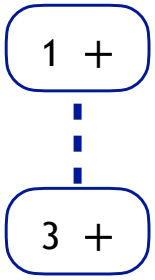
Blossom Algorithm: Example

1 +

Clique Description

y	0	0	0	0	0	0	0	0	0	0
	1	2	3	4	5	6	7	8	9	10
1		42	4	6	14	8	34	58	68	64
2	42		68	36	26	62	50	6	52	24
3	4	68		44	30	26	4	32	60	40
4	6	36	44		2	60	42	44	30	36
5	14	26	30	2		8	42	34	44	62
6	8	62	26	60	8		52	70	6	48
7	34	50	4	42	42	52		38	38	50
8	58	6	32	44	34	70	38		24	32
9	68	52	60	30	44	6	38	24		10
10	64	24	40	36	62	48	50	32	10	

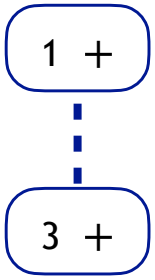
Blossom Algorithm: Example



Clique Description

y	0	0	0	0	0	0	0	0	0	0
	1	2	3	4	5	6	7	8	9	10
1		42	4	6	14	8	34	58	68	64
2	42		68	36	26	62	50	6	52	24
3	4	68		44	30	26	4	32	60	40
4	6	36	44		2	60	42	44	30	36
5	14	26	30	2		8	42	34	44	62
6	8	62	26	60	8		52	70	6	48
7	34	50	4	42	42	52		38	38	50
8	58	6	32	44	34	70	38		24	32
9	68	52	60	30	44	6	38	24		10
10	64	24	40	36	62	48	50	32	10	

Blossom Algorithm: Example

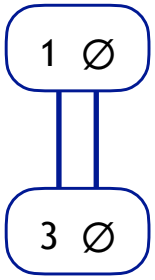


Clique Description

y	4	0	0	0	0	0	0	0	0	0
	1	2	3	4	5	6	7	8	9	10
1		42	4	6	14	8	34	58	68	64
2	42		68	36	26	62	50	6	52	24
3	4	68		44	30	26	4	32	60	40
4	6	36	44		2	60	42	44	30	36
5	14	26	30	2		8	42	34	44	62
6	8	62	26	60	8		52	70	6	48
7	34	50	4	42	42	52		38	38	50
8	58	6	32	44	34	70	38		24	32
9	68	52	60	30	44	6	38	24		10
10	64	24	40	36	62	48	50	32	10	

DUAL VARIABLES UPDATE

Blossom Algorithm: Example

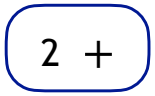


Clique Description

y	4	0	0	0	0	0	0	0	0	0
	1	2	3	4	5	6	7	8	9	10
1		42	4	6	14	8	34	58	68	64
2	42		68	36	26	62	50	6	52	24
3	4	68		44	30	26	4	32	60	40
4	6	36	44		2	60	42	44	30	36
5	14	26	30	2		8	42	34	44	62
6	8	62	26	60	8		52	70	6	48
7	34	50	4	42	42	52		38	38	50
8	58	6	32	44	34	70	38		24	32
9	68	52	60	30	44	6	38	24		10
10	64	24	40	36	62	48	50	32	10	

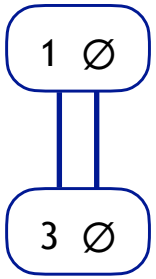
AUGMENT

Blossom Algorithm: Example

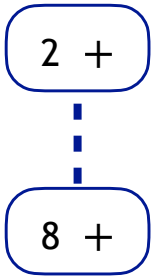


Clique Description

y	4	0	0	0	0	0	0	0	0	0
	1	2	3	4	5	6	7	8	9	10
1		42	4	6	14	8	34	58	68	64
2	42		68	36	26	62	50	6	52	24
3	4	68		44	30	26	4	32	60	40
4	6	36	44		2	60	42	44	30	36
5	14	26	30	2		8	42	34	44	62
6	8	62	26	60	8		52	70	6	48
7	34	50	4	42	42	52		38	38	50
8	58	6	32	44	34	70	38		24	32
9	68	52	60	30	44	6	38	24		10
10	64	24	40	36	62	48	50	32	10	

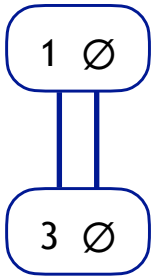


Blossom Algorithm: Example

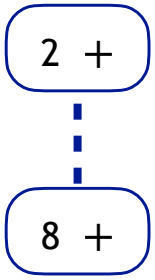


Clique Description

y	4	0	0	0	0	0	0	0	0	0
	1	2	3	4	5	6	7	8	9	10
1		42	4	6	14	8	34	58	68	64
2	42		68	36	26	62	50	6	52	24
3	4	68		44	30	26	4	32	60	40
4	6	36	44		2	60	42	44	30	36
5	14	26	30	2		8	42	34	44	62
6	8	62	26	60	8		52	70	6	48
7	34	50	4	42	42	52		38	38	50
8	58	6	32	44	34	70	38		24	32
9	68	52	60	30	44	6	38	24		10
10	64	24	40	36	62	48	50	32	10	

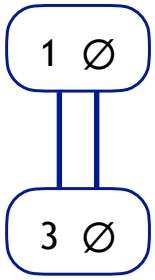


Blossom Algorithm: Example



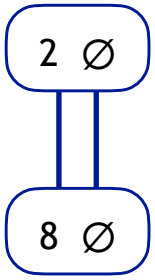
Clique Description

y	4	6	0	0	0	0	0	0	0	0
	1	2	3	4	5	6	7	8	9	10
1		42	4	6	14	8	34	58	68	64
2	42		68	36	26	62	50	6	52	24
3	4	68		44	30	26	4	32	60	40
4	6	36	44		2	60	42	44	30	36
5	14	26	30	2		8	42	34	44	62
6	8	62	26	60	8		52	70	6	48
7	34	50	4	42	42	52		38	38	50
8	58	6	32	44	34	70	38		24	32
9	68	52	60	30	44	6	38	24		10
10	64	24	40	36	62	48	50	32	10	



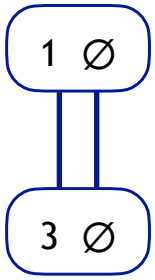
DUAL VARIABLES UPDATE

Blossom Algorithm: Example



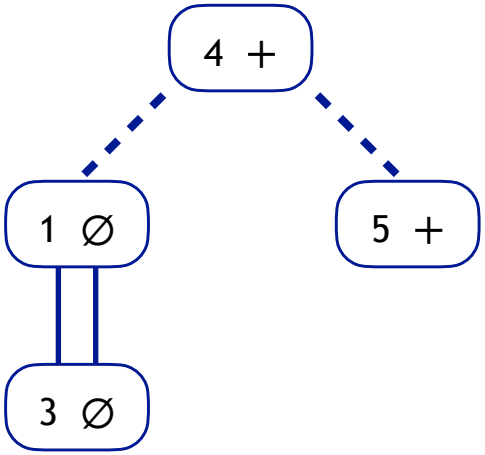
Clique Description

y	4	6	0	0	0	0	0	0	0	0
	1	2	3	4	5	6	7	8	9	10
1		42	4	6	14	8	34	58	68	64
2	42		68	36	26	62	50	6	52	24
3	4	68		44	30	26	4	32	60	40
4	6	36	44		2	60	42	44	30	36
5	14	26	30	2		8	42	34	44	62
6	8	62	26	60	8		52	70	6	48
7	34	50	4	42	42	52		38	38	50
8	58	6	32	44	34	70	38		24	32
9	68	52	60	30	44	6	38	24		10
10	64	24	40	36	62	48	50	32	10	



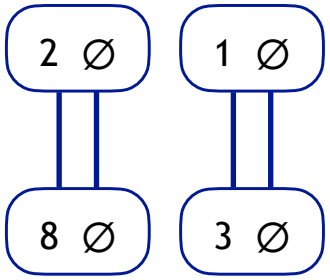
AUGMENT

Blossom Algorithm: Example

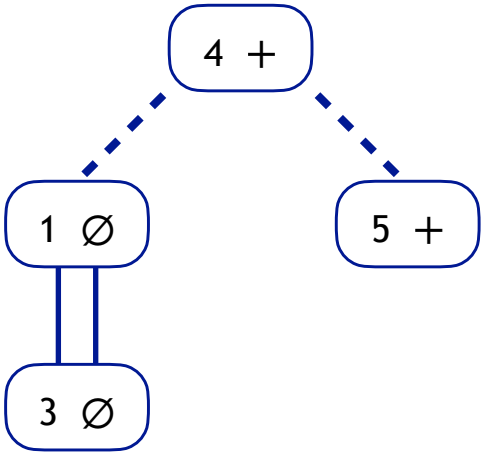


Clique Description

y	4	6	0	0	0	0	0	0	0	0
	1	2	3	4	5	6	7	8	9	10
1		42	4	6	14	8	34	58	68	64
2	42		68	36	26	62	50	6	52	24
3	4	68		44	30	26	4	32	60	40
4	6	36	44		2	60	42	44	30	36
5	14	26	30	2		8	42	34	44	62
6	8	62	26	60	8		52	70	6	48
7	34	50	4	42	42	52		38	38	50
8	58	6	32	44	34	70	38		24	32
9	68	52	60	30	44	6	38	24		10
10	64	24	40	36	62	48	50	32	10	

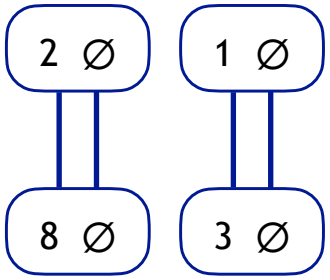


Blossom Algorithm: Example



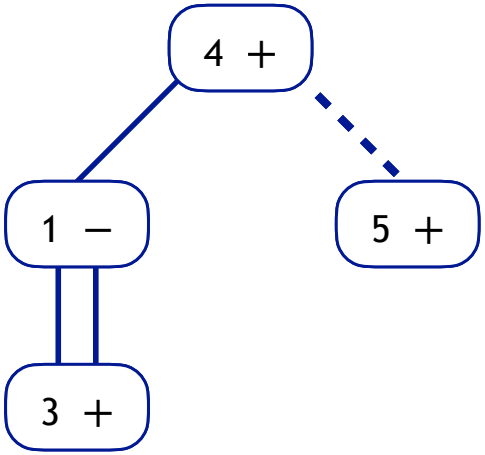
Clique Description

y	4	6	0	2	0	0	0	0	0	0
	1	2	3	4	5	6	7	8	9	10
1		42	4	6	14	8	34	58	68	64
2	42		68	36	26	62	50	6	52	24
3	4	68		44	30	26	4	32	60	40
4	6	36	44		2	60	42	44	30	36
5	14	26	30	2		8	42	34	44	62
6	8	62	26	60	8		52	70	6	48
7	34	50	4	42	42	52		38	38	50
8	58	6	32	44	34	70	38		24	32
9	68	52	60	30	44	6	38	24		10
10	64	24	40	36	62	48	50	32	10	



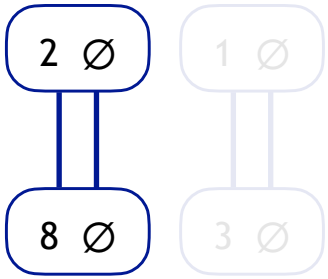
DUAL VARIABLES UPDATE

Blossom Algorithm: Example



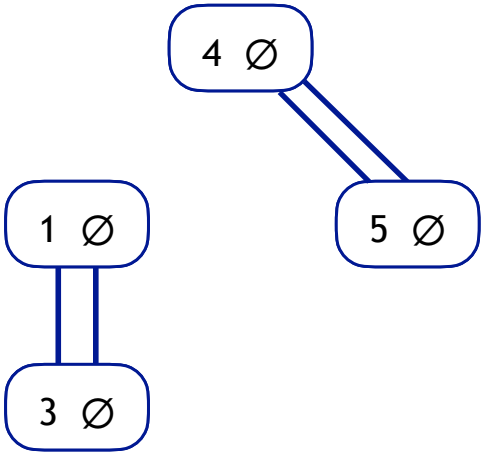
Clique Description

y	4	6	0	2	0	0	0	0	0	0
	1	2	3	4	5	6	7	8	9	10
1		42	4	6	14	8	34	58	68	64
2	42		68	36	26	62	50	6	52	24
3	4	68		44	30	26	4	32	60	40
4	6	36	44		2	60	42	44	30	36
5	14	26	30	2		8	42	34	44	62
6	8	62	26	60	8		52	70	6	48
7	34	50	4	42	42	52		38	38	50
8	58	6	32	44	34	70	38		24	32
9	68	52	60	30	44	6	38	24		10
10	64	24	40	36	62	48	50	32	10	



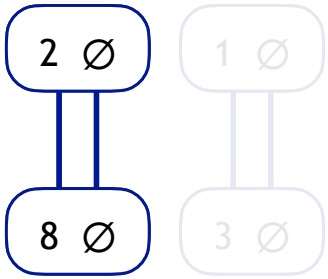
GROW

Blossom Algorithm: Example



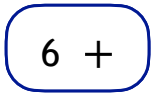
Clique Description

y	4	6	0	2	0	0	0	0	0	0
	1	2	3	4	5	6	7	8	9	10
1		42	4	6	14	8	34	58	68	64
2	42		68	36	26	62	50	6	52	24
3	4	68		44	30	26	4	32	60	40
4	6	36	44		2	60	42	44	30	36
5	14	26	30	2		8	42	34	44	62
6	8	62	26	60	8		52	70	6	48
7	34	50	4	42	42	52		38	38	50
8	58	6	32	44	34	70	38		24	32
9	68	52	60	30	44	6	38	24		10
10	64	24	40	36	62	48	50	32	10	



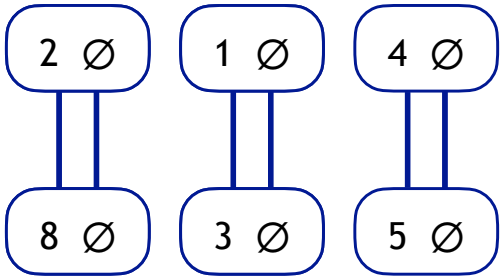
AUGMENT

Blossom Algorithm: Example

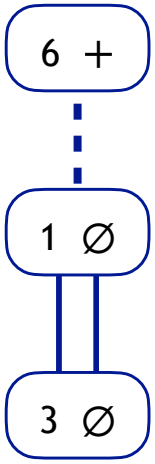


Clique Description

y	4	6	0	2	0	0	0	0	0	0
	1	2	3	4	5	6	7	8	9	10
1		42	4	6	14	8	34	58	68	64
2	42		68	36	26	62	50	6	52	24
3	4	68		44	30	26	4	32	60	40
4	6	36	44		2	60	42	44	30	36
5	14	26	30	2		8	42	34	44	62
6	8	62	26	60	8		52	70	6	48
7	34	50	4	42	42	52		38	38	50
8	58	6	32	44	34	70	38		24	32
9	68	52	60	30	44	6	38	24		10
10	64	24	40	36	62	48	50	32	10	

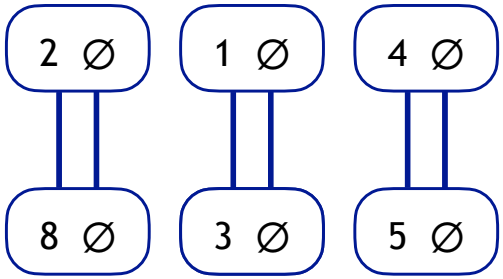


Blossom Algorithm: Example

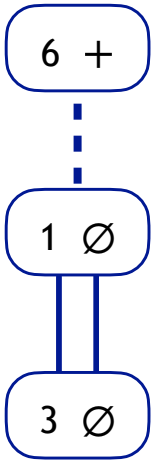


Clique Description

y	4	6	0	2	0	0	0	0	0	0
	1	2	3	4	5	6	7	8	9	10
1		42	4	6	14	8	34	58	68	64
2	42		68	36	26	62	50	6	52	24
3	4	68		44	30	26	4	32	60	40
4	6	36	44		2	60	42	44	30	36
5	14	26	30	2		8	42	34	44	62
6	8	62	26	60	8		52	70	6	48
7	34	50	4	42	42	52		38	38	50
8	58	6	32	44	34	70	38		24	32
9	68	52	60	30	44	6	38	24		10
10	64	24	40	36	62	48	50	32	10	

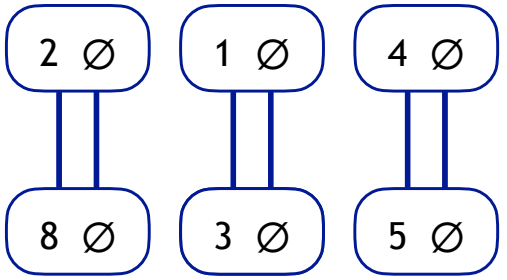


Blossom Algorithm: Example



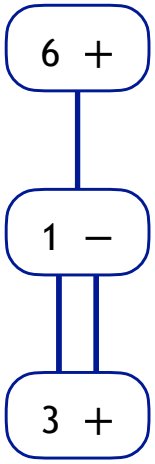
Clique Description

y	4	6	0	2	0	4	0	0	0	0
	1	2	3	4	5	6	7	8	9	10
1		42	4	6	14	8	34	58	68	64
2	42		68	36	26	62	50	6	52	24
3	4	68		44	30	26	4	32	60	40
4	6	36	44		2	60	42	44	30	36
5	14	26	30	2		8	42	34	44	62
6	8	62	26	60	8		52	70	6	48
7	34	50	4	42	42	52		38	38	50
8	58	6	32	44	34	70	38		24	32
9	68	52	60	30	44	6	38	24		10
10	64	24	40	36	62	48	50	32	10	



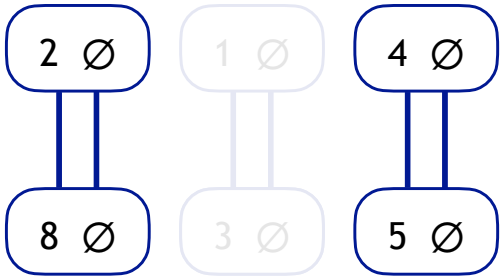
DUAL VARIABLES UPDATE

Blossom Algorithm: Example



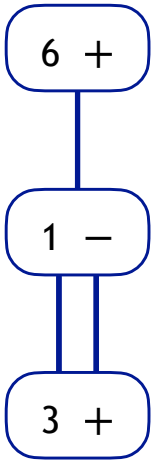
Clique Description

y	4	6	0	2	0	4	0	0	0	0
	1	2	3	4	5	6	7	8	9	10
1		42	4	6	14	8	34	58	68	64
2	42		68	36	26	62	50	6	52	24
3	4	68		44	30	26	4	32	60	40
4	6	36	44		2	60	42	44	30	36
5	14	26	30	2		8	42	34	44	62
6	8	62	26	60	8		52	70	6	48
7	34	50	4	42	42	52		38	38	50
8	58	6	32	44	34	70	38		24	32
9	68	52	60	30	44	6	38	24		10
10	64	24	40	36	62	48	50	32	10	



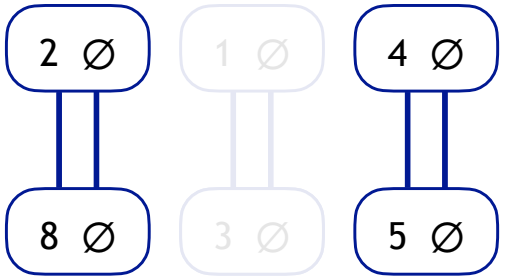
GROW

Blossom Algorithm: Example

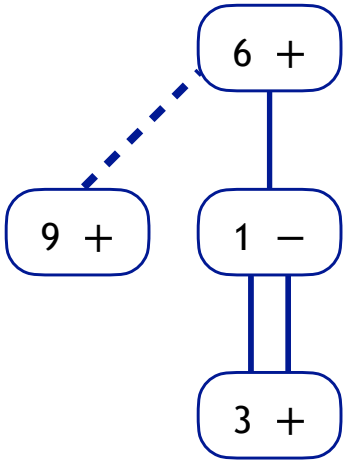


Clique Description

y	4	6	0	2	0	4	0	0	0	0
	1	2	3	4	5	6	7	8	9	10
1		42	4	6	14	8	34	58	68	64
2	42		68	36	26	62	50	6	52	24
3	4	68		44	30	26	4	32	60	40
4	6	36	44		2	60	42	44	30	36
5	14	26	30	2		8	42	34	44	62
6	8	62	26	60	8		52	70	6	48
7	34	50	4	42	42	52		38	38	50
8	58	6	32	44	34	70	38		24	32
9	68	52	60	30	44	6	38	24		10
10	64	24	40	36	62	48	50	32	10	

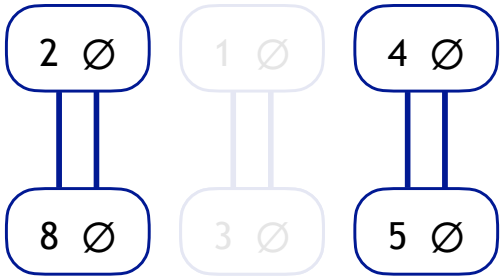


Blossom Algorithm: Example

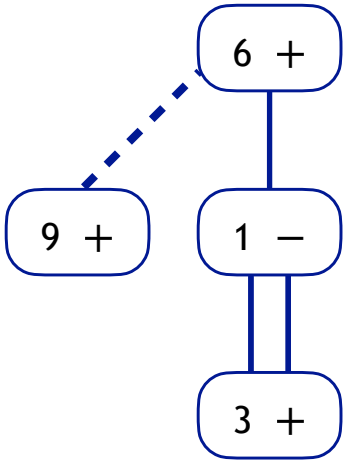


Clique Description

y	4	6	0	2	0	4	0	0	0	0
	1	2	3	4	5	6	7	8	9	10
1		42	4	6	14	8	34	58	68	64
2	42		68	36	26	62	50	6	52	24
3	4	68		44	30	26	4	32	60	40
4	6	36	44		2	60	42	44	30	36
5	14	26	30	2		8	42	34	44	62
6	8	62	26	60	8		52	70	6	48
7	34	50	4	42	42	52		38	38	50
8	58	6	32	44	34	70	38		24	32
9	68	52	60	30	44	6	38	24		10
10	64	24	40	36	62	48	50	32	10	

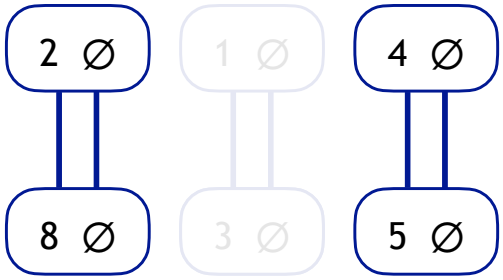


Blossom Algorithm: Example



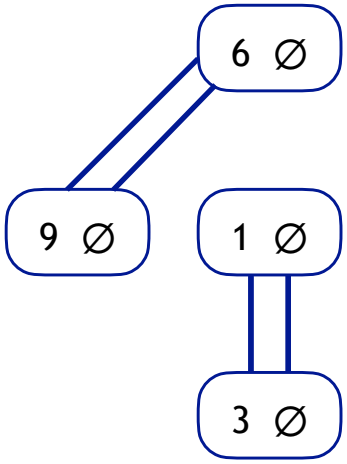
Clique Description

y	2	6	2	2	0	6	0	0	0	0
	1	2	3	4	5	6	7	8	9	10
1		42	4	6	14	8	34	58	68	64
2	42		68	36	26	62	50	6	52	24
3	4	68		44	30	26	4	32	60	40
4	6	36	44		2	60	42	44	30	36
5	14	26	30	2		8	42	34	44	62
6	8	62	26	60	8		52	70	6	48
7	34	50	4	42	42	52		38	38	50
8	58	6	32	44	34	70	38		24	32
9	68	52	60	30	44	6	38	24		10
10	64	24	40	36	62	48	50	32	10	



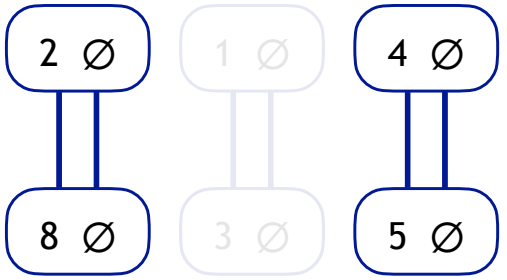
DUAL VARIABLES UPDATE

Blossom Algorithm: Example



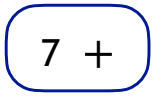
Clique Description

y	2	6	2	2	0	6	0	0	0	0
	1	2	3	4	5	6	7	8	9	10
1		42	4	6	14	8	34	58	68	64
2	42		68	36	26	62	50	6	52	24
3	4	68		44	30	26	4	32	60	40
4	6	36	44		2	60	42	44	30	36
5	14	26	30	2		8	42	34	44	62
6	8	62	26	60	8		52	70	6	48
7	34	50	4	42	42	52		38	38	50
8	58	6	32	44	34	70	38		24	32
9	68	52	60	30	44	6	38	24		10
10	64	24	40	36	62	48	50	32	10	



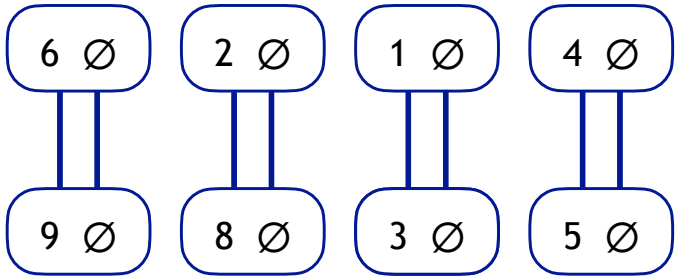
AUGMENT

Blossom Algorithm: Example

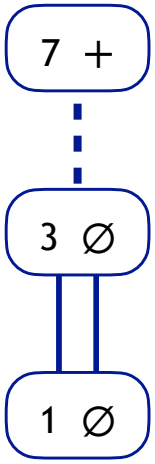


Clique Description

y	2	6	2	2	0	6	0	0	0	0
	1	2	3	4	5	6	7	8	9	10
1		42	4	6	14	8	34	58	68	64
2	42		68	36	26	62	50	6	52	24
3	4	68		44	30	26	4	32	60	40
4	6	36	44		2	60	42	44	30	36
5	14	26	30	2		8	42	34	44	62
6	8	62	26	60	8		52	70	6	48
7	34	50	4	42	42	52		38	38	50
8	58	6	32	44	34	70	38		24	32
9	68	52	60	30	44	6	38	24		10
10	64	24	40	36	62	48	50	32	10	

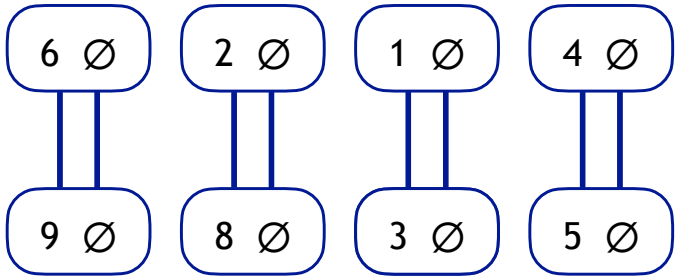


Blossom Algorithm: Example

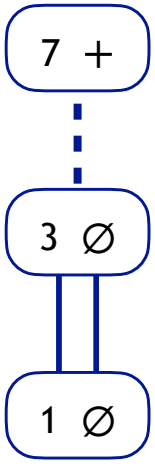


Clique Description

y	2	6	2	2	0	6	0	0	0	0
	1	2	3	4	5	6	7	8	9	10
1		42	4	6	14	8	34	58	68	64
2	42		68	36	26	62	50	6	52	24
3	4	68		44	30	26	4	32	60	40
4	6	36	44		2	60	42	44	30	36
5	14	26	30	2		8	42	34	44	62
6	8	62	26	60	8		52	70	6	48
7	34	50	4	42	42	52		38	38	50
8	58	6	32	44	34	70	38		24	32
9	68	52	60	30	44	6	38	24		10
10	64	24	40	36	62	48	50	32	10	

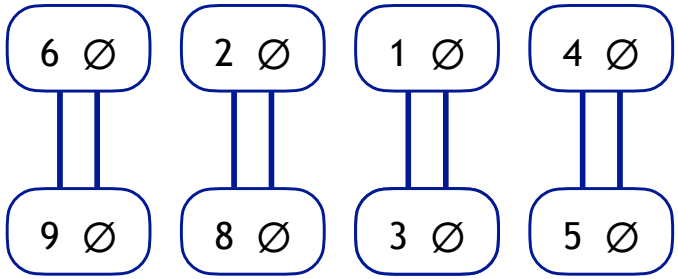


Blossom Algorithm: Example



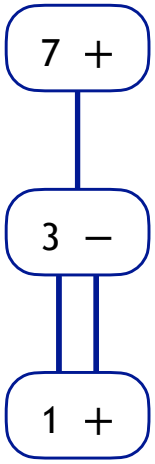
Clique Description

y	2	6	2	2	0	6	2	0	0	0
	1	2	3	4	5	6	7	8	9	10
1		42	4	6	14	8	34	58	68	64
2	42		68	36	26	62	50	6	52	24
3	4	68		44	30	26	4	32	60	40
4	6	36	44		2	60	42	44	30	36
5	14	26	30	2		8	42	34	44	62
6	8	62	26	60	8		52	70	6	48
7	34	50	4	42	42	52		38	38	50
8	58	6	32	44	34	70	38		24	32
9	68	52	60	30	44	6	38	24		10
10	64	24	40	36	62	48	50	32	10	



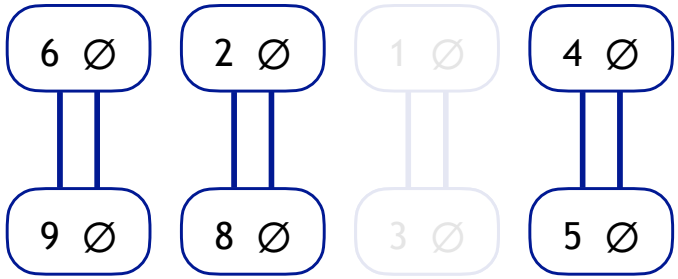
DUAL VARIABLES UPDATE

Blossom Algorithm: Example



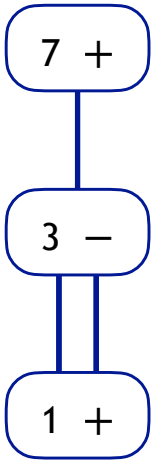
Clique Description

y	2	6	2	2	0	6	2	0	0	0
	1	2	3	4	5	6	7	8	9	10
1		42	4	6	14	8	34	58	68	64
2	42		68	36	26	62	50	6	52	24
3	4	68		44	30	26	4	32	60	40
4	6	36	44		2	60	42	44	30	36
5	14	26	30	2		8	42	34	44	62
6	8	62	26	60	8		52	70	6	48
7	34	50	4	42	42	52		38	38	50
8	58	6	32	44	34	70	38		24	32
9	68	52	60	30	44	6	38	24		10
10	64	24	40	36	62	48	50	32	10	



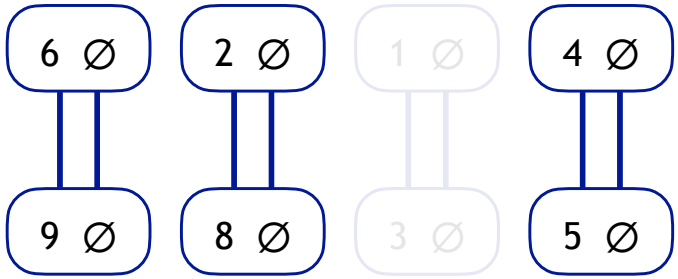
GROW

Blossom Algorithm: Example

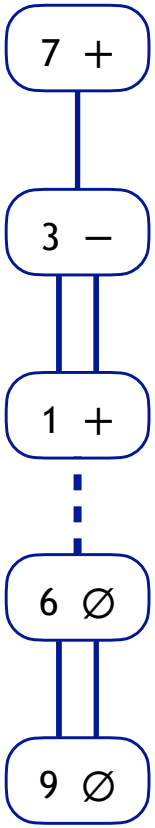


Clique Description

y	2	6	2	2	0	6	2	0	0	0
	1	2	3	4	5	6	7	8	9	10
1		42	4	6	14	8	34	58	68	64
2	42		68	36	26	62	50	6	52	24
3	4	68		44	30	26	4	32	60	40
4	6	36	44		2	60	42	44	30	36
5	14	26	30	2		8	42	34	44	62
6	8	62	26	60	8		52	70	6	48
7	34	50	4	42	42	52		38	38	50
8	58	6	32	44	34	70	38		24	32
9	68	52	60	30	44	6	38	24		10
10	64	24	40	36	62	48	50	32	10	

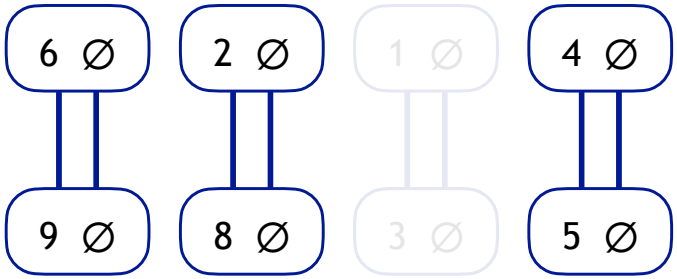


Blossom Algorithm: Example

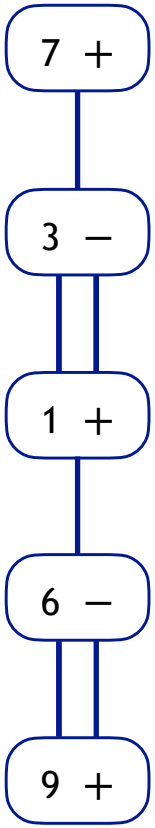


Clique Description

y	2	6	2	2	0	6	2	0	0	0
	1	2	3	4	5	6	7	8	9	10
1		42	4	6	14	8	34	58	68	64
2	42		68	36	26	62	50	6	52	24
3	4	68		44	30	26	4	32	60	40
4	6	36	44		2	60	42	44	30	36
5	14	26	30	2		8	42	34	44	62
6	8	62	26	60	8		52	70	6	48
7	34	50	4	42	42	52		38	38	50
8	58	6	32	44	34	70	38		24	32
9	68	52	60	30	44	6	38	24		10
10	64	24	40	36	62	48	50	32	10	

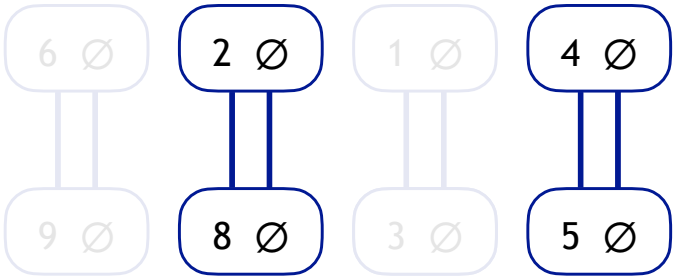


Blossom Algorithm: Example



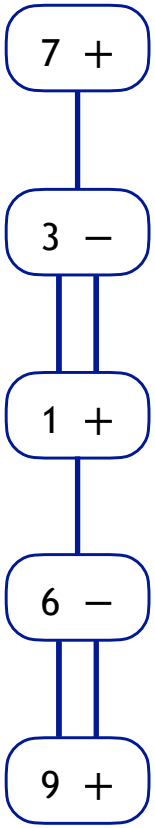
Clique Description

y	2	6	2	2	0	6	2	0	0	0
	1	2	3	4	5	6	7	8	9	10
1		42	4	6	14	8	34	58	68	64
2	42		68	36	26	62	50	6	52	24
3	4	68		44	30	26	4	32	60	40
4	6	36	44		2	60	42	44	30	36
5	14	26	30	2		8	42	34	44	62
6	8	62	26	60	8		52	70	6	48
7	34	50	4	42	42	52		38	38	50
8	58	6	32	44	34	70	38		24	32
9	68	52	60	30	44	6	38	24		10
10	64	24	40	36	62	48	50	32	10	



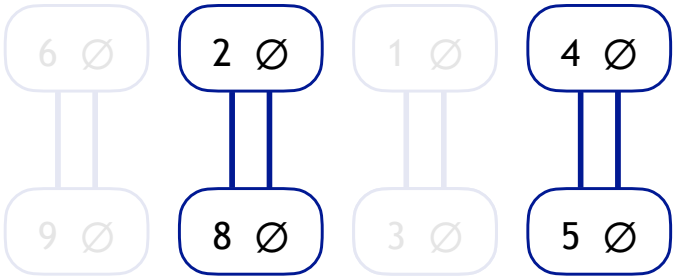
GROW

Blossom Algorithm: Example

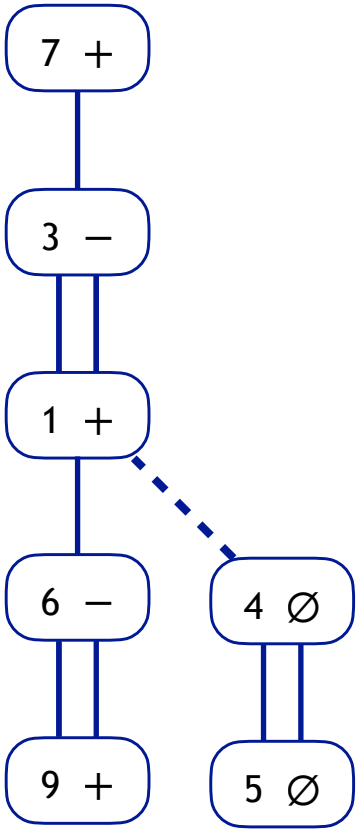


Clique Description

y	2	6	2	2	0	6	2	0	0	0
	1	2	3	4	5	6	7	8	9	10
1		42	4	6	14	8	34	58	68	64
2	42		68	36	26	62	50	6	52	24
3	4	68		44	30	26	4	32	60	40
4	6	36	44		2	60	42	44	30	36
5	14	26	30	2		8	42	34	44	62
6	8	62	26	60	8		52	70	6	48
7	34	50	4	42	42	52		38	38	50
8	58	6	32	44	34	70	38		24	32
9	68	52	60	30	44	6	38	24		10
10	64	24	40	36	62	48	50	32	10	

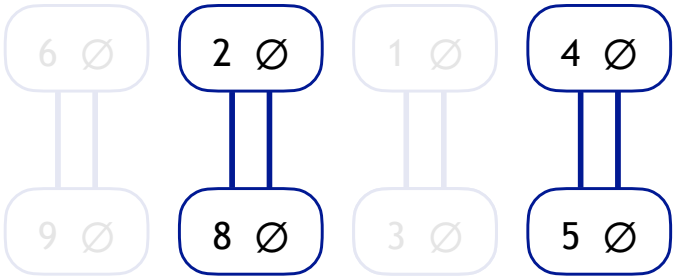


Blossom Algorithm: Example

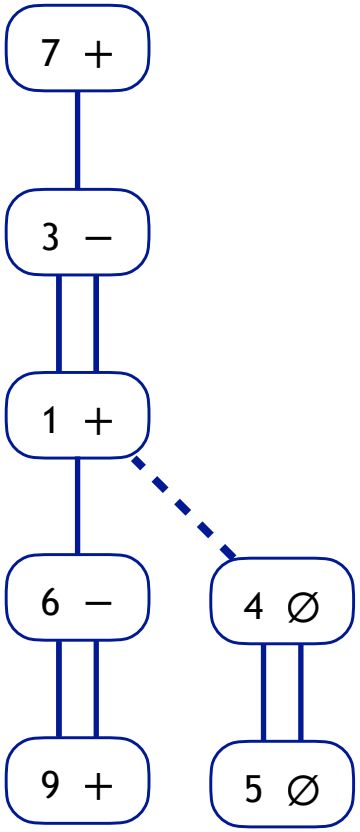


Clique Description

y	2	6	2	2	0	6	2	0	0	0
	1	2	3	4	5	6	7	8	9	10
1		42	4	6	14	8	34	58	68	64
2	42		68	36	26	62	50	6	52	24
3	4	68		44	30	26	4	32	60	40
4	6	36	44		2	60	42	44	30	36
5	14	26	30	2		8	42	34	44	62
6	8	62	26	60	8		52	70	6	48
7	34	50	4	42	42	52		38	38	50
8	58	6	32	44	34	70	38		24	32
9	68	52	60	30	44	6	38	24		10
10	64	24	40	36	62	48	50	32	10	

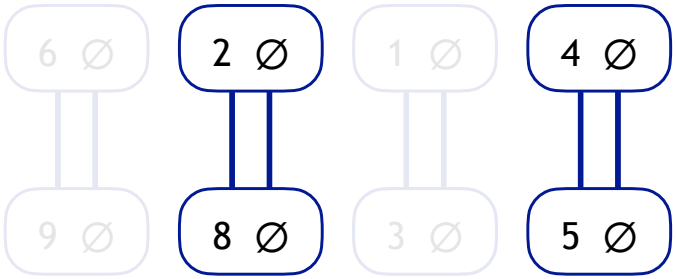


Blossom Algorithm: Example



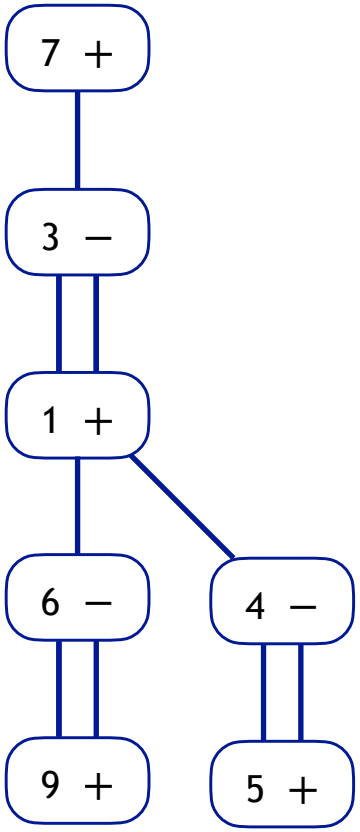
Clique Description

y	4	6	0	2	0	4	4	0	2	0
	1	2	3	4	5	6	7	8	9	10
1		42	4	6	14	8	34	58	68	64
2	42		68	36	26	62	50	6	52	24
3	4	68		44	30	26	4	32	60	40
4	6	36	44		2	60	42	44	30	36
5	14	26	30	2		8	42	34	44	62
6	8	62	26	60	8		52	70	6	48
7	34	50	4	42	42	52		38	38	50
8	58	6	32	44	34	70	38		24	32
9	68	52	60	30	44	6	38	24		10
10	64	24	40	36	62	48	50	32	10	



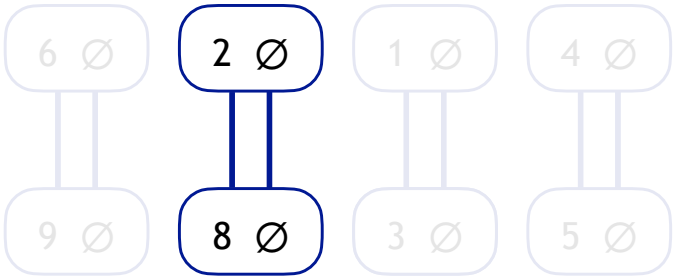
DUAL VARIABLES UPDATE

Blossom Algorithm: Example



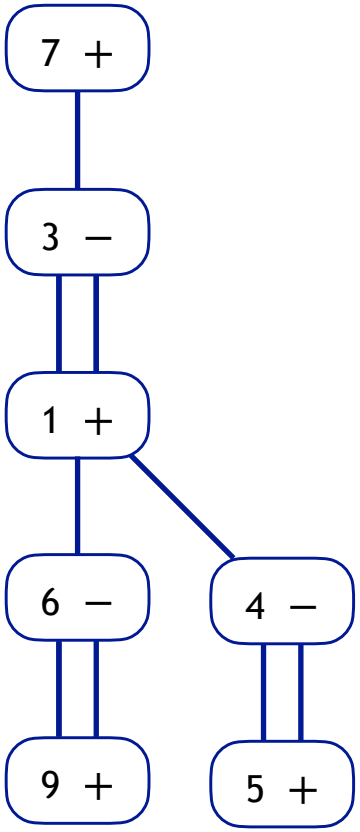
Clique Description

y	4	6	0	2	0	4	4	0	2	0
	1	2	3	4	5	6	7	8	9	10
1		42	4	6	14	8	34	58	68	64
2	42		68	36	26	62	50	6	52	24
3	4	68		44	30	26	4	32	60	40
4	6	36	44		2	60	42	44	30	36
5	14	26	30	2		8	42	34	44	62
6	8	62	26	60	8		52	70	6	48
7	34	50	4	42	42	52		38	38	50
8	58	6	32	44	34	70	38		24	32
9	68	52	60	30	44	6	38	24		10
10	64	24	40	36	62	48	50	32	10	



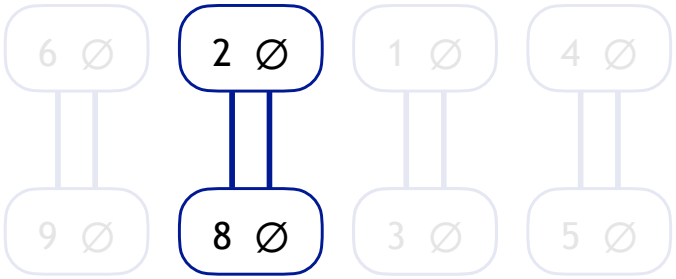
GROW

Blossom Algorithm: Example

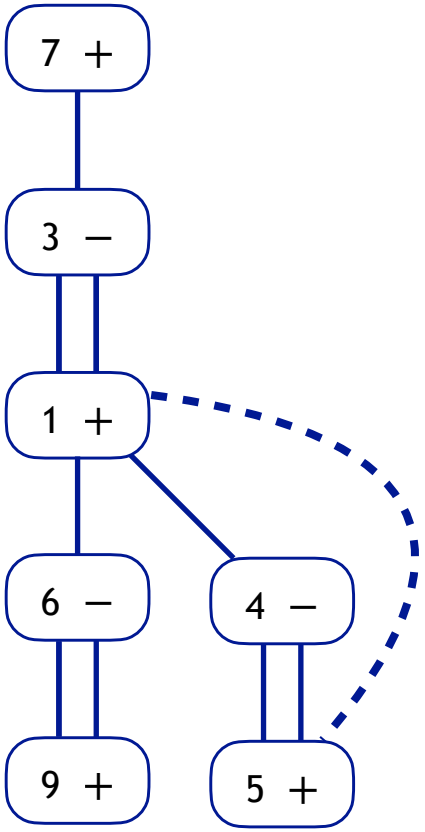


Clique Description

y	4	6	0	2	0	4	4	0	2	0
	1	2	3	4	5	6	7	8	9	10
1		42	4	6	14	8	34	58	68	64
2	42		68	36	26	62	50	6	52	24
3	4	68		44	30	26	4	32	60	40
4	6	36	44		2	60	42	44	30	36
5	14	26	30	2		8	42	34	44	62
6	8	62	26	60	8		52	70	6	48
7	34	50	4	42	42	52		38	38	50
8	58	6	32	44	34	70	38		24	32
9	68	52	60	30	44	6	38	24		10
10	64	24	40	36	62	48	50	32	10	

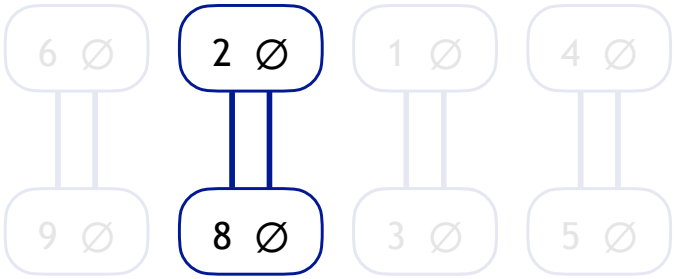


Blossom Algorithm: Example

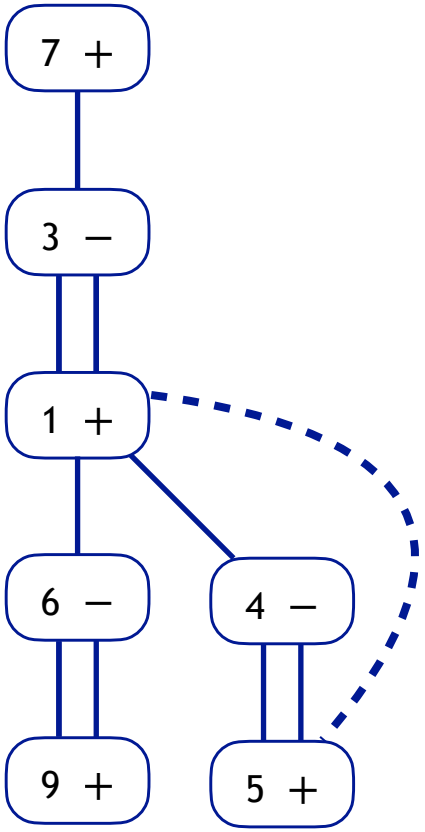


Clique Description

y	4	6	0	2	0	4	4	0	2	0
	1	2	3	4	5	6	7	8	9	10
1		42	4	6	14	8	34	58	68	64
2	42		68	36	26	62	50	6	52	24
3	4	68		44	30	26	4	32	60	40
4	6	36	44		2	60	42	44	30	36
5	14	26	30	2		8	42	34	44	62
6	8	62	26	60	8		52	70	6	48
7	34	50	4	42	42	52		38	38	50
8	58	6	32	44	34	70	38		24	32
9	68	52	60	30	44	6	38	24		10
10	64	24	40	36	62	48	50	32	10	

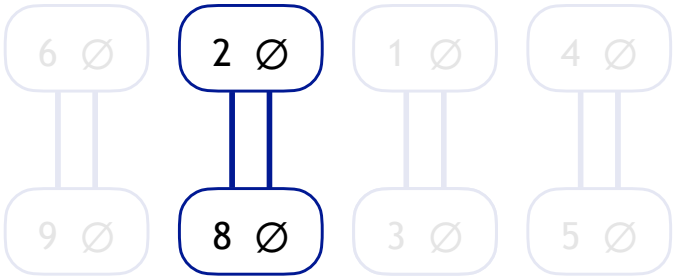


Blossom Algorithm: Example



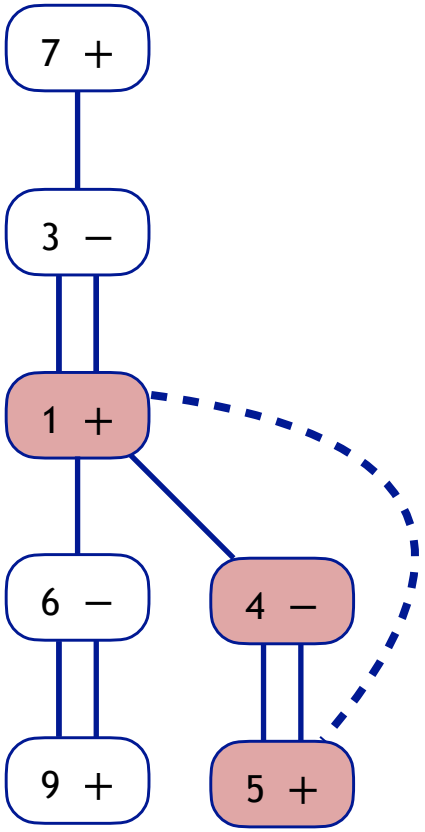
Clique Description

y	9	6	-5	-3	5	-1	9	0	7	0
	1	2	3	4	5	6	7	8	9	10
1		42	4	6	14	8	34	58	68	64
2	42		68	36	26	62	50	6	52	24
3	4	68		44	30	26	4	32	60	40
4	6	36	44		2	60	42	44	30	36
5	14	26	30	2		8	42	34	44	62
6	8	62	26	60	8		52	70	6	48
7	34	50	4	42	42	52		38	38	50
8	58	6	32	44	34	70	38		24	32
9	68	52	60	30	44	6	38	24		10
10	64	24	40	36	62	48	50	32	10	



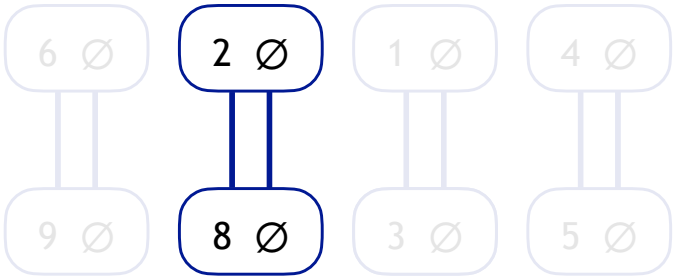
DUAL VARIABLES UPDATE

Blossom Algorithm: Example



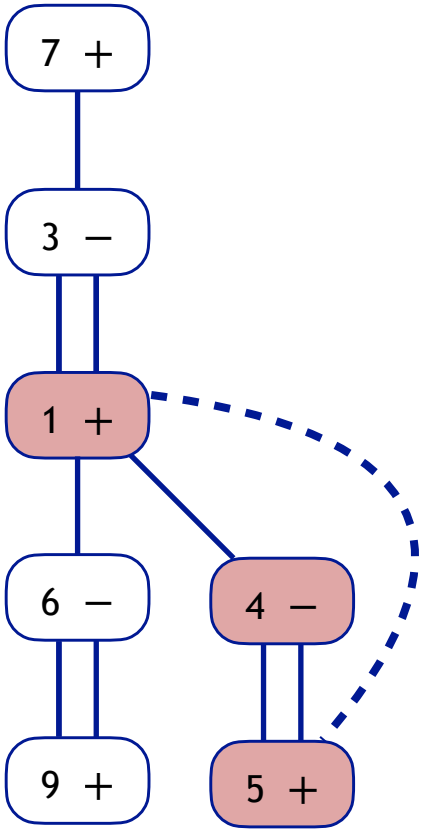
Clique Description

y	9	6	-5	-3	5	-1	9	0	7	0
	1	2	3	4	5	6	7	8	9	10
1		42	4	6	14	8	34	58	68	64
2	42		68	36	26	62	50	6	52	24
3	4	68		44	30	26	4	32	60	40
4	6	36	44		2	60	42	44	30	36
5	14	26	30	2		8	42	34	44	62
6	8	62	26	60	8		52	70	6	48
7	34	50	4	42	42	52		38	38	50
8	58	6	32	44	34	70	38		24	32
9	68	52	60	30	44	6	38	24		10
10	64	24	40	36	62	48	50	32	10	



SHRINK

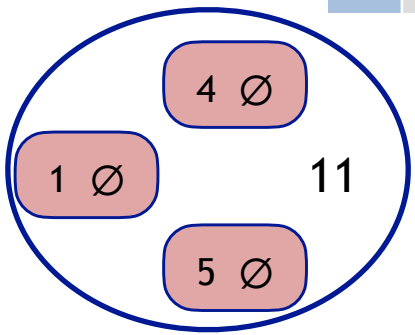
Blossom Algorithm: Example

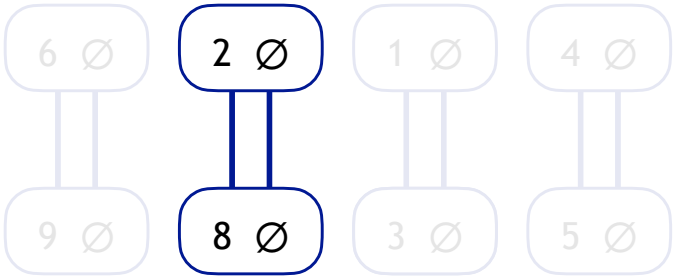


Clique Description

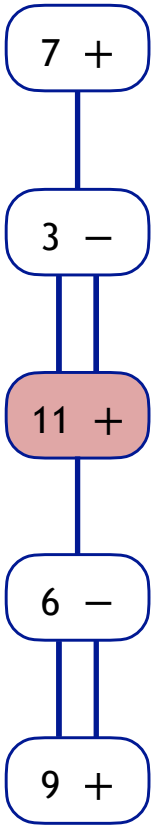
y	9	6	-5	-3	5	-1	9	0	7	0	0
	1	2	3	4	5	6	7	8	9	10	11
1		42	4	6	14	8	34	58	68	64	
2	42		68	36	26	62	50	6	52	24	
3	4	68		44	30	26	4	32	60	40	
4	6	36	44		2	60	42	44	30	36	
5	14	26	30	2		8	42	34	44	62	
6	8	62	26	60	8		52	70	6	48	
7	34	50	4	42	42	52		38	38	50	
8	58	6	32	44	34	70	38		24	32	
9	68	52	60	30	44	6	38	24		10	
10	64	24	40	36	62	48	50	32	10		
11											

SHRINK





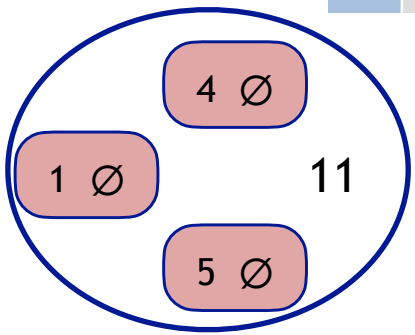
Blossom Algorithm: Example

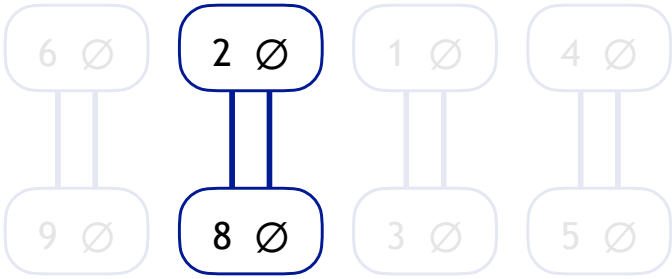


Clique Description

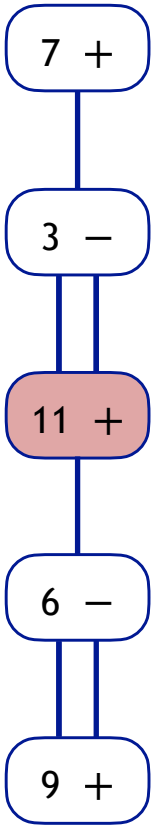
y	9	6	-5	-3	5	-1	9	0	7	0	0
	1	2	3	4	5	6	7	8	9	10	11
1		42	4	6	14	8	34	58	68	64	
2	42		68	36	26	62	50	6	52	24	
3	4	68		44	30	26	4	32	60	40	
4	6	36	44		2	60	42	44	30	36	
5	14	26	30	2		8	42	34	44	62	
6	8	62	26	60	8		52	70	6	48	
7	34	50	4	42	42	52		38	38	50	
8	58	6	32	44	34	70	38		24	32	
9	68	52	60	30	44	6	38	24		10	
10	64	24	40	36	62	48	50	32	10		
11											

SHRINK





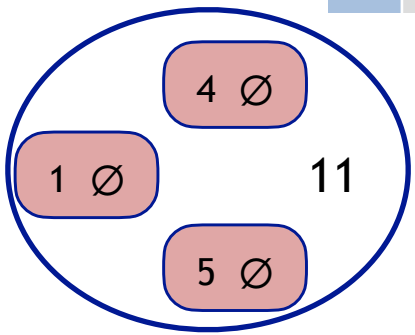
Blossom Algorithm: Example

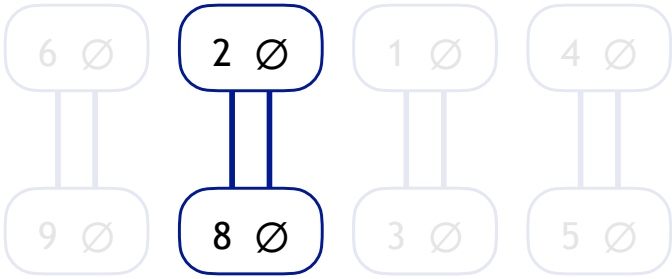


Clique Description

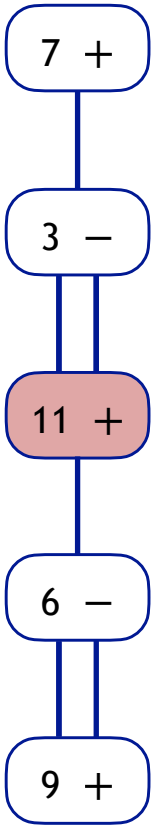
y	9	6	-5	-3	5	-1	9	0	7	0	0
	1	2	3	4	5	6	7	8	9	10	11
1		42	4	6	14	8	34	58	68	64	
2	42		68	36	26	62	50	6	52	24	21
3	4	68		44	30	26	4	32	60	40	-5
4	6	36	44		2	60	42	44	30	36	
5	14	26	30	2		8	42	34	44	62	
6	8	62	26	60	8		52	70	6	48	-1
7	34	50	4	42	42	52		38	38	50	25
8	58	6	32	44	34	70	38		24	32	29
9	68	52	60	30	44	6	38	24		10	33
10	64	24	40	36	62	48	50	32	10		39
11											

SHRINK





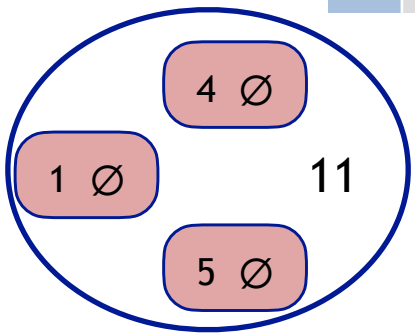
Blossom Algorithm: Example

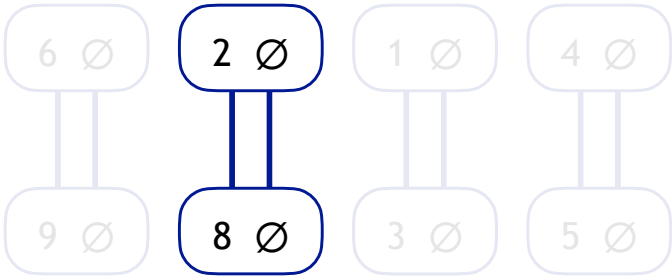


Clique Description

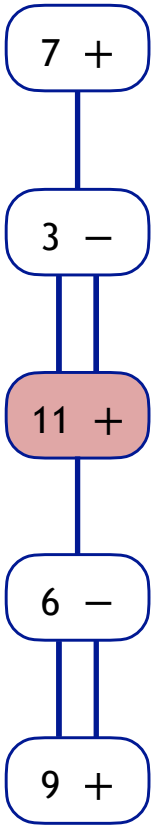
y	9	6	-5	-3	5	-1	9	0	7	0	0
	1	2	3	4	5	6	7	8	9	10	11
1		42	4	6	14	8	34	58	68	64	
2	42		68	36	26	62	50	6	52	24	21
3	4	68		44	30	26	4	32	60	40	-5
4	6	36	44		2	60	42	44	30	36	
5	14	26	30	2		8	42	34	44	62	
6	8	62	26	60	8		52	70	6	48	-1
7	34	50	4	42	42	52		38	38	50	25
8	58	6	32	44	34	70	38		24	32	29
9	68	52	60	30	44	6	38	24		10	33
10	64	24	40	36	62	48	50	32	10		39
11		21	-5			-1	25	29	33	39	

SHRINK





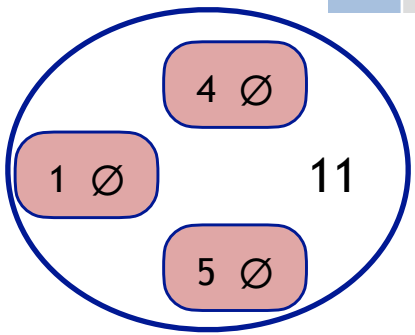
Blossom Algorithm: Example

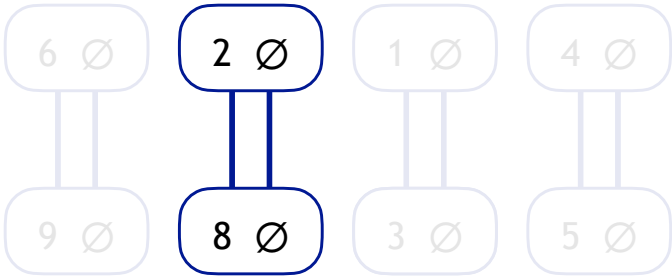


Clique Description

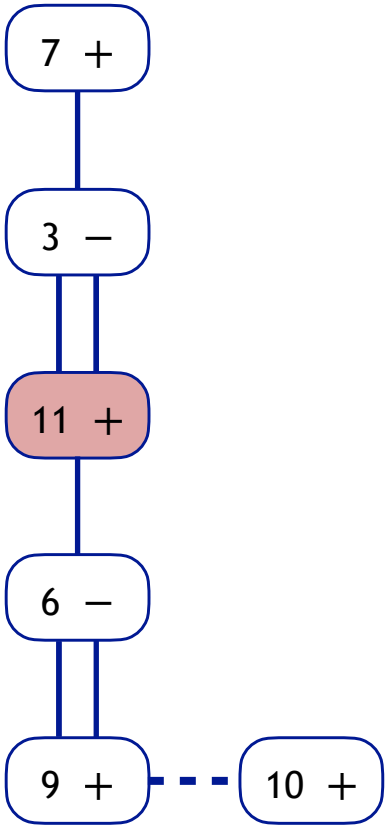
y	9	6	-5	-3	5	-1	9	0	7	0	0
	1	2	3	4	5	6	7	8	9	10	11
1		42	4	6	14	8	34	58	68	64	
2	42		68	36	26	62	50	6	52	24	21
3	4	68		44	30	26	4	32	60	40	-5
4	6	36	44		2	60	42	44	30	36	
5	14	26	30	2		8	42	34	44	62	
6	8	62	26	60	8		52	70	6	48	-1
7	34	50	4	42	42	52		38	38	50	25
8	58	6	32	44	34	70	38		24	32	29
9	68	52	60	30	44	6	38	24		10	33
10	64	24	40	36	62	48	50	32	10		39
11		21	-5			-1	25	29	33	39	

SHRINK





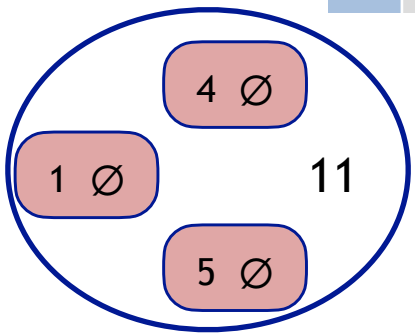
Blossom Algorithm: Example

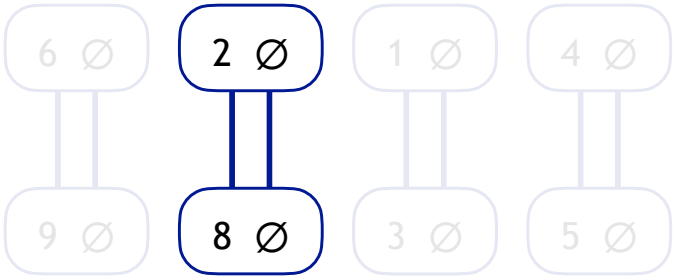


Clique Description

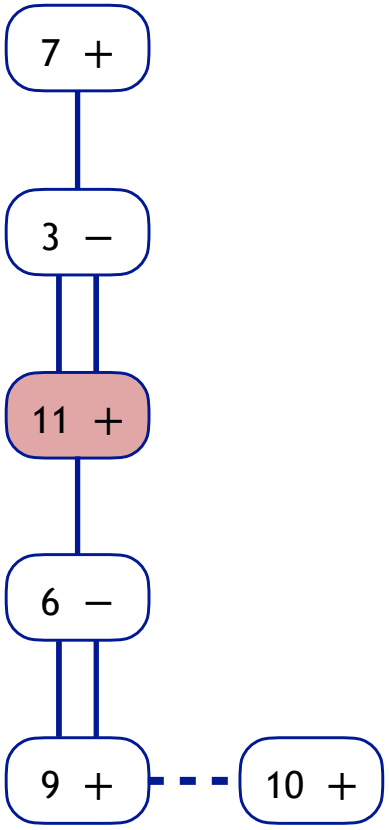
y	9	6	-5	-3	5	-1	9	0	7	0	0
	1	2	3	4	5	6	7	8	9	10	11
1		42	4	6	14	8	34	58	68	64	
2	42		68	36	26	62	50	6	52	24	21
3	4	68		44	30	26	4	32	60	40	-5
4	6	36	44		2	60	42	44	30	36	
5	14	26	30	2		8	42	34	44	62	
6	8	62	26	60	8		52	70	6	48	-1
7	34	50	4	42	42	52		38	38	50	25
8	58	6	32	44	34	70	38		24	32	29
9	68	52	60	30	44	6	38	24		10	33
10	64	24	40	36	62	48	50	32	10		39
11		21	-5			-1	25	29	33	39	

SHRINK



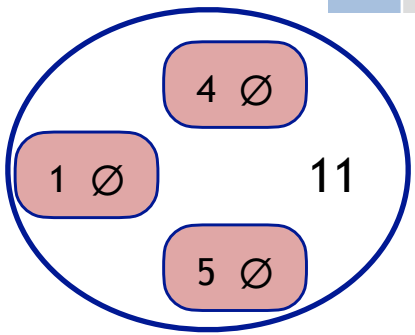


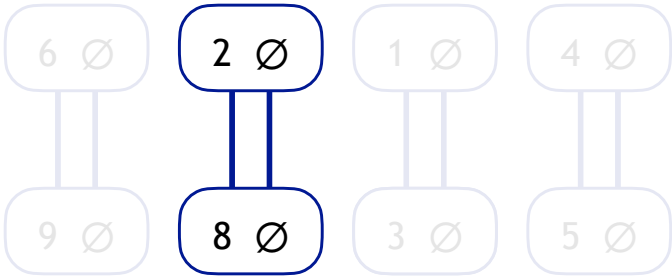
Blossom Algorithm: Example



Clique Description

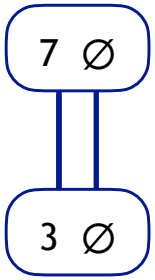
y	9	6	-8	-3	5	-4	12	0	10	0	3
	1	2	3	4	5	6	7	8	9	10	11
1		42	4	6	14	8	34	58	68	64	
2	42		68	36	26	62	50	6	52	24	21
3	4	68		44	30	26	4	32	60	40	-5
4	6	36	44		2	60	42	44	30	36	
5	14	26	30	2		8	42	34	44	62	
6	8	62	26	60	8		52	70	6	48	-1
7	34	50	4	42	42	52		38	38	50	25
8	58	6	32	44	34	70	38		24	32	29
9	68	52	60	30	44	6	38	24		10	33
10	64	24	40	36	62	48	50	32	10		39
11		21	-5			-1	25	29	33	39	

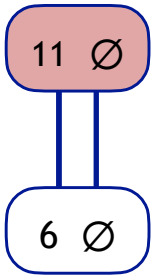


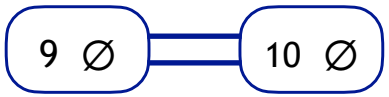


DUAL VARIABLES UPDATE

Blossom Algorithm: Example

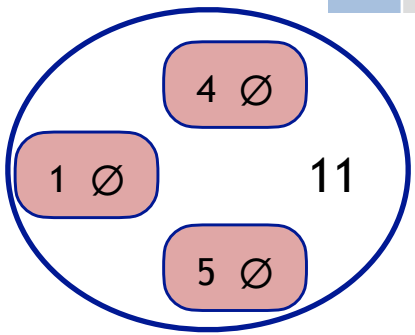


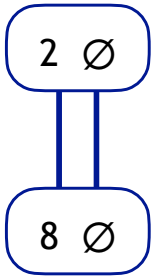




Clique Description

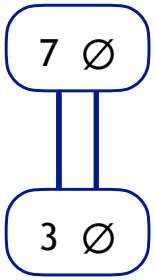
y	9	6	-8	-3	5	-4	12	0	10	0	3
	1	2	3	4	5	6	7	8	9	10	11
1		42	4	6	14	8	34	58	68	64	
2	42		68	36	26	62	50	6	52	24	21
3	4	68		44	30	26	4	32	60	40	-5
4	6	36	44		2	60	42	44	30	36	
5	14	26	30	2		8	42	34	44	62	
6	8	62	26	60	8		52	70	6	48	-1
7	34	50	4	42	42	52		38	38	50	25
8	58	6	32	44	34	70	38		24	32	29
9	68	52	60	30	44	6	38	24		10	33
10	64	24	40	36	62	48	50	32	10		39
11		21	-5			-1	25	29	33	39	

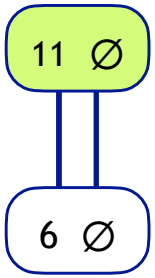


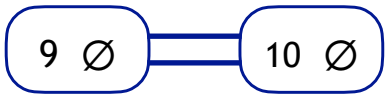


AUGMENT

Blossom Algorithm: Example

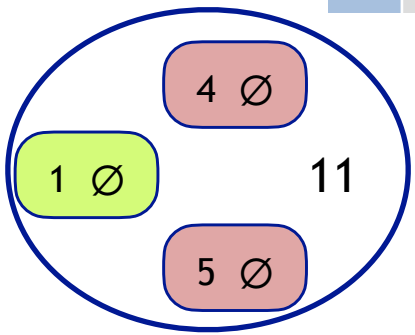


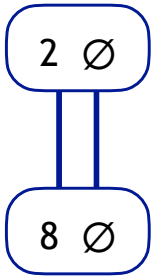




Clique Description

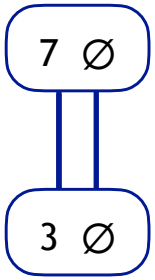
y	9	6	-8	-3	5	-4	12	0	10	0	3
	1	2	3	4	5	6	7	8	9	10	11
1		42	4	6	14	8	34	58	68	64	
2	42		68	36	26	62	50	6	52	24	21
3	4	68		44	30	26	4	32	60	40	-5
4	6	36	44		2	60	42	44	30	36	
5	14	26	30	2		8	42	34	44	62	
6	8	62	26	60	8		52	70	6	48	-1
7	34	50	4	42	42	52		38	38	50	25
8	58	6	32	44	34	70	38		24	32	29
9	68	52	60	30	44	6	38	24		10	33
10	64	24	40	36	62	48	50	32	10		39
11		21	-5			-1	25	29	33	39	

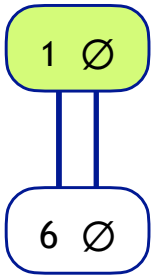


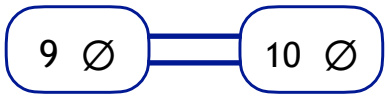


EXPAND

Blossom Algorithm: Example



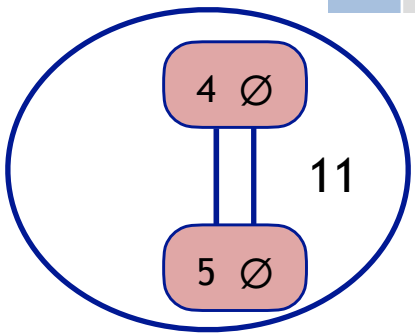


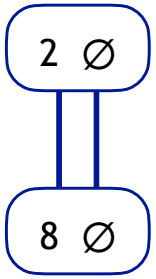


Clique Description

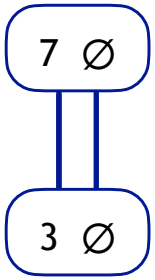
y	9	6	-8	-3	5	-4	12	0	10	0	3
	1	2	3	4	5	6	7	8	9	10	11
1		42	4	6	14	8	34	58	68	64	
2	42		68	36	26	62	50	6	52	24	21
3	4	68		44	30	26	4	32	60	40	-5
4	6	36	44		2	60	42	44	30	36	
5	14	26	30	2		8	42	34	44	62	
6	8	62	26	60	8		52	70	6	48	-1
7	34	50	4	42	42	52		38	38	50	25
8	58	6	32	44	34	70	38		24	32	29
9	68	52	60	30	44	6	38	24		10	33
10	64	24	40	36	62	48	50	32	10		39
11		21	-5			-1	25	29	33	39	

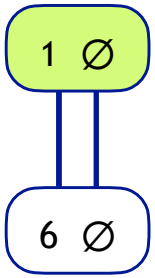
EXPAND

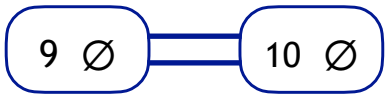




Blossom Algorithm: Example

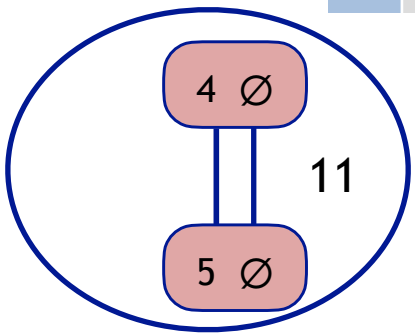


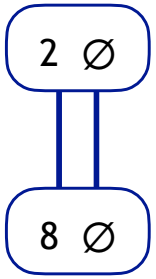




Clique Description

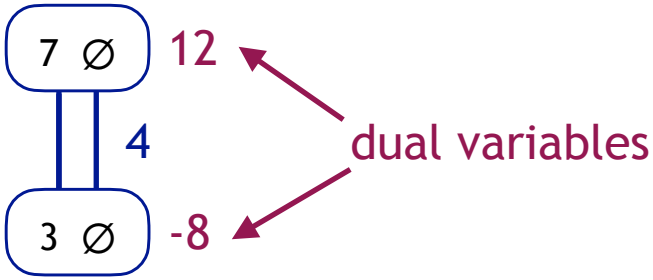
y	9	6	-8	-3	5	-4	12	0	10	0	3
	1	2	3	4	5	6	7	8	9	10	11
1		42	4	6	14	8	34	58	68	64	
2	42		68	36	26	62	50	6	52	24	21
3	4	68		44	30	26	4	32	60	40	-5
4	6	36	44		2	60	42	44	30	36	
5	14	26	30	2		8	42	34	44	62	
6	8	62	26	60	8		52	70	6	48	-1
7	34	50	4	42	42	52		38	38	50	25
8	58	6	32	44	34	70	38		24	32	29
9	68	52	60	30	44	6	38	24		10	33
10	64	24	40	36	62	48	50	32	10		39
11		21	-5			-1	25	29	33	39	

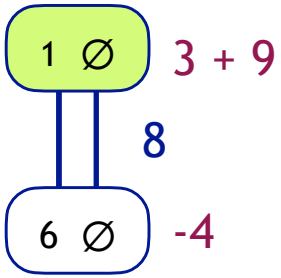


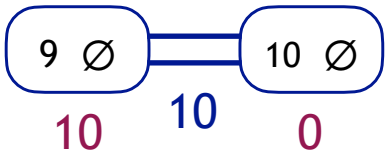


OPTIMAL SOLUTION

Blossom Algorithm: Example

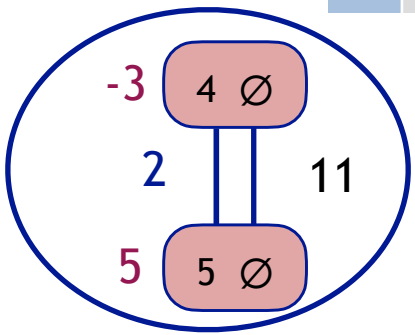


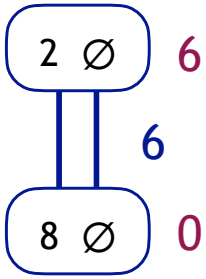




Clique Description

y	9	6	-8	-3	5	-4	12	0	10	0	3
	1	2	3	4	5	6	7	8	9	10	11
1		42	4	6	14	8	34	58	68	64	
2	42		68	36	26	62	50	6	52	24	21
3	4	68		44	30	26	4	32	60	40	-5
4	6	36	44		2	60	42	44	30	36	
5	14	26	30	2		8	42	34	44	62	
6	8	62	26	60	8		52	70	6	48	-1
7	34	50	4	42	42	52		38	38	50	25
8	58	6	32	44	34	70	38		24	32	29
9	68	52	60	30	44	6	38	24		10	33
10	64	24	40	36	62	48	50	32	10		39
11		21	-5			-1	25	29	33	39	





OPTIMAL SOLUTION

Blossom Algorithm

Why the algorithm runs in polynomial time?

1. The augmentation increases the size of matching.
2. After the SHRINK the blossom vertex is +.

Blossom Algorithm

Why the algorithm runs in polynomial time?

1. The augmentation increases the size of matching.
2. After the SHRINK the blossom vertex is +.
3. The EXPAND can be applied only to a — vertex.

Blossom Algorithm

Why the algorithm runs in polynomial time?

1. The augmentation increases the size of matching.
2. After the SHRINK the blossom vertex is +.
3. The EXPAND can be applied only to a − vertex.
4. The vertex changes from + to − only after augmentation.

Blossom Algorithm

Why the algorithm runs in polynomial time?

1. The augmentation increases the size of matching.
2. After the SHRINK the blossom vertex is $+$.
3. The EXPAND can be applied only to a $-$ vertex.
4. The vertex changes from $+$ to $-$ only after augmentation.
5. Between two augmentations at most $O(n)$ other operations.

Blossom Algorithm

Why the algorithm runs in polynomial time?

1. The augmentation increases the size of matching.
2. After the SHRINK the blossom vertex is $+$.
3. The EXPAND can be applied only to a $-$ vertex.
4. The vertex changes from $+$ to $-$ only after augmentation.
5. Between two augmentations at most $O(n)$ other operations.

Why the obtained solution is optimal?

1. The blossoms can be changed back into a matching.
2. The value of the dual is equal to the value of primal.

Practice vs Theory

Problem:
Clearing a cycle of size ℓ requires 2ℓ operating theatres and 2ℓ surgical teams available at the same time.

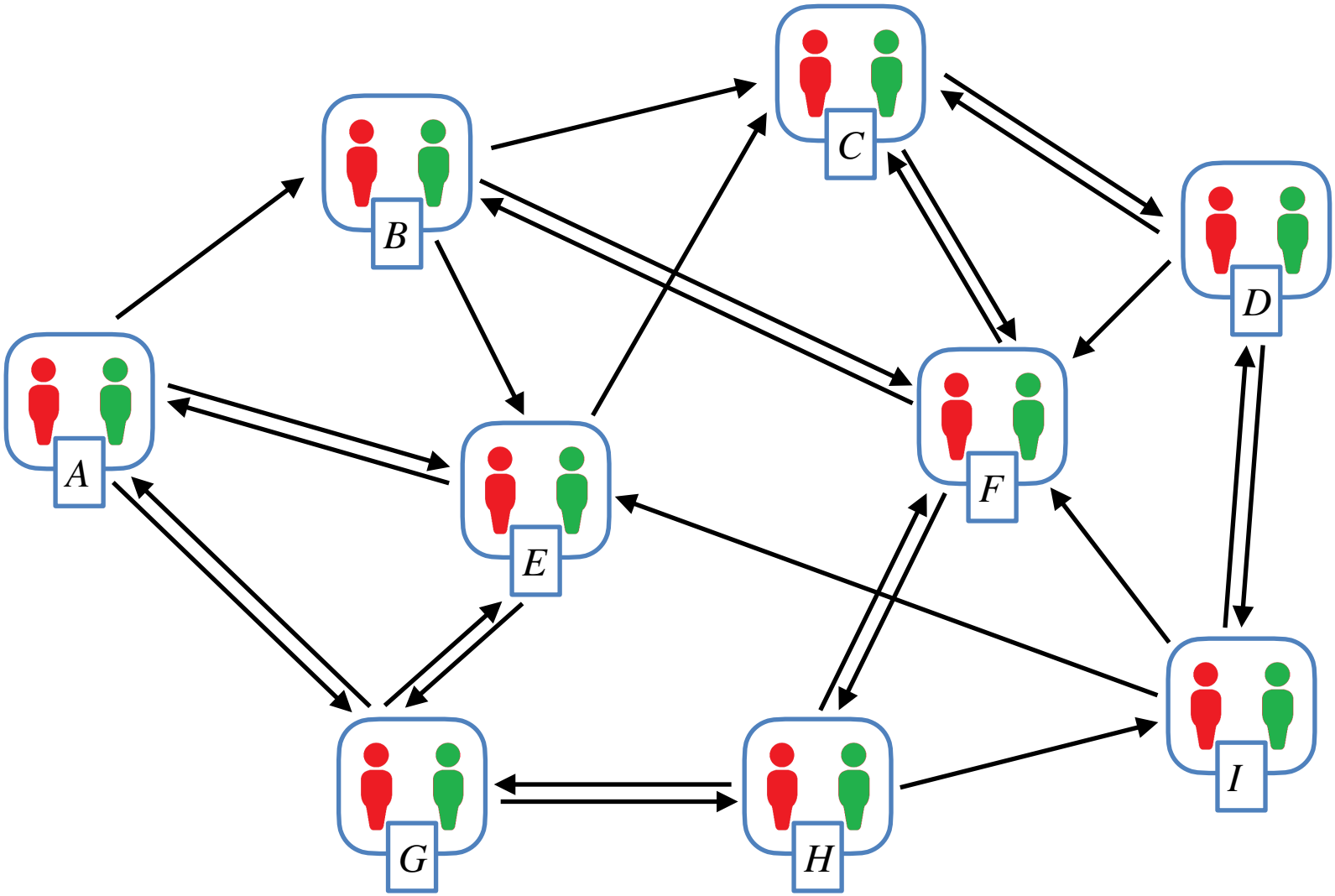
Solution:
Use only small cycles.

Matching Theory!

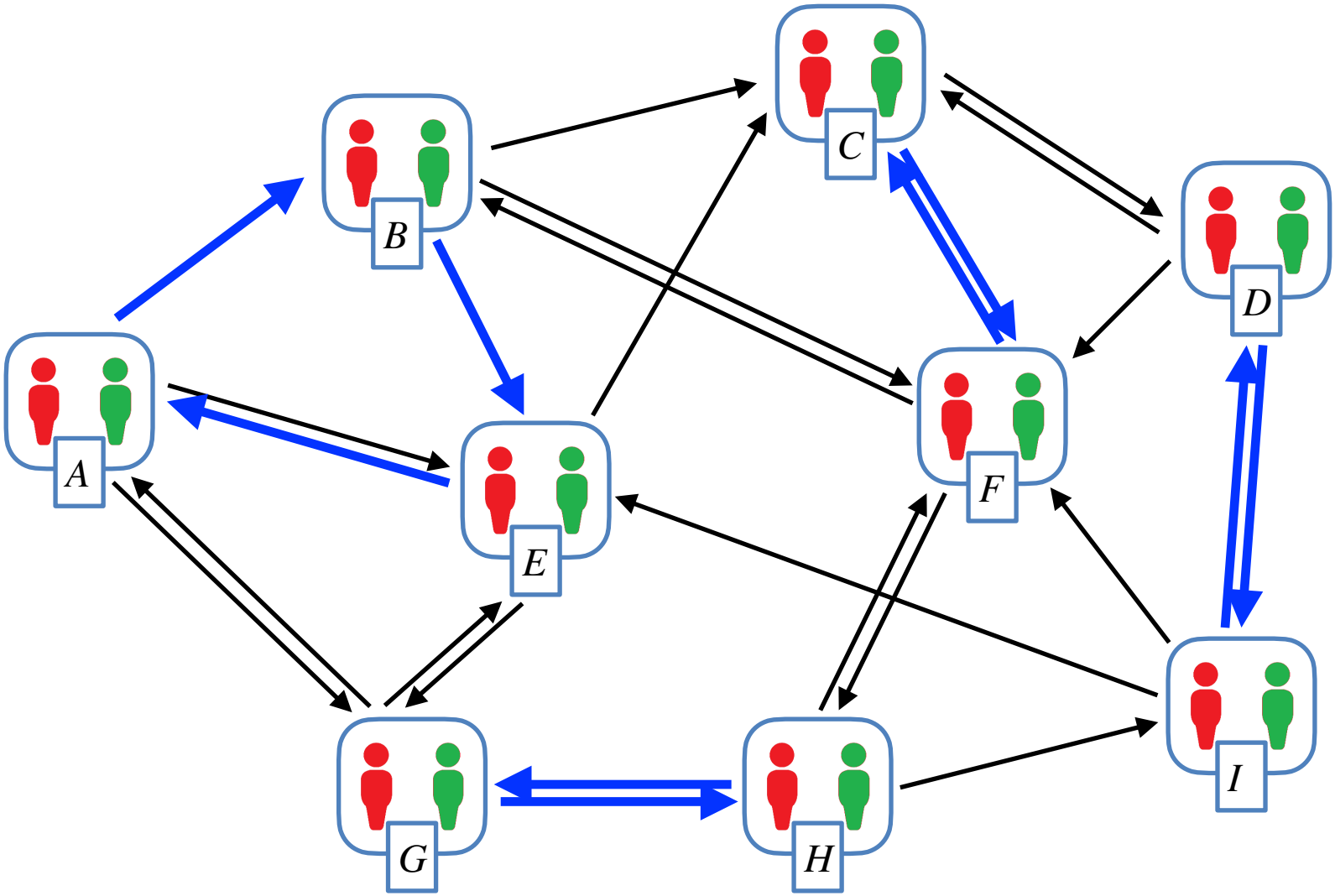
Properties:
Maximum-cardinality matchings can be found in polynomial time. How to make it incentive compatible? (Using **max-weight-matching.**)

Next step: cycles of size at most 3

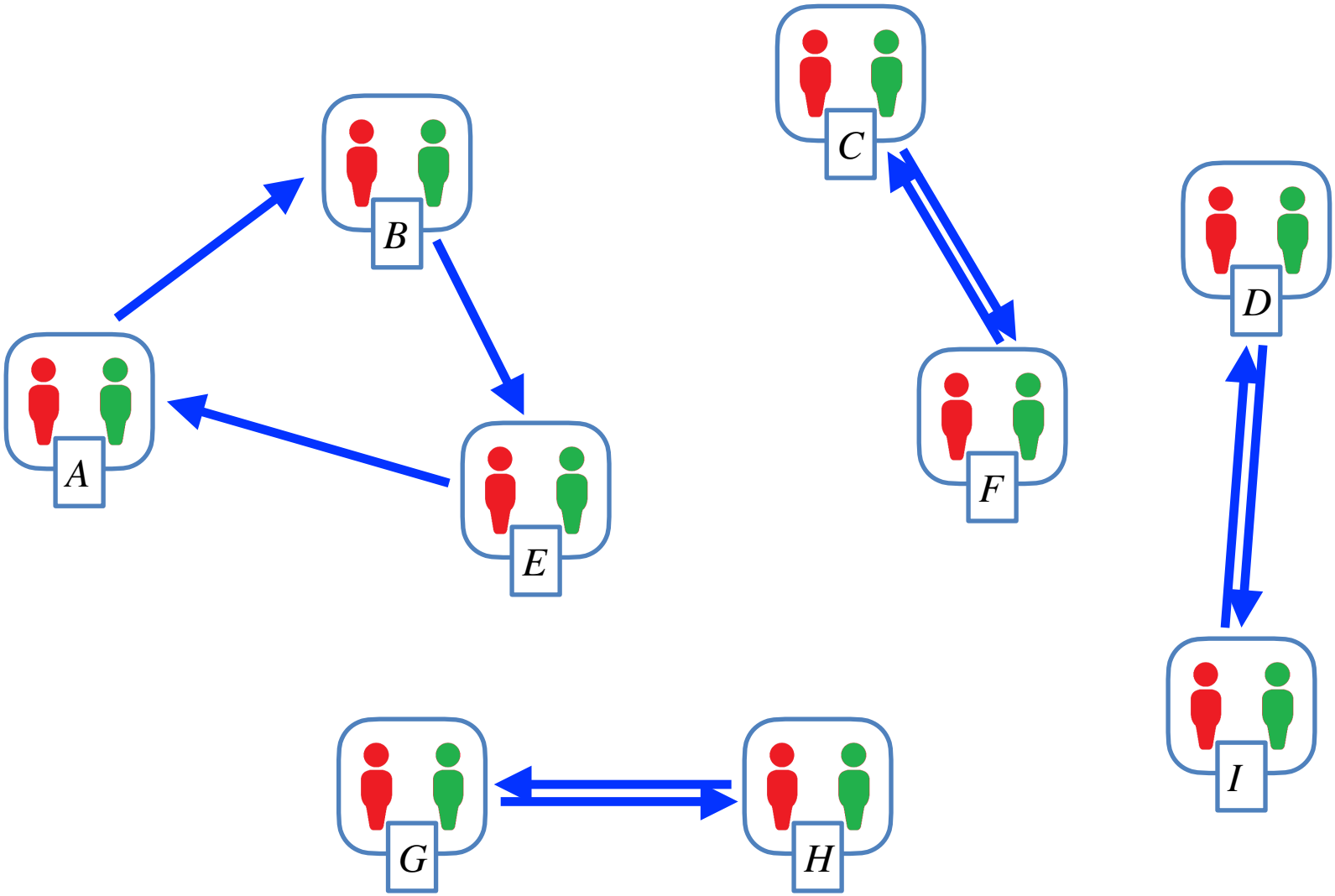
Next step: cycles of size at most 3



Next step: cycles of size at most 3



Next step: cycles of size at most 3



Cycles of size at most 3: algorithm

Cycles of size at most 3: algorithm

Theorem:
The problem of finding a cover with cycles of size at most 3 that covers the maximal number of pairs is NP-hard.

Cycles of size at most 3: algorithm

For each edge a binary variable x_e.

maximize: $\sum_{c \in \mathcal{C}} w_e \cdot x_e$

subject to:

Cycles of size at most 3: algorithm

For each edge a binary variable x_e.

maximize: $\sum_{c \in \mathscr{C}} w_e \cdot x_e$

subject to:

$$\sum_{e_{\text{out}}=(v_i,\cdot)} x_{e_{\text{out}}} = \sum_{e_{\text{in}}=(\cdot,v_i)} x_{e_{\text{in}}} \text{ for } v \in V$$

Cycles of size at most 3: algorithm

For each edge a binary variable x_e.

maximize: $\sum_{c \in \mathscr{C}} w_e \cdot x_e$

subject to:

$$\sum_{e_{\text{out}}=(v_i,\cdot)} x_{e_{\text{out}}} = \sum_{e_{\text{in}}=(\cdot,v_i)} x_{e_{\text{in}}} \quad \text{for } v \in V$$

$$\sum_{e_{\text{out}}=(v_i,\cdot)} x_{e_{\text{out}}} \leq 1 \quad \text{for } v \in V$$

Cycles of size at most 3: algorithm

For each edge a binary variable x_e.

maximize: $\sum_{c \in \mathscr{C}} w_e \cdot x_e$

subject to:

$$\sum_{e_{\text{out}}=(v_i,\cdot)} x_{e_{\text{out}}} = \sum_{e_{\text{in}}=(\cdot,v_i)} x_{e_{\text{in}}} \quad \text{for } v \in V$$

$$\sum_{e_{\text{out}}=(v_i,\cdot)} x_{e_{\text{out}}} \leq 1 \quad \text{for } v \in V$$

$x_{e_1} + x_{e_2} + x_{e_3} \leq 2$ for non-cyclic path $(x_{e_1}, x_{e_2}, x_{e_3})$

Cycles of size at most 3: algorithm

EDGE FORMULATION

For each edge a binary variable x_e.

maximize: $\sum_{c \in \mathscr{C}} w_e \cdot x_e$

subject to:

$$\sum_{e_{\text{out}}=(v_i,\cdot)} x_{e_{\text{out}}} = \sum_{e_{\text{in}}=(\cdot,v_i)} x_{e_{\text{in}}} \quad \text{for } v \in V$$

$$\sum_{e_{\text{out}}=(v_i,\cdot)} x_{e_{\text{out}}} \leq 1 \quad \text{for } v \in V$$

$x_{e_1} + x_{e_2} + x_{e_3} \leq 2$ for non-cyclic path $(x_{e_1}, x_{e_2}, x_{e_3})$

CYCLES FORMULATION

$\mathscr{C}$: set of all cycles of size ≤ 3.
for $c \in \mathscr{C}$: a binary variable x_c

maximize: $\sum_{c \in \mathscr{C}} w_c \cdot x_c$

subject to: $\sum_{c:\, v \in c} x_c \leq 1$ for $v \in V$.

Cycles of size at most 3: algorithm

EDGE FORMULATION

For each edge a binary variable x_e.

maximize: $\sum_{c \in \mathscr{C}} w_e \cdot x_e$

subject to:

$$\sum_{e_{\text{out}}=(v_i,\cdot)} x_{e_{\text{out}}} = \sum_{e_{\text{in}}=(\cdot,v_i)} x_{e_{\text{in}}} \quad \text{for } v \in V$$

$$\sum_{e_{\text{out}}=(v_i,\cdot)} x_{e_{\text{out}}} \leq 1 \quad \text{for } v \in V$$

$x_{e_1} + x_{e_2} + x_{e_3} \leq 2$ for non-cyclic path $(x_{e_1}, x_{e_2}, x_{e_3})$

CYCLES FORMULATION

$\mathscr{C}$: set of all cycles of size ≤ 3.
for $c \in \mathscr{C}$: a binary variable x_c

maximize: $\sum_{c \in \mathscr{C}} w_c \cdot x_c$

subject to: $\sum_{c:\, v \in c} x_c \leq 1$ for $v \in V$.

With m edges:

- size of cycle formulation: $O(m^2)$
- Size of edge formulation: $O(m^3)$

Cycles of size at most 3: algorithm

Theorem:
The LP relaxation of the cycle formulation weakly dominates the LP relaxation of the edge formulation.

Consider a solution to the LP relaxation of the cycle formulation.

From it we build a solution to the LP relaxation of the edge formulation:
For each edge its value is the sum of values of all the cycles of which it belongs.

How about capacity and conservation constraints?

Cycles of size at most 3: algorithm

EDGE FORMULATION

For each edge a binary variable x_e.

maximize: $\sum_{c \in \mathscr{C}} w_e \cdot x_e$

subject to:

$$\sum_{e_{\text{out}}=(v_i,\cdot)} x_{e_{\text{out}}} = \sum_{e_{\text{in}}=(\cdot,v_i)} x_{e_{\text{in}}} \quad \text{for } v \in V$$

$$\sum_{e_{\text{out}}=(v_i,\cdot)} x_{e_{\text{out}}} \leq 1 \quad \text{for } v \in V$$

$x_{e_1} + x_{e_2} + x_{e_3} \leq 2$ for non-cyclic path $(x_{e_1}, x_{e_2}, x_{e_3})$

CYCLES FORMULATION

$\mathscr{C}$: set of all cycles of size ≤ 3.
for $c \in \mathscr{C}$: a binary variable x_c

maximize: $\sum_{c \in \mathscr{C}} w_c \cdot x_c$

subject to: $\sum_{c:\, v \in c} x_c \leq 1$ for $v \in V$.

Cycles of size at most 3: algorithm

EDGE FORMULATION

For each edge a binary variable x_e.

maximize: $\sum_{c \in \mathscr{C}} w_e \cdot x_e$

subject to:

$$\sum_{e_{\text{out}}=(v_i,\cdot)} x_{e_{\text{out}}} = \sum_{e_{\text{in}}=(\cdot,v_i)} x_{e_{\text{in}}} \quad \text{for } v \in V$$

$$\sum_{e_{\text{out}}=(v_i,\cdot)} x_{e_{\text{out}}} \leq 1 \quad \text{for } v \in V$$

$x_{e_1} + x_{e_2} + x_{e_3} \leq 2$ for non-cyclic path $(x_{e_1}, x_{e_2}, x_{e_3})$

CYCLES FORMULATION

$\mathscr{C}$: set of all cycles of size ≤ 3.
for $c \in \mathscr{C}$: a binary variable x_c

maximize: $\sum_{c \in \mathscr{C}} w_c \cdot x_c$

subject to: $\sum_{c:\, v \in c} x_c \leq 1$ for $v \in V$.

Cycles of size at most 3: algorithm

Theorem:
The LP relaxation of the cycle formulation weakly dominates the LP relaxation of the edge formulation.

Consider a solution to the LP relaxation of the cycle formulation.
From it we build a solution to the LP relaxation of the edge formulation:
For each edge its value is the sum of values of all the cycles of which it belongs.

How about capacity and conservation constraints?
The path constraints:

Cycles of size at most 3: algorithm

Theorem:
The LP relaxation of the cycle formulation weakly dominates the LP relaxation of the edge formulation.

Consider a solution to the LP relaxation of the cycle formulation.
From it we build a solution to the LP relaxation of the edge formulation:
For each edge its value is the sum of values of all the cycles of which it belongs.

How about capacity and conservation constraints?
The path constraints:
$e_c(p)$: the number of edges that c and p share.

Cycles of size at most 3: algorithm

Theorem:
The LP relaxation of the cycle formulation weakly dominates the LP relaxation of the edge formulation.

Consider a solution to the LP relaxation of the cycle formulation.
From it we build a solution to the LP relaxation of the edge formulation:
For each edge its value is the sum of values of all the cycles of which it belongs.

How about capacity and conservation constraints?
The path constraints:

$e_c(p)$: the number of edges that c and p share.

$v_c(p)$: the number of interior vertices of p that are on cycle c.

Cycles of size at most 3: algorithm

Theorem:
The LP relaxation of the cycle formulation weakly dominates the LP relaxation of the edge formulation.

Consider a solution to the LP relaxation of the cycle formulation.
From it we build a solution to the LP relaxation of the edge formulation:
For each edge its value is the sum of values of all the cycles of which it belongs.

How about capacity and conservation constraints?
The path constraints:

$e_c(p)$: the number of edges that c and p share.

$v_c(p)$: the number of interior vertices of p that are on cycle c.

$$\sum_{e \in p} x_e = \sum_{c \in \mathcal{C}} x_c \cdot e_c(p)$$

Cycles of size at most 3: algorithm

Theorem:
The LP relaxation of the cycle formulation weakly dominates the LP relaxation of the edge formulation.

Consider a solution to the LP relaxation of the cycle formulation.
From it we build a solution to the LP relaxation of the edge formulation:
For each edge its value is the sum of values of all the cycles of which it belongs.

How about capacity and conservation constraints?
The path constraints:

$e_c(p)$: the number of edges that c and p share.

$v_c(p)$: the number of interior vertices of p that are on cycle c.

$$\sum_{e \in p} x_e = \sum_{c \in \mathcal{C}} x_c \cdot e_c(p) \leq \sum_{c \in \mathcal{C}} x_c \cdot v_c(p)$$

Cycles of size at most 3: algorithm

Theorem:
The LP relaxation of the cycle formulation weakly dominates the LP relaxation of the edge formulation.

Consider a solution to the LP relaxation of the cycle formulation.
From it we build a solution to the LP relaxation of the edge formulation:
For each edge its value is the sum of values of all the cycles of which it belongs.

How about capacity and conservation constraints?
The path constraints:

$e_c(p)$: the number of edges that c and p share.

$v_c(p)$: the number of interior vertices of p that are on cycle c.

$$\sum_{e \in p} x_e = \sum_{c \in \mathcal{C}} x_c \cdot e_c(p) \leq \sum_{c \in \mathcal{C}} x_c \cdot v_c(p) \leq \sum_{v \in \text{interior}(p)} \sum_{v \in c} x_c$$

Cycles of size at most 3: algorithm

Theorem:
The LP relaxation of the cycle formulation weakly dominates the LP relaxation of the edge formulation.

Consider a solution to the LP relaxation of the cycle formulation.
From it we build a solution to the LP relaxation of the edge formulation:
For each edge its value is the sum of values of all the cycles of which it belongs.

How about capacity and conservation constraints?
The path constraints:

$e_c(p)$: the number of edges that c and p share.

$v_c(p)$: the number of interior vertices of p that are on cycle c.

$$\sum_{e \in p} x_e = \sum_{c \in \mathcal{C}} x_c \cdot e_c(p) \leq \sum_{c \in \mathcal{C}} x_c \cdot v_c(p) \leq \sum_{v \in \text{interior}(p)} \sum_{v \in c} x_c \leq |\text{interior}(p)|$$

Cycles of size at most 3: algorithm

Theorem:
The LP relaxation of the cycle formulation weakly dominates the LP relaxation of the edge formulation.

The reverse implication is not true. Consider a cycle of size $n > 3$.

Cycles of size at most 3: algorithm

EDGE FORMULATION

For each edge a binary variable x_e.

maximize: $\sum_{c \in \mathscr{C}} w_e \cdot x_e$

subject to:

$$\sum_{e_{\text{out}}=(v_i,\cdot)} x_{e_{\text{out}}} = \sum_{e_{\text{in}}=(\cdot,v_i)} x_{e_{\text{in}}} \quad \text{for } v \in V$$

$$\sum_{e_{\text{out}}=(v_i,\cdot)} x_{e_{\text{out}}} \leq 1 \quad \text{for } v \in V$$

$x_{e_1} + x_{e_2} + x_{e_3} \leq 2$ for non-cyclic path $(x_{e_1}, x_{e_2}, x_{e_3})$

CYCLES FORMULATION

$\mathscr{C}$: set of all cycles of size ≤ 3.
for $c \in \mathscr{C}$: a binary variable x_c

maximize: $\sum_{c \in \mathscr{C}} w_c \cdot x_c$

subject to: $\sum_{c:\, v \in c} x_c \leq 1$ for $v \in V$.

Also in practice this formulation is more efficient!

Cycles of size at most 3: algorithm

CYCLES FORMULATION

$\mathscr{C}$: set of all cycles of size ≤ 3.
for $c \in \mathscr{C}$: a binary variable x_c

maximize: $\sum_{c \in \mathscr{C}} w_c \cdot x_c$

subject to: $\sum_{c:\, v \in c} x_c \leq 1$ for $v \in V$.

Also in practice this formulation is more efficient!

Cycles of size at most 3: algorithm

This is not sufficiently efficient.

CYCLES FORMULATION

$\mathscr{C}$: set of all cycles of size ≤ 3.
for $c \in \mathscr{C}$: a binary variable x_c

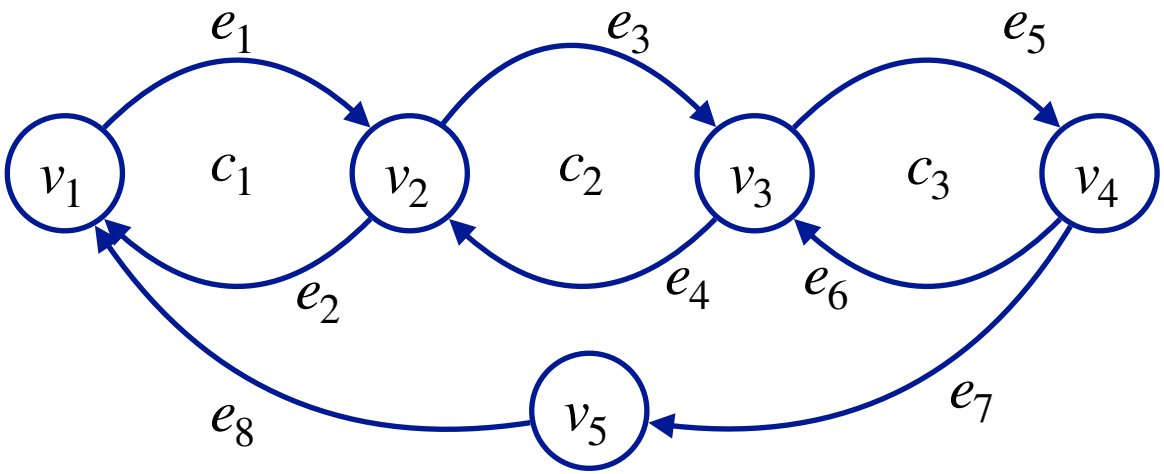
maximize: $\sum_{c \in \mathscr{C}} w_c \cdot x_c$

subject to: $\sum_{c:\, v \in c} x_c \leq 1$ for $v \in V$.

Heuristic algorithms using LP:
David J. Abraham, Avrim Blum, Tuomas Sandholm: Clearing algorithms for barter exchange markets: enabling nationwide kidney exchanges. EC 2007: 295-304

Also in practice this formulation is more efficient!

Cycles of size at most 3: LP relaxation



CYCLES FORMULATION

max: $2c_1 + 2c_2 + 2c_3$

$$
\begin{array}{lrcll}
\text{s.t:} & c_1 & & \leq 1 & (v_1) \\
 & c_1 + c_2 & & \leq 1 & (v_2) \\
 & c_2 + c_3 & & \leq 1 & (v_3) \\
 & c_3 & & \leq 1 & (v_4) \\
\text{with} & c_1, c_2, c_3 & & \geq 0 &
\end{array}
$$

DUAL FORMULATION

min: $v_1 + v_2 + v_3 + v_4$

$$
\begin{array}{lrcll}
\text{s.t:} & v_1 + v_2 & & \geq 2 & (c_1) \\
 & v_2 + v_3 & & \geq 2 & (c_2) \\
 & v_3 + v_4 & & \geq 2 & (c_3) \\
\text{with} & v_1, v_2, v_3, v_4 & & \geq 0 &
\end{array}
$$

Cycles of size at most 3: LP relaxation

We can first use fewer cycles, getting a smaller linear program P.

CYCLES FORMULATION

max: $2c_1 + 2c_2 + 2c_3$

$$
\begin{array}{lrcl|l}
\text{s.t:} & c_1 & & \leq 1 & (v_1) \\
& c_1 + c_2 & & \leq 1 & (v_2) \\
& & c_2 + c_3 & \leq 1 & (v_3) \\
& & c_3 & \leq 1 & (v_4) \\
\text{with} & c_1, c_2, c_3 & & \geq 0 &
\end{array}
$$

DUAL FORMULATION

min: $v_1 + v_2 + v_3 + v_4$

$$
\begin{array}{lrcl|l}
\text{s.t:} & v_1 + v_2 & & \geq 2 & (c_1) \\
& & v_2 + v_3 & \geq 2 & (c_2) \\
& & & v_3 + v_4 \geq 2 & (c_3) \\
\text{with} & v_1, v_2, v_3, v_4 & & \geq 0 &
\end{array}
$$

Cycles of size at most 3: LP relaxation

We can first use fewer cycles, getting a smaller linear program P'.

CYCLES FORMULATION

$$
\begin{array}{lrrrrl}
\text{max:} & 2c_1 + & & 2c_3 & & \\
\text{s.t:} & c_1 & & & \leq 1 & (v_1) \\
 & c_1 & & & \leq 1 & (v_2) \\
 & & & c_3 & \leq 1 & (v_3) \\
 & & & c_3 & \leq 1 & (v_4) \\
\text{with} & c_1, & & c_3 & \geq 0 &
\end{array}
$$

DUAL FORMULATION

$$
\begin{array}{lrrrrl}
\text{min:} & v_1 + v_2 + v_3 + v_4 & & & & \\
\text{s.t:} & v_1 + v_2 & & & \geq 2 & (c_1) \\
 & & v_2 + v_3 & & \geq 2 & (c_2) \\
 & & & v_3 + v_4 & \geq 2 & (c_3) \\
\text{with} & v_1, v_2, v_3, v_4 & & & \geq 0 &
\end{array}
$$

Cycles of size at most 3: LP relaxation

We can first use fewer cycles, getting a smaller linear program P'. A dual program D' will simply have less constraints.

CYCLES FORMULATION

$$
\begin{array}{lrrrl}
\text{max: } & 2c_1 + & & 2c_3 & \\
\text{s.t:} & c_1 & & \leq 1 & (v_1) \\
 & c_1 & & \leq 1 & (v_2) \\
 & & c_3 & \leq 1 & (v_3) \\
 & & c_3 & \leq 1 & (v_4) \\
\text{with} & c_1, & c_3 & \geq 0 &
\end{array}
$$

DUAL FORMULATION

$$
\begin{array}{lrrrl}
\text{min: } & v_1 + v_2 + v_3 + v_4 & & & \\
\text{s.t:} & v_1 + v_2 & & \geq 2 & (c_1) \\
 & & v_3 + v_4 & \geq 2 & (c_3) \\
\text{with} & v_1, v_2, v_3, v_4 & & \geq 0 &
\end{array}
$$

Cycles of size at most 3: LP relaxation

We can first use fewer cycles, getting a smaller linear program P'. A dual program D' will simply have less constraints. However, we can then check the other constraints of the original D.

CYCLES FORMULATION

$$
\begin{array}{lrrrl|l}
\text{max: } & 2c_1 + & & 2c_3 & & \\
\text{s.t:} & c_1 & & & \leq 1 & (v_1) \\
 & c_1 & & & \leq 1 & (v_2) \\
 & & & c_3 & \leq 1 & (v_3) \\
 & & & c_3 & \leq 1 & (v_4) \\
\text{with} & c_1, & & c_3 & \geq 0 &
\end{array}
$$

DUAL FORMULATION

$$
\begin{array}{lrrrl|l}
\text{min: } & v_1 + v_2 + v_3 + v_4 & & & & \\
\text{s.t:} & v_1 + v_2 & & & \geq 2 & (c_1) \\
 & & v_2 + v_3 & & \geq 2 & (c_2) \\
 & & & v_3 + v_4 & \geq 2 & (c_3) \\
\text{with} & v_1, v_2, v_3, v_4 & & & \geq 0 &
\end{array}
$$

Cycles of size at most 3: LP relaxation

We can first use fewer cycles, getting a smaller linear program P'. A dual program D' will simply have less constraints. However, we can then check the other constraints of the original D. If all constraints are satisfied our solution is optimal with respect to D, and so with respect to P.

CYCLES FORMULATION

$$
\begin{array}{lllll|l}
\text{max: } 2c_1 + & & 2c_3 & & & \\
\text{s.t:} & c_1 & & \leq 1 & & (v_1) \\
& c_1 & & \leq 1 & & (v_2) \\
& & c_3 & \leq 1 & & (v_3) \\
& & c_3 & \leq 1 & & (v_4) \\
\text{with} & c_1, & c_3 & \geq 0 & &
\end{array}
$$

DUAL FORMULATION

$$
\begin{array}{lllll|l}
\text{min: } v_1 + v_2 + v_3 + v_4 & & & & & \\
\text{s.t:} & v_1 + v_2 & & \geq 2 & & (c_1) \\
& & v_2 + v_3 & \geq 2 & & (c_2) \\
& & & v_3 + v_4 & \geq 2 & (c_3) \\
\text{with} & v_1, v_2, v_3, v_4 & & \geq 0 & &
\end{array}
$$

Cycles of size at most 3: LP relaxation

We can first use fewer cycles, getting a smaller linear program P'. A dual program D' will simply have less constraints. However, we can then check the other constraints of the original D. If all constraints are satisfied our solution is optimal with respect to D, and so with respect to P. If not, then we add the cycle corresponding to violated constraint to the primal.

CYCLES FORMULATION

max: $2c_1 + 2c_2 + 2c_3$

$$
\begin{array}{lrcl|l}
\text{s.t:} & c_1 & & \leq 1 & (v_1) \\
& c_1 + c_2 & & \leq 1 & (v_2) \\
& & c_2 + c_3 & \leq 1 & (v_3) \\
& & c_3 & \leq 1 & (v_4) \\
\text{with} & c_1, c_2, c_3 & & \geq 0 &
\end{array}
$$

DUAL FORMULATION

min: $v_1 + v_2 + v_3 + v_4$

$$
\begin{array}{lrcl|l}
\text{s.t:} & v_1 + v_2 & & \geq 2 & (c_1) \\
& & v_2 + v_3 & \geq 2 & (c_2) \\
& & & v_3 + v_4 \geq 2 & (c_3) \\
\text{with} & v_1, v_2, v_3, v_4 & & \geq 0 &
\end{array}
$$

Cycles of size at most 3: LP relaxation

We can first use fewer cycles, getting a smaller linear program P'. A dual program D' will simply have less constraints. However, we can then check the other constraints of the original D. If all constraints are satisfied our solution is optimal with respect to D, and so with respect to P. If not, then we add the cycle corresponding to violated constraint to the primal.
And so we can build the LP iteratively.
(otherwise it would not fit in memory.)

CYCLES FORMULATION

max: $2c_1 + 2c_2 + 2c_3$

$$
\begin{array}{lrcl|l}
\text{s.t:} & c_1 & & \leq 1 & (v_1) \\
& c_1 + c_2 & & \leq 1 & (v_2) \\
& & c_2 + c_3 & \leq 1 & (v_3) \\
& & c_3 & \leq 1 & (v_4) \\
\text{with} & c_1, c_2, c_3 & & \geq 0 &
\end{array}
$$

DUAL FORMULATION

min: $v_1 + v_2 + v_3 + v_4$

$$
\begin{array}{lrcl|l}
\text{s.t:} & v_1 + v_2 & & \geq 2 & (c_1) \\
& & v_2 + v_3 & \geq 2 & (c_2) \\
& & & v_3 + v_4 \geq 2 & (c_3) \\
\text{with} & v_1, v_2, v_3, v_4 & & \geq 0 &
\end{array}
$$

Cycles of size at most 3: LP relaxation

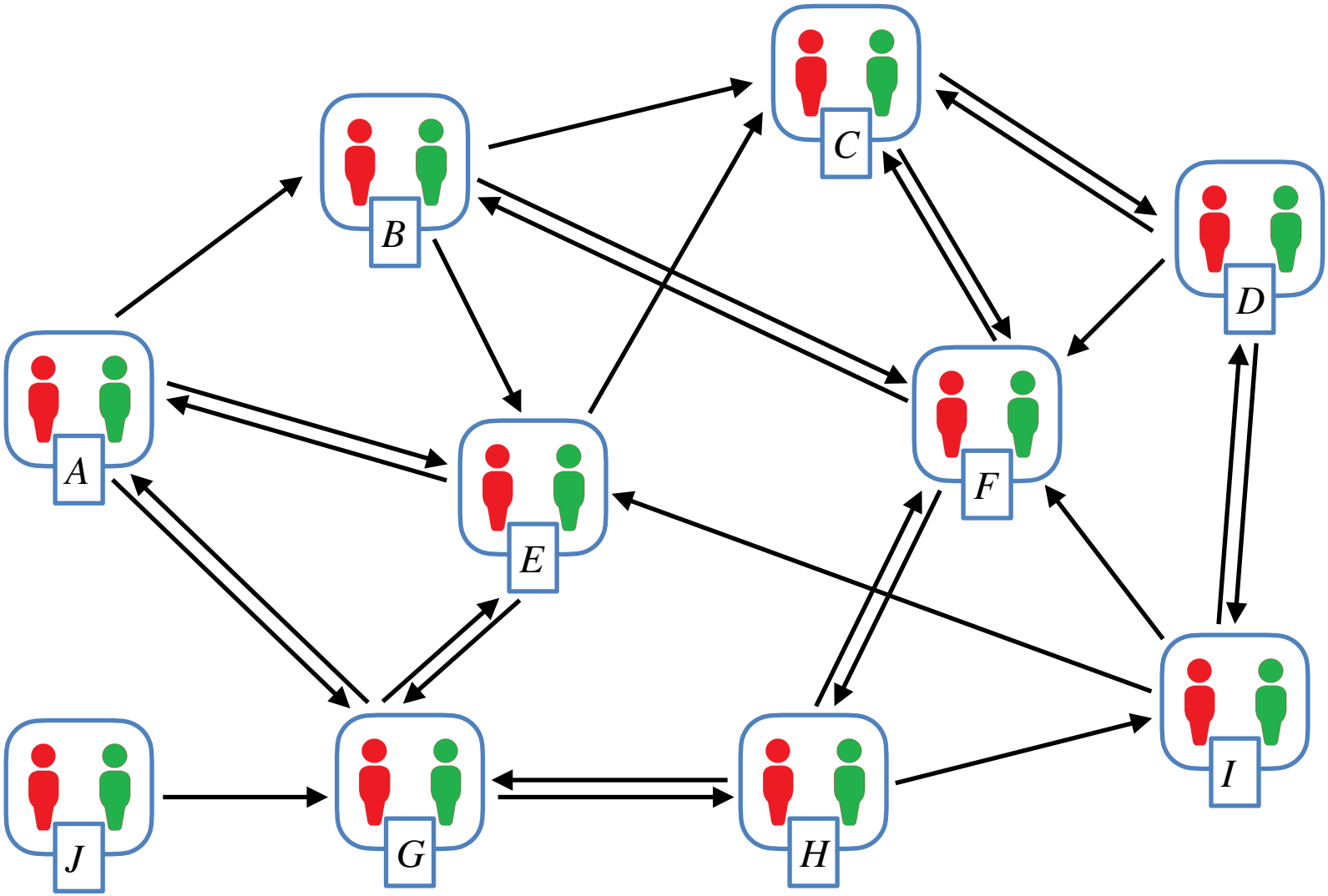
We can first use fewer cycles, getting a smaller linear program P'. A dual program D' will simply have less constraints. However, we can then check the other constraints of the original D. If all constraints are satisfied our solution is optimal with respect to D, and so with respect to P. If not, then we add the cycle corresponding to violated constraint to the primal.
And so we can build the LP iteratively.

Branch and bound based on LP with several optimisations.

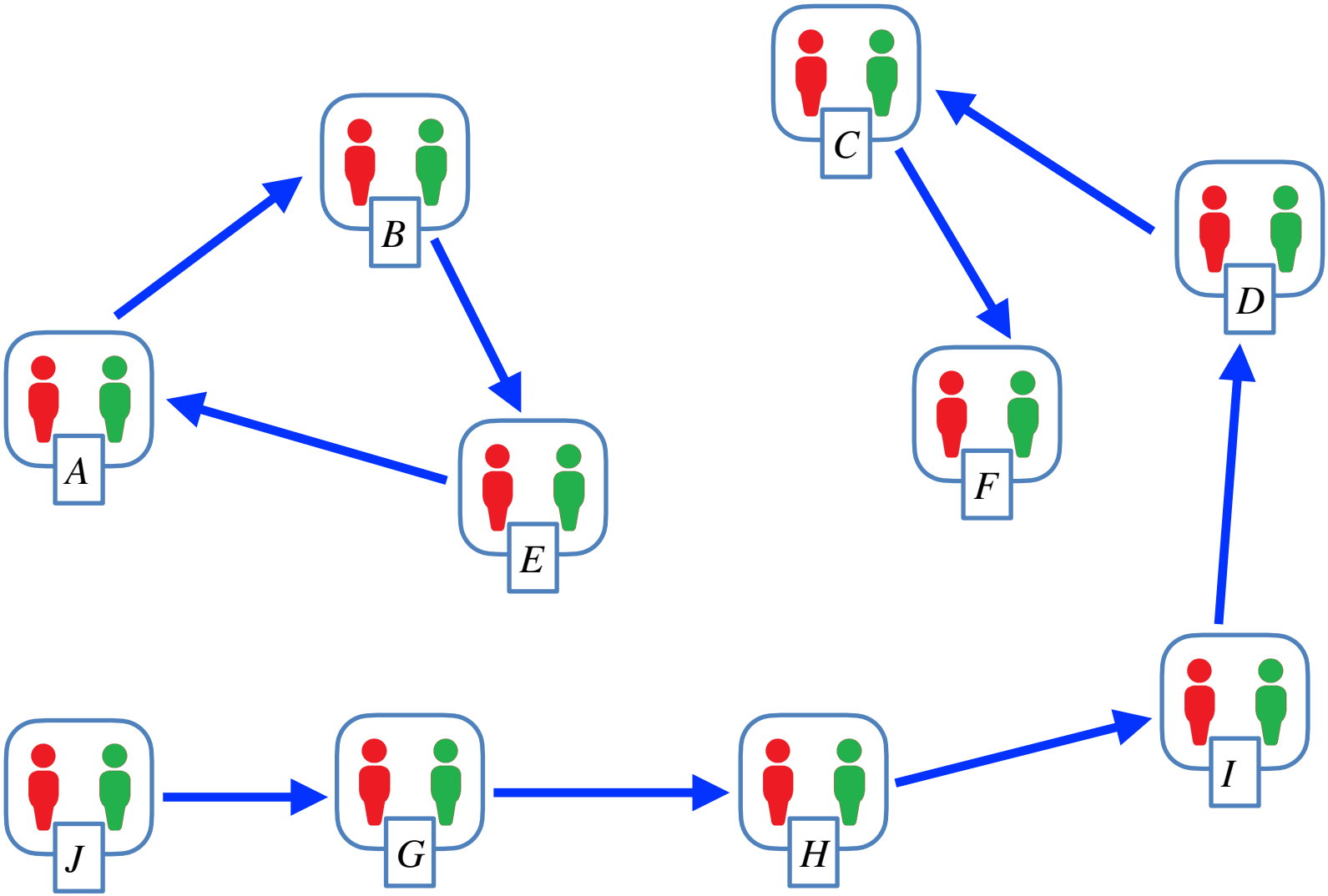
More details:
David J. Abraham, Avrim Blum, Tuomas Sandholm: Clearing algorithms for barter exchange markets: enabling nationwide kidney exchanges. EC 2007: 295-304

Further complication: altruistic donors

Further complication: altruistic donors



Further complication: altruistic donors



Further complication: altruistic donors

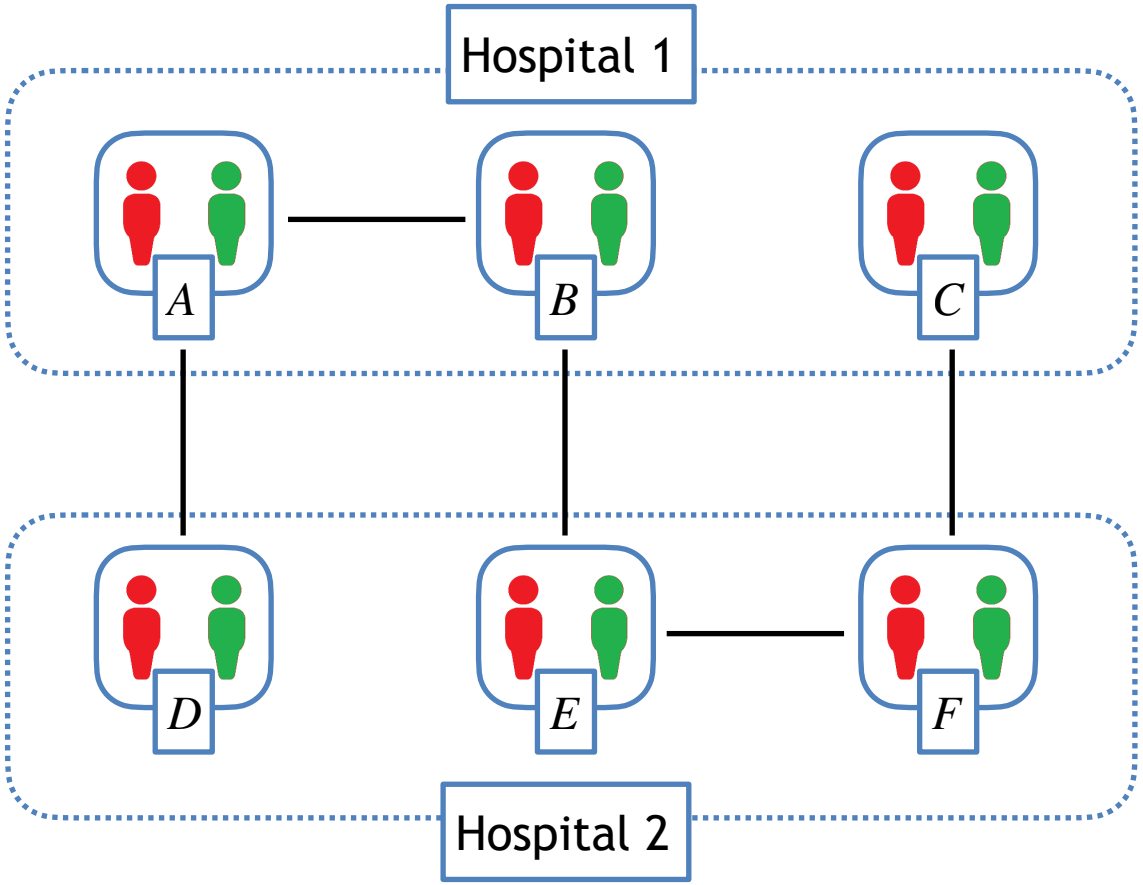
Source: New York Times
(https://www.nytimes.com/2012/02/19/health/lives-forever-linked-through-kidney-transplant-chain-124.html)

Strategic hospitals

Problem:
Hospitals typically report only the pairs that are hard to match.

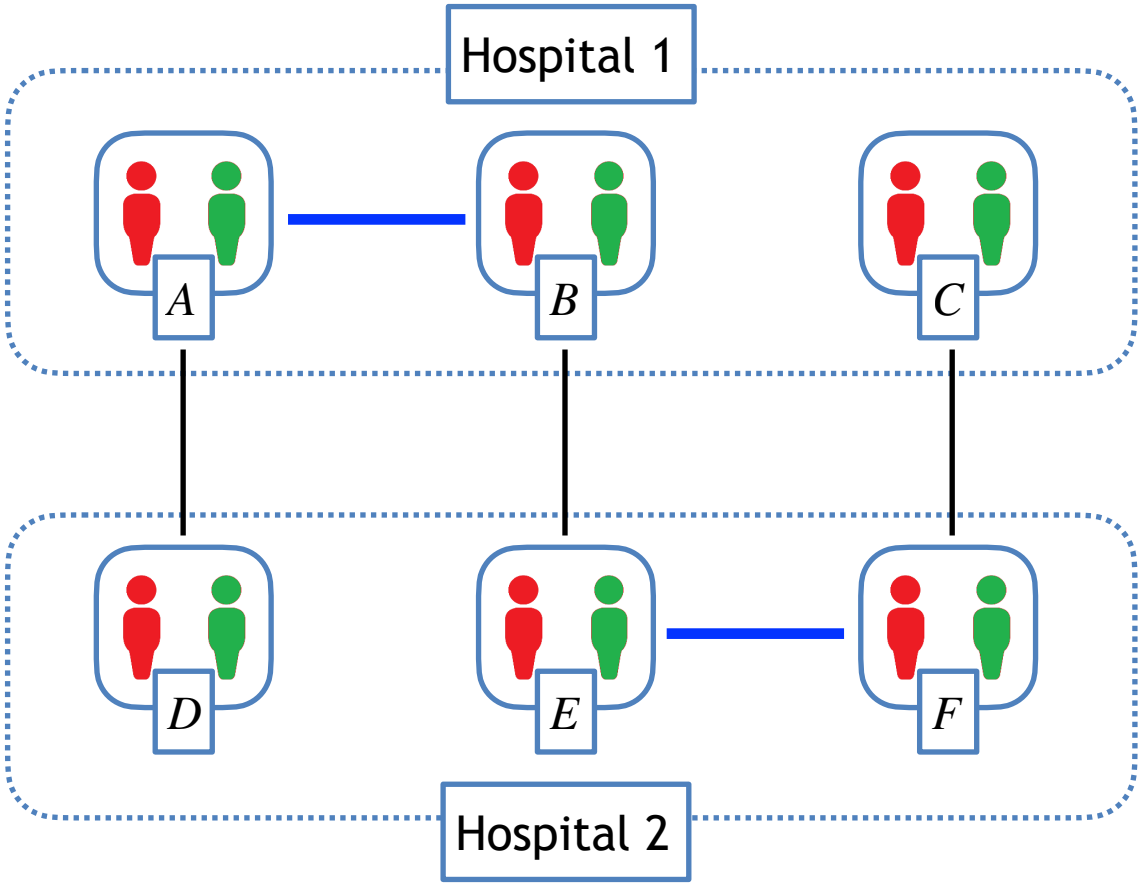
Strategic hospitals: non-optimality

Problem:
Hospitals typically report only the pairs that are hard to match.



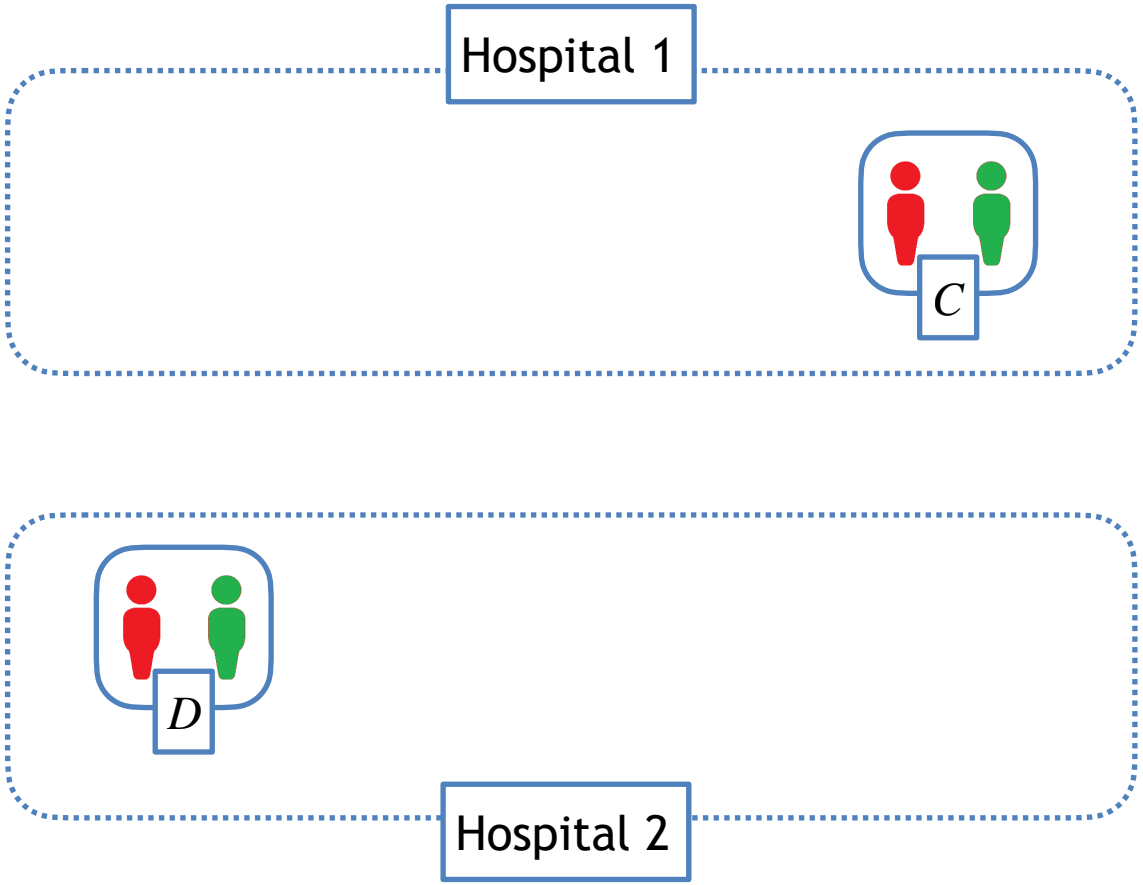
Strategic hospitals: non-optimality

Problem:
Hospitals typically report only the pairs that are hard to match.



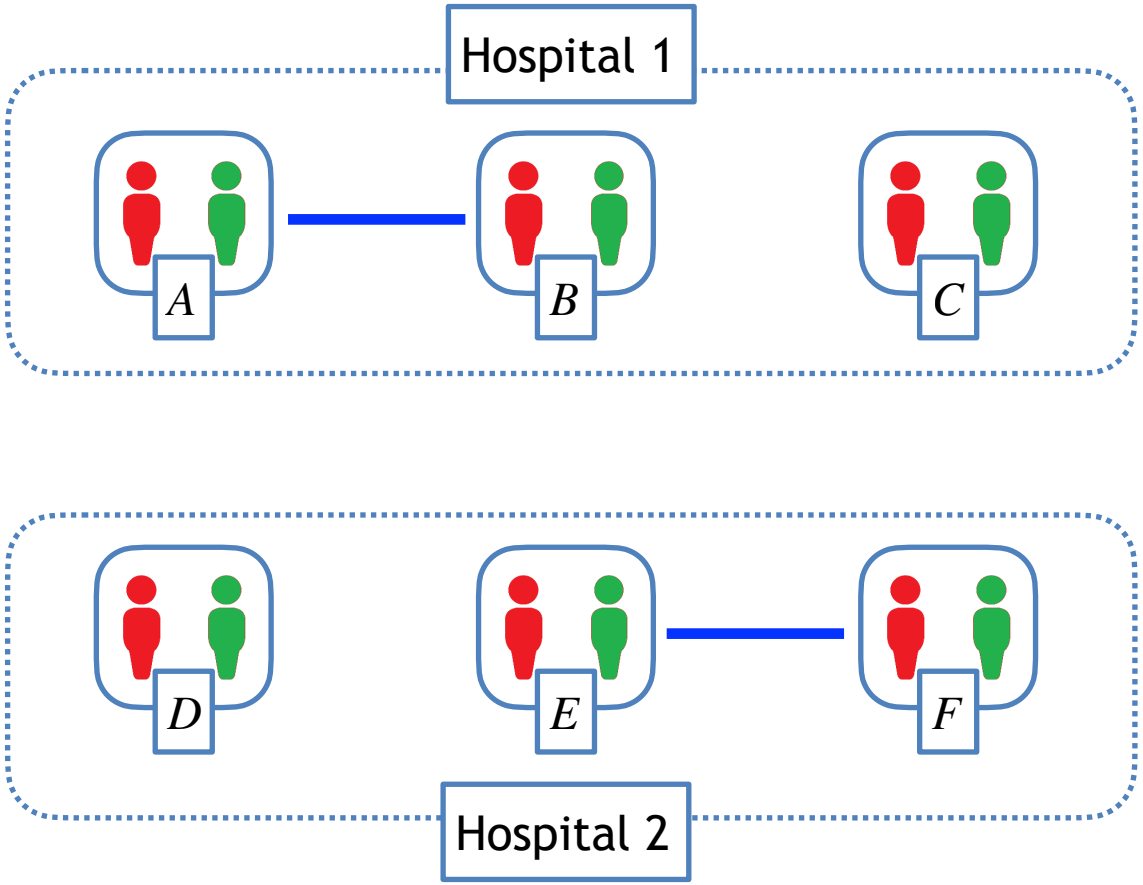
Strategic hospitals: non-optimality

Problem:
Hospitals typically report only the pairs that are hard to match.



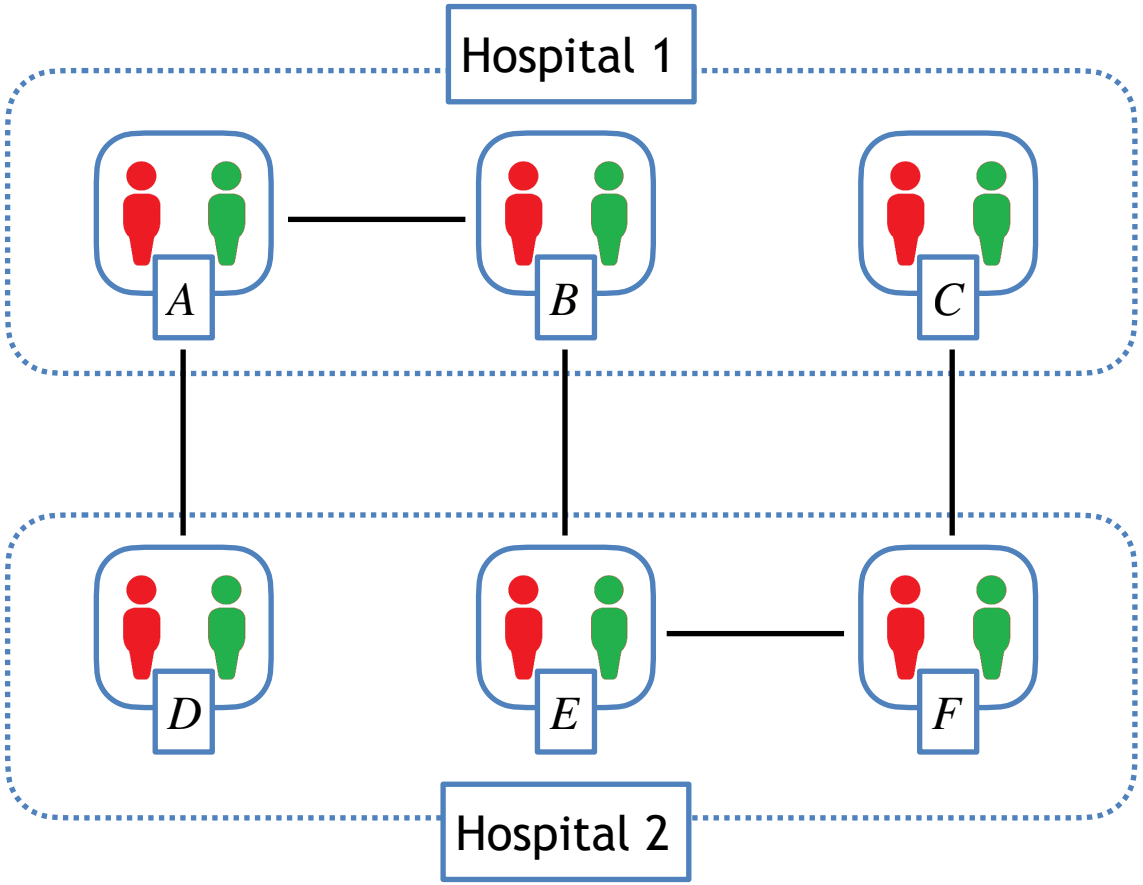
Strategic hospitals: non-optimality

Problem:
Hospitals typically report only the pairs that are hard to match.



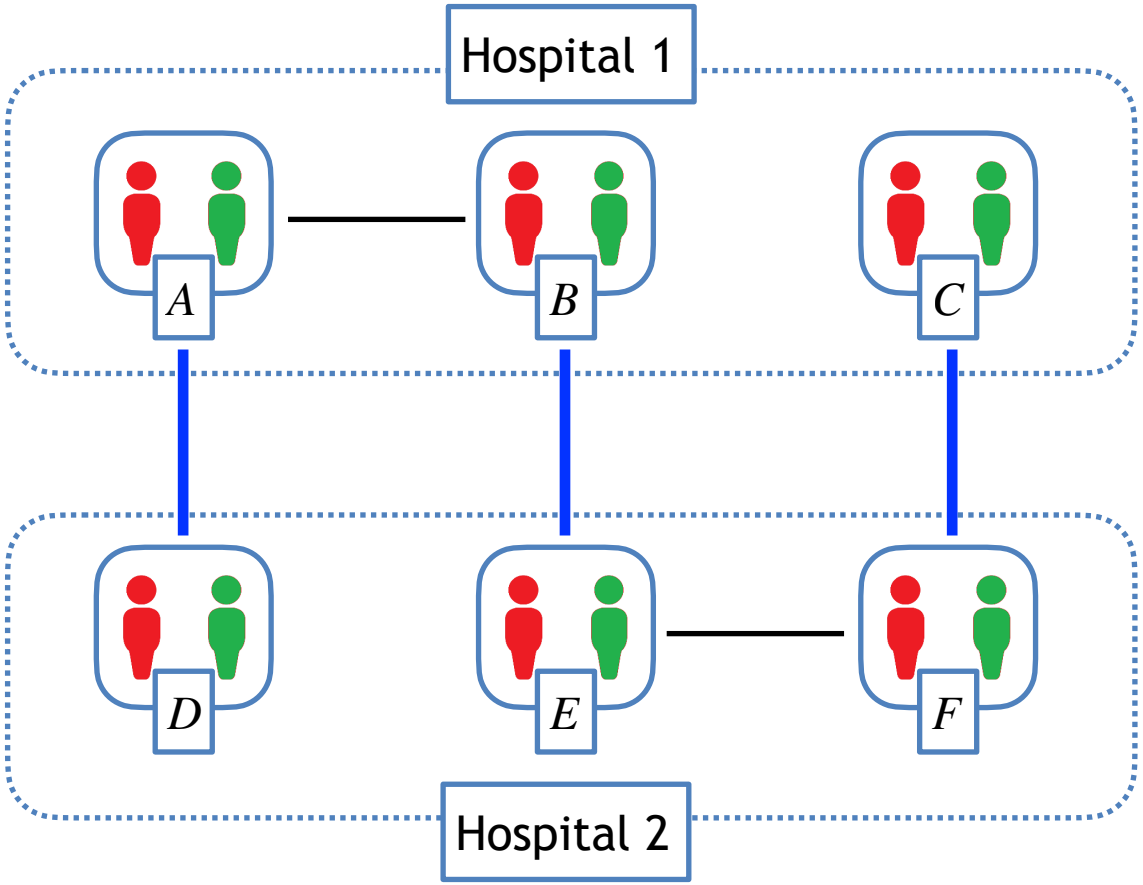
Strategic hospitals: non-optimality

Problem:
Hospitals typically report only the pairs that are hard to match.



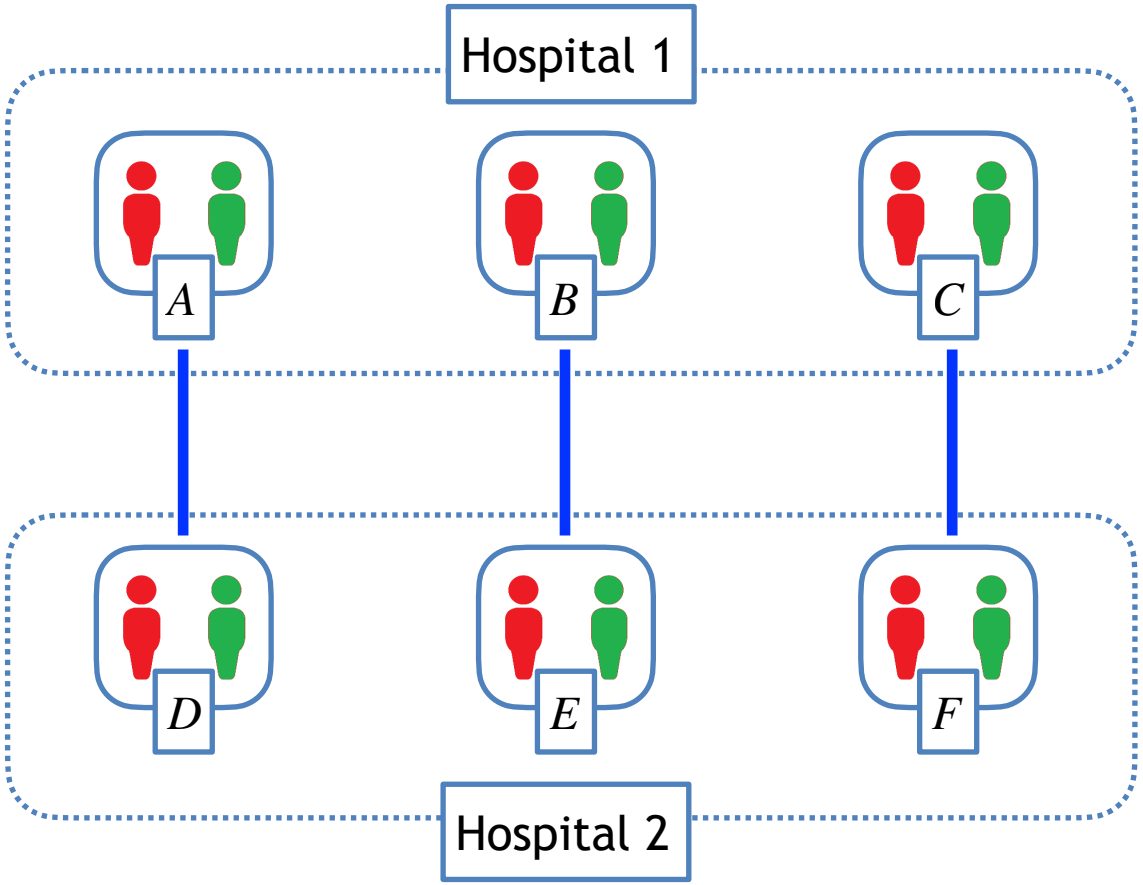
Strategic hospitals: non-optimality

Problem:
Hospitals typically report only the pairs that are hard to match.



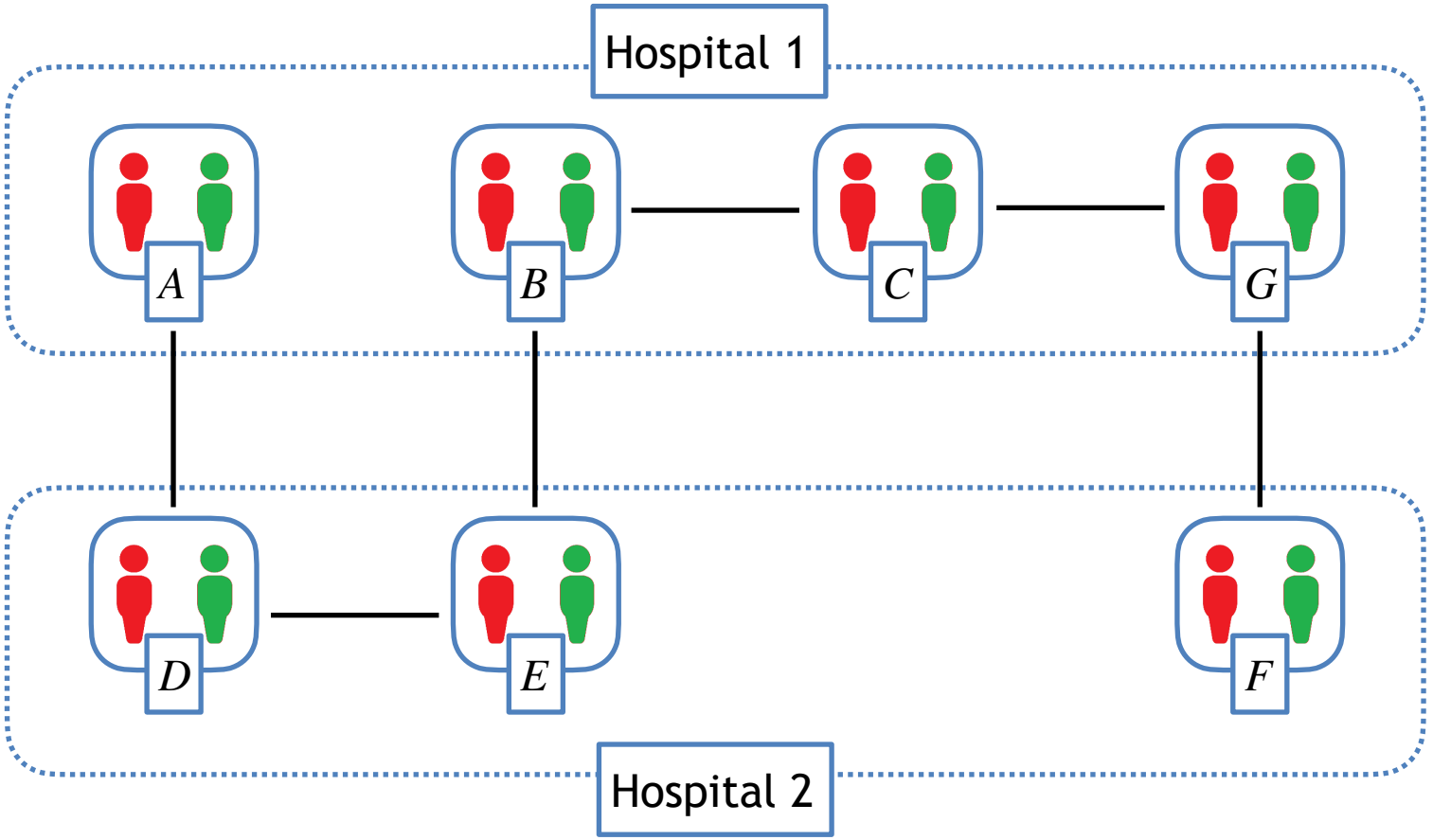
Strategic hospitals: non-optimality

Problem:
Hospitals typically report only the pairs that are hard to match.



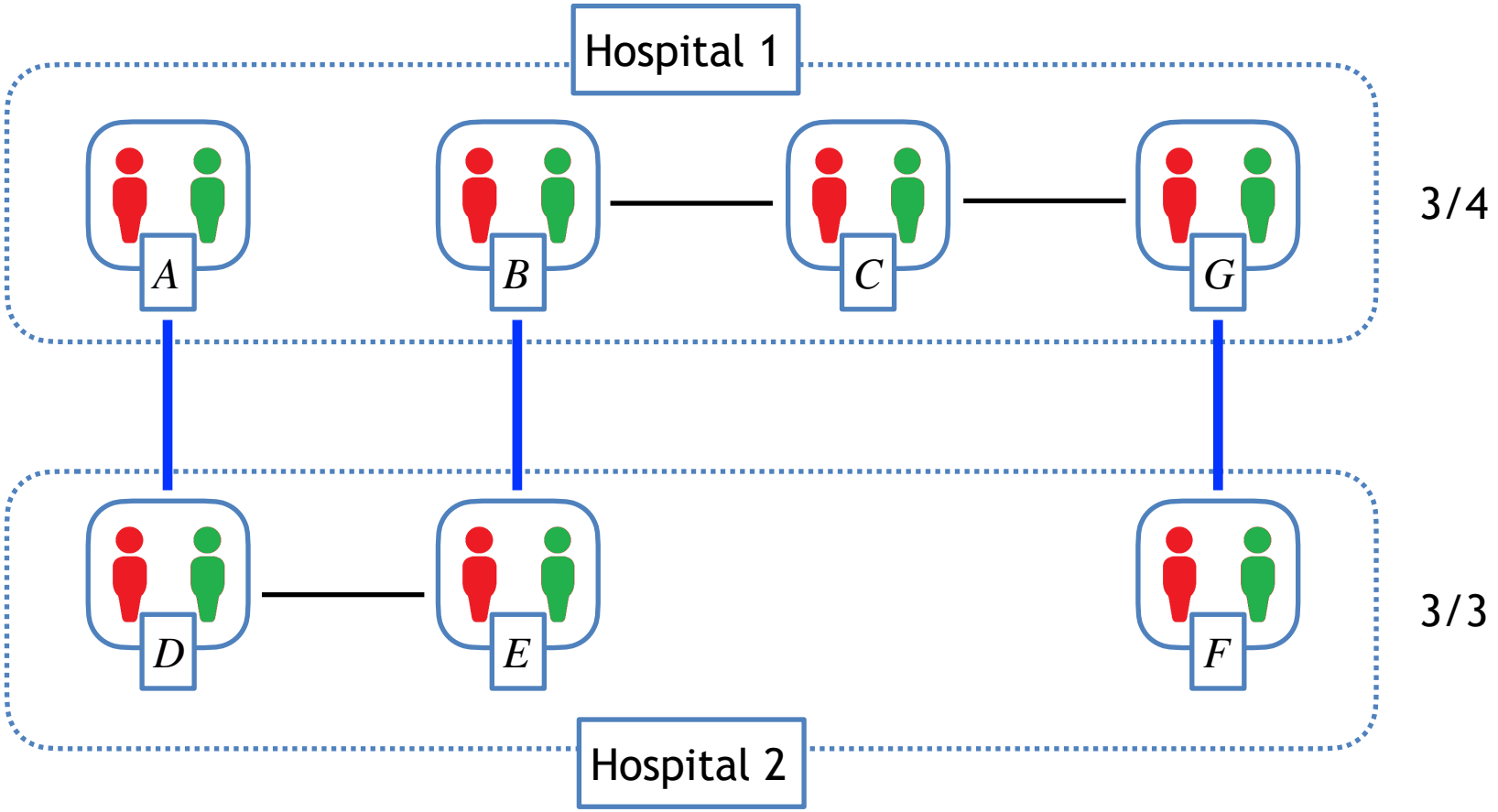
Strategic hospitals: incentives

Problem:
Hospitals typically report only the pairs that are hard to match.



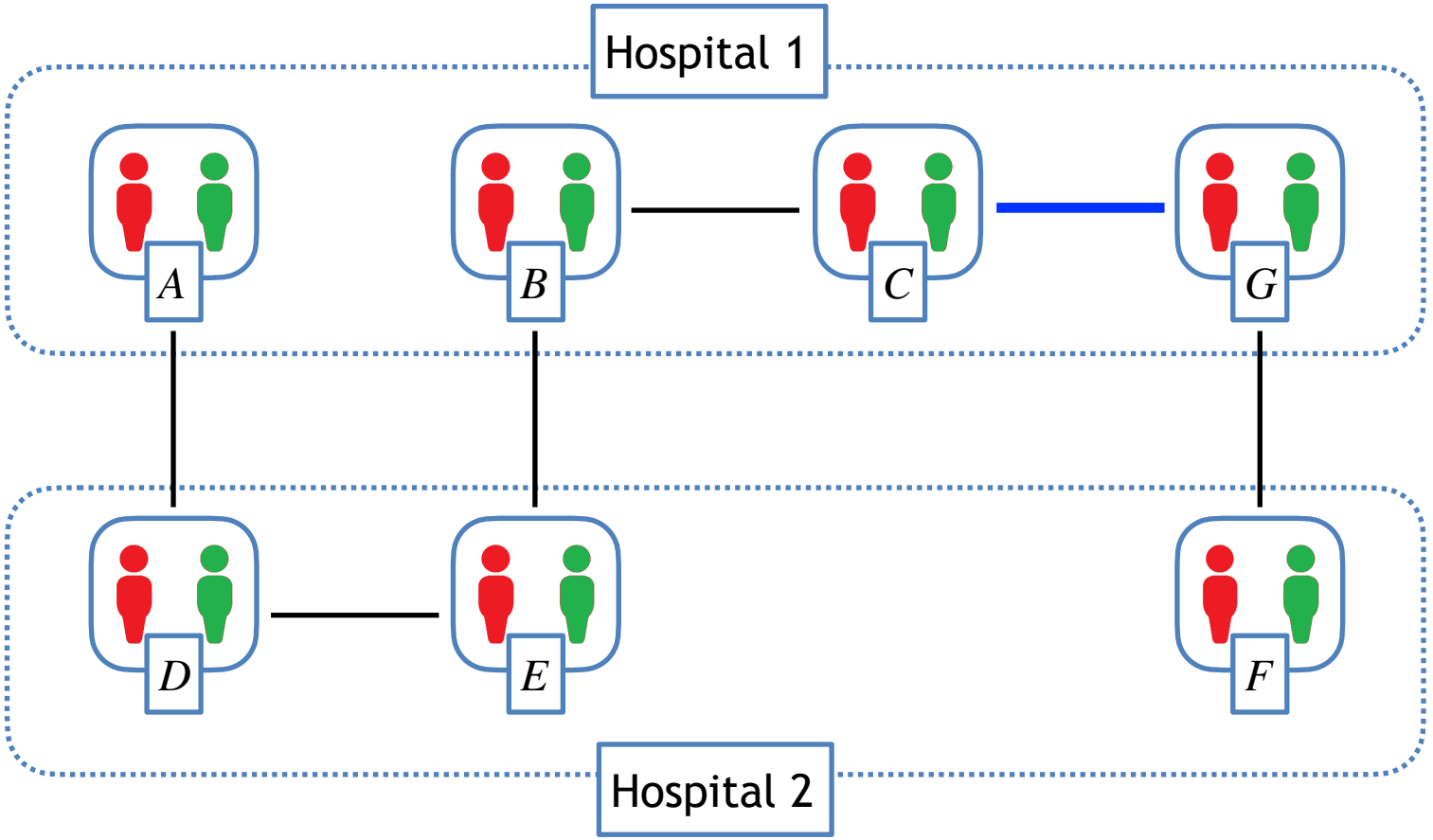
Strategic hospitals: incentives

Problem:
Hospitals typically report only the pairs that are hard to match.



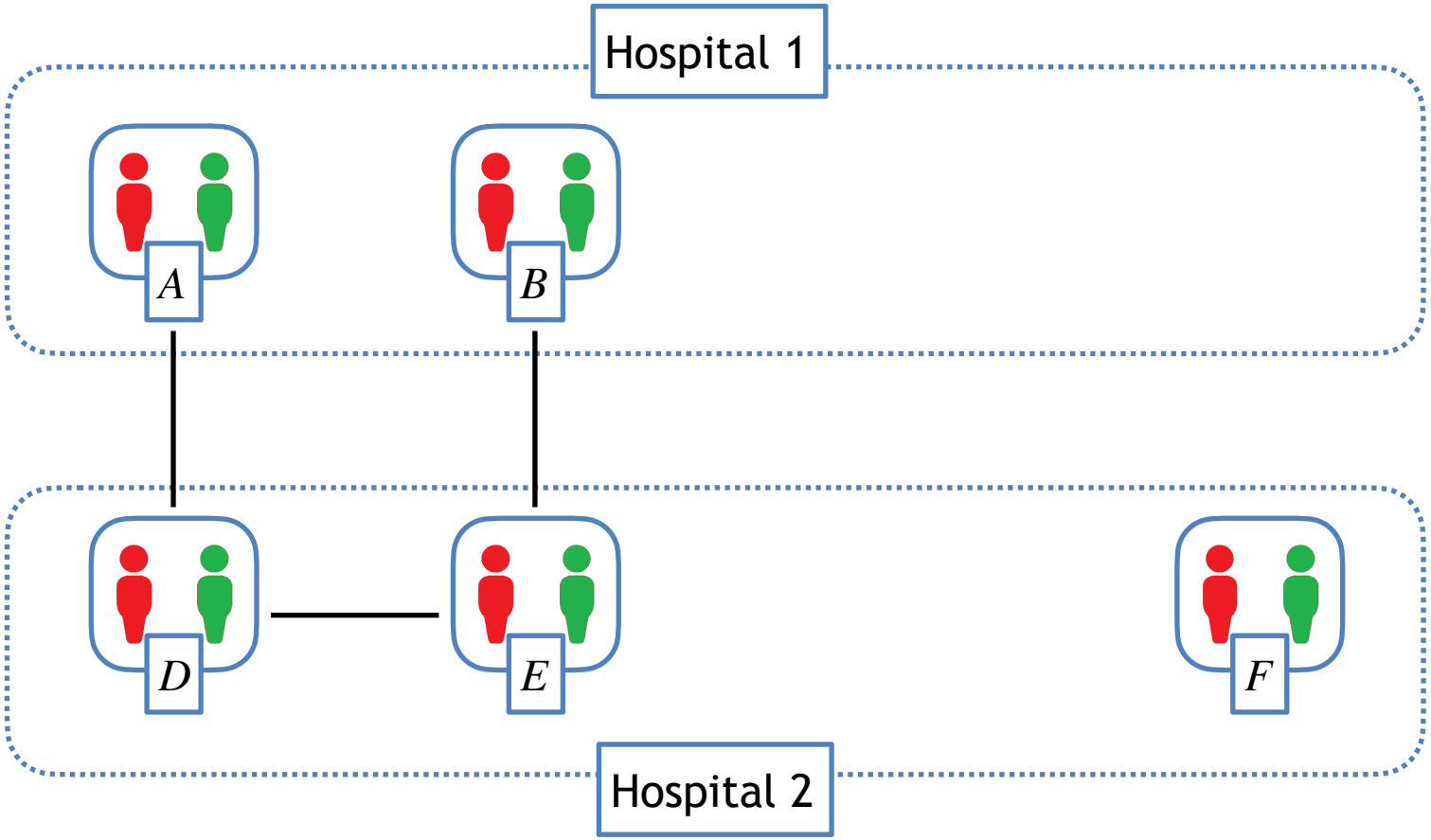
Strategic hospitals: incentives

Problem:
Hospitals typically report only the pairs that are hard to match.



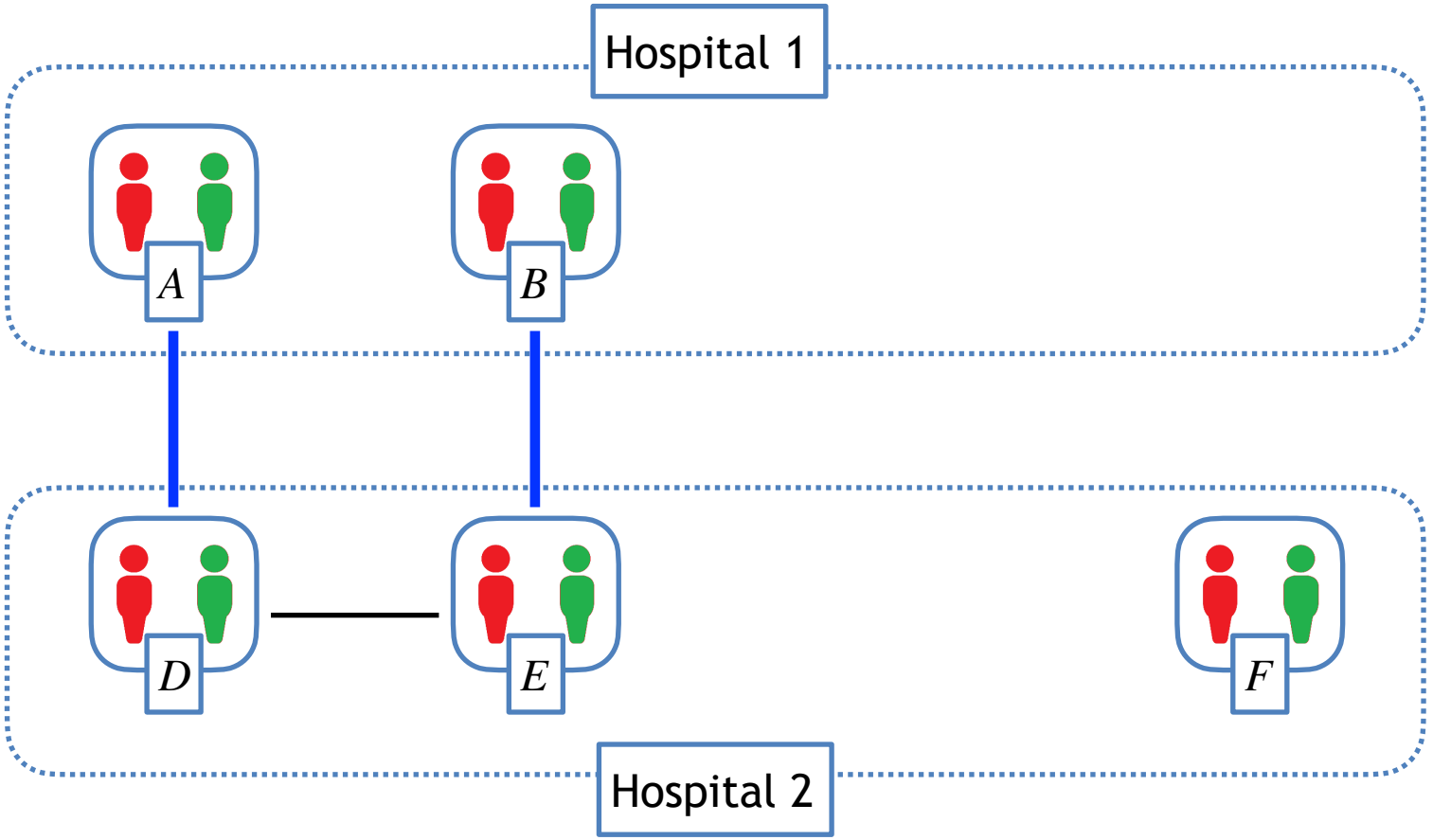
Strategic hospitals: incentives

Problem:
Hospitals typically report only the pairs that are hard to match.



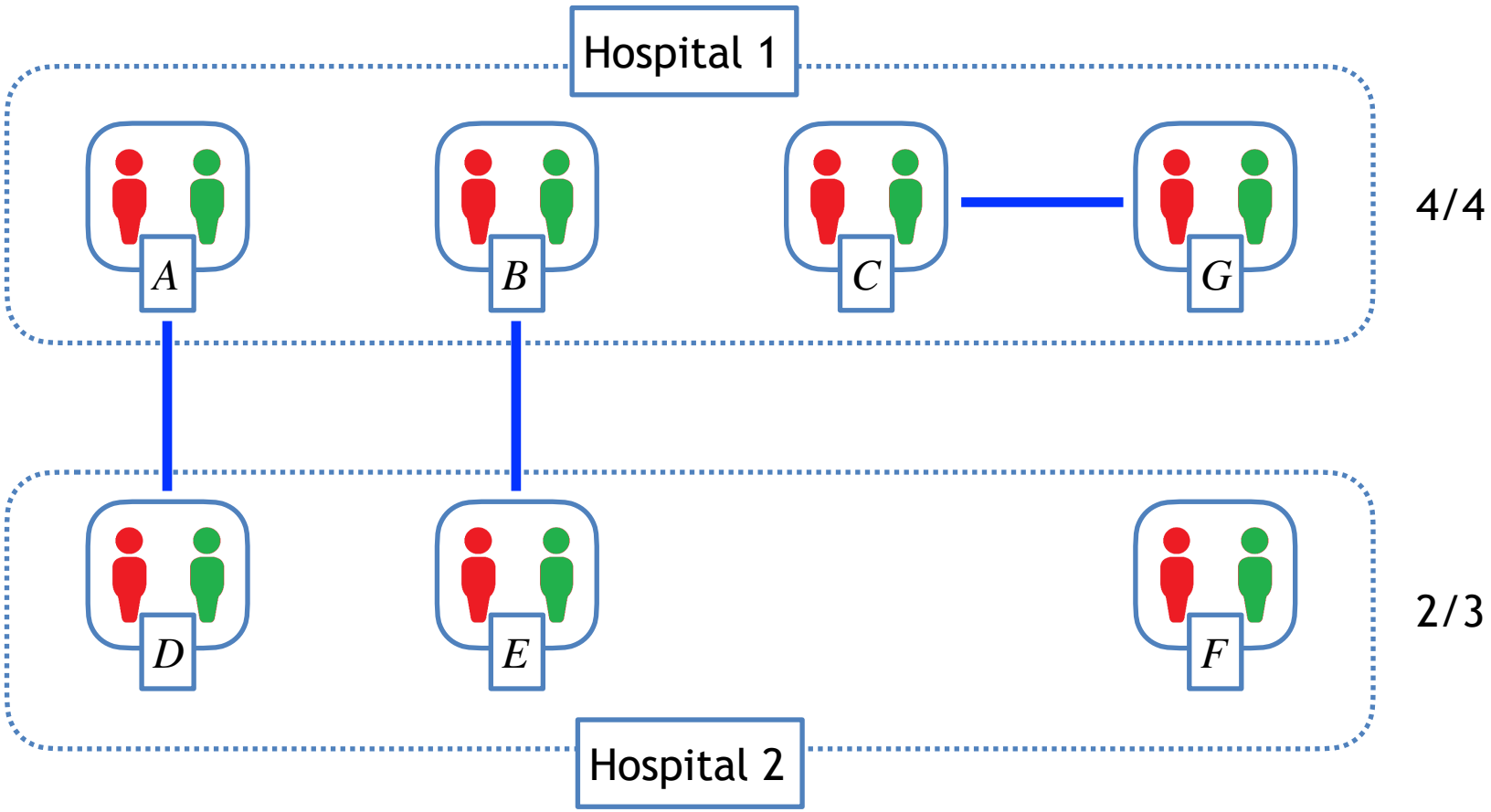
Strategic hospitals: incentives

Problem:
Hospitals typically report only the pairs that are hard to match.



Strategic hospitals: incentives

Problem:
Hospitals typically report only the pairs that are hard to match.



Strategic hospitals: incentives

Problem:
Hospitals typically report only the pairs that are hard to match.

This is a big challenge to design mechanisms, where the hospitals will have incentive to report all pairs.

Strategic hospitals: incentives

Problem:
Hospitals typically report only the pairs that are hard to match.

This is a big challenge to design mechanisms, where the hospitals will have incentive to report all pairs.

Mechanisms based on credits:
John P. Dickerson, Avinatan Hassidim, Tuomas Sandholm, David Sarne: Strategy-Proof and Efficient Kidney Exchange Using a Credit Mechanism. AAAI 2015: 921-928

Summary

1. Top-trading cycle. (**Econ Theory!**)
2. Covering with matchings. (Graph Theory and CS!)
3. Covering with cycles of size at most 3. (**AI!**)
4. Strategic hospitals. (Mechanism Design!)

Literature

Top Trading Cycle on Wikipedia.

Kolmogorov, Vladimir (2009), Blossom V: A new implementation of a minimum cost perfect matching algorithm, Mathematical Programming Computation, 1 (1): 43–67

David J. Abraham, Avrim Blum, Tuomas Sandholm: Clearing algorithms for barter exchange markets: enabling nationwide kidney exchanges. EC 2007: 295-304

Alvin E. Roth, Tayfun Sönmez, M. Utku Ünver. Kidney Exchange. The Quarterly Journal of Economics, Volume 119, Issue 2, May 2004, Pages 457–488

(A very good paper describing the general problem and issues in kidney exchange.)