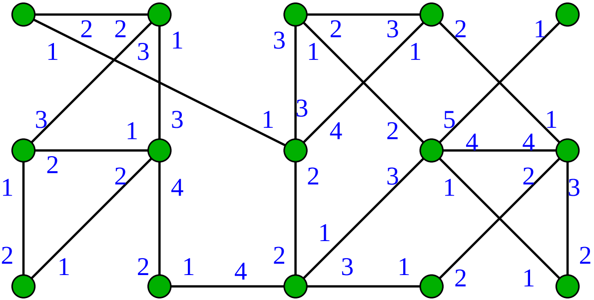
Generalized stable roommates problems

Tamás Fleiner[1]

Summer School on
Matching Problems, Markets, and Mechanisms
25 June 2013, Budapest

[1]Budapest University of Technology and Economics

The stable roommates problem

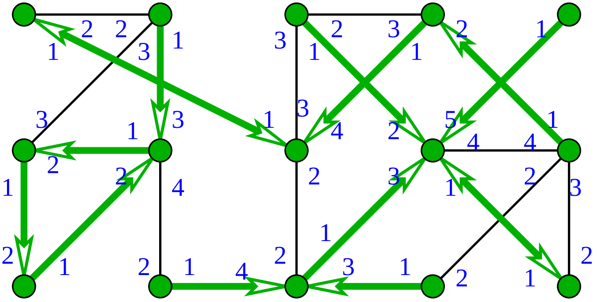


One sided market situation: any two agents can form a partnership.

It might happen that no stable matching exists.

Irving gave an algoritm that finds a stable matching if exists.

The stable roommates problem



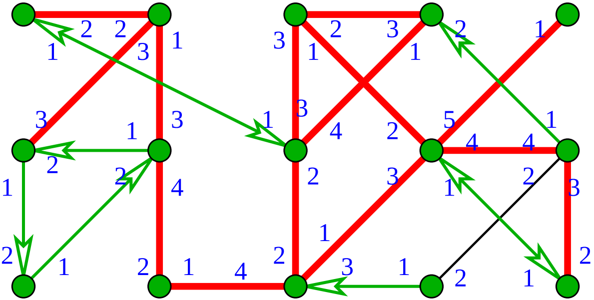
One sided market situation: any two agents can form a partnership.

It might happen that no stable matching exists.

Irving gave an algoritm that finds a stable matching if exists.

This algorithm uses similar proposal

The stable roommates problem



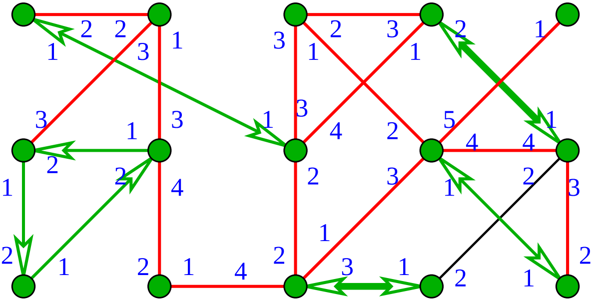
One sided market situation: any two agents can form a partnership.

It might happen that no stable matching exists.

Irving gave an algoritm that finds a stable matching if exists.

This algorithm uses similar proposal and rejection steps,

The stable roommates problem



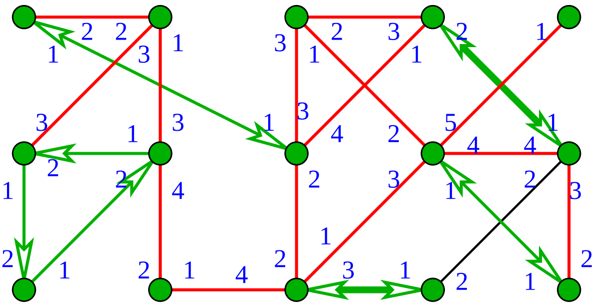
One sided market situation: any two agents can form a partnership.

It might happen that no stable matching exists.

Irving gave an algoritm that finds a stable matching if exists.

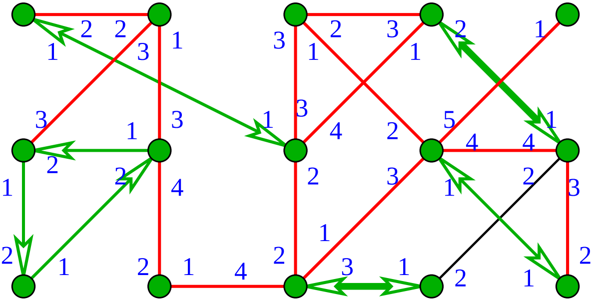
This algorithm uses similar proposal and rejection steps,

The stable roommates problem



One sided market situation: any two agents can form a partnership.
It might happen that no stable matching exists.
Irving gave an algoritm that finds a stable matching if exists.
This algorithm uses similar proposal and rejection steps,
and certain new kinds of steps.

The stable roommates problem



One sided market situation: any two agents can form a partnership.
It might happen that no stable matching exists.
Irving gave an algoritm that finds a stable matching if exists.
This algorithm uses similar proposal and rejection steps,
and certain new kinds of steps.
We shall extend Irving's algorithm to more general situations.

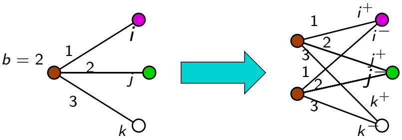
Finding a stable b-matching

Def: Graph $G = (V, E)$ and quota function $b : V \to \mathbb{N}$ is given. A ***b*-matching** is a subset M of e st each vertex v is incident to at most $b(v)$ edges of M. If we also have linear preferences for the vertices then b-matching S is **stable** if it dominates all other edges: if $e = uv \in E \setminus S$ then either u is incident to $b(u)$ edges of S that are all preferred to e or similar holds for v.

Finding a stable b-matching

Def: Graph $G = (V, E)$ and quota function $b : V \to \mathbb{N}$ is given. A ***b*-matching** is a subset M of e st each vertex v is incident to at most $b(v)$ edges of M. If we also have linear preferences for the vertices then b-matching S is **stable** if it dominates all other edges: if $e = uv \in E \setminus S$ then either u is incident to $b(u)$ edges of S that are all preferred to e or similar holds for v.
Natural question: Is it possible to reduce the problem of finding a stable b-matching to the problem of finding a stable matching?

Finding a stable b-matching

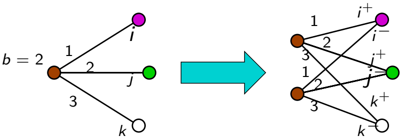


Def: Graph $G = (V, E)$ and quota function $b : V \to \mathbb{N}$ is given. A ***b*-matching** is a subset M of e st each vertex v is incident to at most $b(v)$ edges of M. If we also have linear preferences for the vertices then b-matching S is **stable** if it dominates all other edges: if $e = uv \in E \setminus S$ then either u is incident to $b(u)$ edges of S that are all preferred to e or similar holds for v.

Natural question: Is it possible to reduce the problem of finding a stable b-matching to the problem of finding a stable matching?

Idea: Node spitting.

Finding a stable b-matching



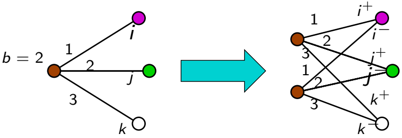
Def: Graph $G = (V, E)$ and quota function $b : V \to \mathbb{N}$ is given. A ***b*-matching** is a subset M of e st each vertex v is incident to at most $b(v)$ edges of M. If we also have linear preferences for the vertices then b-matching S is **stable** if it dominates all other edges: if $e = uv \in E \setminus S$ then either u is incident to $b(u)$ edges of S that are all preferred to e or similar holds for v.

Natural question: Is it possible to reduce the problem of finding a stable b-matching to the problem of finding a stable matching?

Idea: Node spitting.

Problem:

Finding a stable b-matching



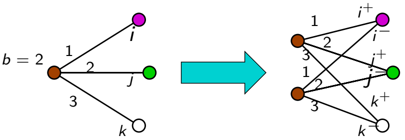
Def: Graph $G = (V, E)$ and quota function $b : V \to \mathbb{N}$ is given. A ***b*-matching** is a subset M of e st each vertex v is incident to at most $b(v)$ edges of M. If we also have linear preferences for the vertices then b-matching S is **stable** if it dominates all other edges: if $e = uv \in E \setminus S$ then either u is incident to $b(u)$ edges of S that are all preferred to e or similar holds for v.

Natural question: Is it possible to reduce the problem of finding a stable b-matching to the problem of finding a stable matching?

Idea: Node spitting.

Problem: The same edge may appear more times in a matching.

Finding a stable b-matching



Def: Graph $G = (V, E)$ and quota function $b : V \to \mathbb{N}$ is given. A ***b*-matching** is a subset M of e st each vertex v is incident to at most $b(v)$ edges of M. If we also have linear preferences for the vertices then b-matching S is **stable** if it dominates all other edges: if $e = uv \in E \setminus S$ then either u is incident to $b(u)$ edges of S that are all preferred to e or similar holds for v.

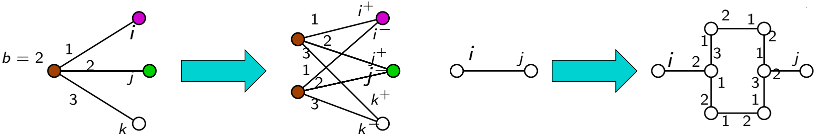
Natural question: Is it possible to reduce the problem of finding a stable b-matching to the problem of finding a stable matching?

Idea: Node spitting.

Problem: The same edge may appear more times in a matching.

New idea: This cannot happen if b has the **mto** (many-to-one) property: for any edge $e = uv$ of G we have $b(u) = 1$ or $b(v) = 1$.

Finding a stable b-matching



Def: Graph $G = (V, E)$ and quota function $b : V \to \mathbb{N}$ is given. A ***b*-matching** is a subset M of e st each vertex v is incident to at most $b(v)$ edges of M. If we also have linear preferences for the vertices then b-matching S is **stable** if it dominates all other edges: if $e = uv \in E \setminus S$ then either u is incident to $b(u)$ edges of S that are all preferred to e or similar holds for v.

Natural question: Is it possible to reduce the problem of finding a stable b-matching to the problem of finding a stable matching?

Idea: Node spitting.

Problem: The same edge may appear more times in a matching.

New idea: This cannot happen if b has the **mto** (many-to-one) property: for any edge $e = uv$ of G we have $b(u) = 1$ or $b(v) = 1$.

Solution: A simple construction achieves the mto property.

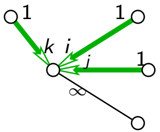
Finding a stable matching

In Irving's algorithm we keep on deleting edges such that

- no new stable matching is created
- not all stable matchings are killed

until a single stable matching remains or we conclude that no stable matching exists.

Finding a stable matching



In Irving's algorithm we keep on deleting edges such that

- no new stable matching is created
- not all stable matchings are killed

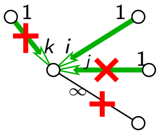
until a single stable matching remains or we conclude that no stable matching exists.

Def: Edge $e = uv$ is a **1-arc** if e is the first choice of u.

Observation: If v prefers 1-arc $e = uv$ to f then the deletion of f does not change the set of stable matchings.

Basis of Phase 1 of Irving's algorithm.

Finding a stable matching



In Irving's algorithm we keep on deleting edges such that

- no new stable matching is created
- not all stable matchings are killed

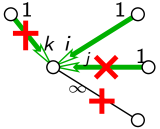
until a single stable matching remains or we conclude that no stable matching exists.

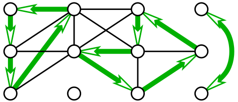
Def: Edge $e = uv$ is a **1-arc** if e is the first choice of u.

Observation: If v prefers 1-arc $e = uv$ to f then the deletion of f does not change the set of stable matchings.

Basis of Phase 1 of Irving's algorithm.

Finding a stable matching





In Irving's algorithm we keep on deleting edges such that

- no new stable matching is created
- not all stable matchings are killed

until a single stable matching remains or we conclude that no stable matching exists.

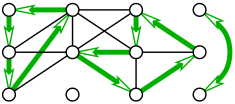
Def: Edge $e = uv$ is a **1-arc** if e is the first choice of u.

Observation: If v prefers 1-arc $e = uv$ to f then the deletion of f does not change the set of stable matchings.

Basis of Phase 1 of Irving's algorithm.

End of Phase 1: If no more GS-type deletion is possible then for each vertex v, the last choice of v is a 1-arc pointing to v, i.e. 1-arcs form vertex-disjoint oriented cycles.

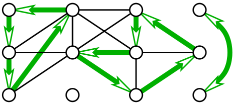
Phase 2



If no more GS-deletion is possible and all 1-arcs are bidirected then we are left with a stable b-matching. Otherwise there is a vertex u incident to at least two edges.

Def: If $e = uv$ is the 2nd choice of v then vu is a **2-arc**.

Phase 2



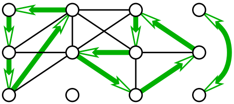
If no more GS-deletion is possible and all 1-arcs are bidirected then we are left with a stable b-matching. Otherwise there is a vertex u incident to at least two edges.

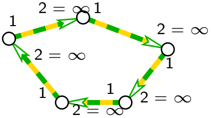
Def: If $e = uv$ is the 2nd choice of v then vu is a **2-arc**.

Observation: After Phase 1, each vertex u receives at most one 2-arc. Moreover, if u recives a 2-arc then u sends a unique 1-arc that is not bidirected.

Corollary: There is a cycle formed alternatingly by 1-arcs and 2-arcs. This is called a rotation.

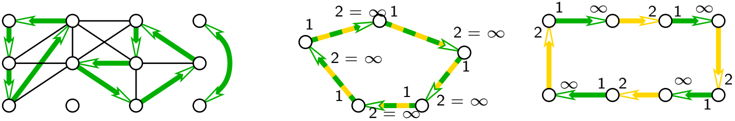
Phase 2





Def: If $e = uv$ is the 2nd choice of v then vu is a 2-**arc**.
Observation: After Phase 1, each vertex u receives at most one 2-arc. Moreover, if u recives a 2-arc then u sends a unique 1-arc that is not bidirected.
Corollary: There is a cycle formed alternatingly by 1-arcs and 2-arcs. This is called a rotation.
Two cases are possible: Either a rotation is an odd cycle and each arc in it is both a 1-arc and a 2 arc

Phase 2



Def: If $e = uv$ is the 2nd choice of v then vu is a **2-arc**.

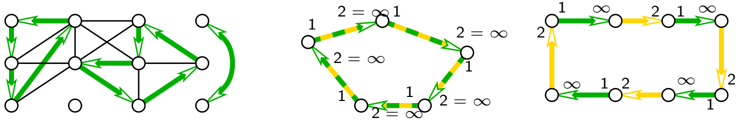
Observation: After Phase 1, each vertex u receives at most one 2-arc. Moreover, if u recives a 2-arc then u sends a unique 1-arc that is not bidirected.

Corollary: There is a cycle formed alternatingly by 1-arcs and 2-arcs. This is called a rotation.

Two cases are possible: Either a rotation is an odd cycle and each arc in it is both a 1-arc and a 2 arc

or all the set of 1-arcs and the set of 2 arcs are disjoint.

Phase 2



Def: If $e = uv$ is the 2nd choice of v then vu is a **2-arc**.

Observation: After Phase 1, each vertex u receives at most one 2-arc. Moreover, if u recives a 2-arc then u sends a unique 1-arc that is not bidirected.

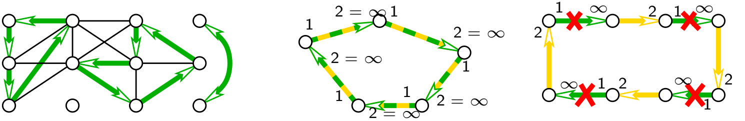
Corollary: There is a cycle formed alternatingly by 1-arcs and 2-arcs. This is called a rotation.

Two cases are possible: Either a rotation is an odd cycle and each arc in it is both a 1-arc and a 2 arc

or all the set of 1-arcs and the set of 2 arcs are disjoint.

In the first case, no stable matching exists.

Phase 2



Def: If $e = uv$ is the 2nd choice of v then vu is a **2-arc**.

Observation: After Phase 1, each vertex u receives at most one 2-arc. Moreover, if u recives a 2-arc then u sends a unique 1-arc that is not bidirected.

Corollary: There is a cycle formed alternatingly by 1-arcs and 2-arcs. This is called a rotation.

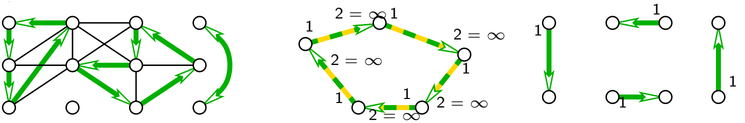
Two cases are possible: Either a rotation is an odd cycle and each arc in it is both a 1-arc and a 2 arc

or all the set of 1-arcs and the set of 2 arcs are disjoint.

In the first case, no stable matching exists.

In the second case, we can delete all 1-arcs of the rotation: no new stable matching is created and not all stable matchings are killed.

Phase 2



Def: If $e = uv$ is the 2nd choice of v then vu is a **2-arc**.

Observation: After Phase 1, each vertex u receives at most one 2-arc. Moreover, if u recives a 2-arc then u sends a unique 1-arc that is not bidirected.

Corollary: There is a cycle formed alternatingly by 1-arcs and 2-arcs. This is called a rotation.

Two cases are possible: Either a rotation is an odd cycle and each arc in it is both a 1-arc and a 2 arc

or all the set of 1-arcs and the set of 2 arcs are disjoint.

In the first case, no stable matching exists.

In the second case, we can delete all 1-arcs of the rotation: no new stable matching is created and not all stable matchings are killed.

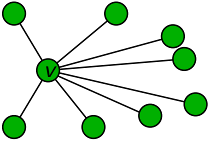
After eliminating this rotation, reversed 2-arcs become 1-arcs.

(And we may execute further GS-deletions.)

One sided markets with choice functions

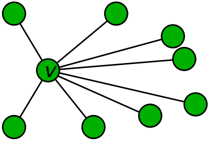
Vertices of the graph are agents

One sided markets with choice functions



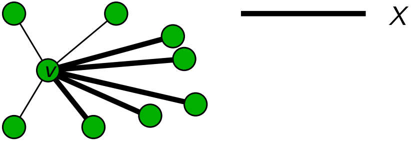
Vertices of the graph are agents edges are possible partnerships (contracts).

One sided markets with choice functions



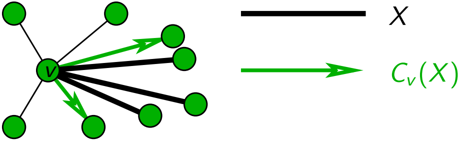
Vertices of the graph are agents edges are possible partnerships (contracts). Each agent v has a choice function $\mathcal{C}_v$:

One sided markets with choice functions



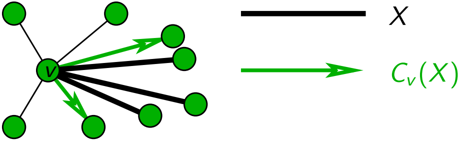
Vertices of the graph are agents edges are possible partnerships (contracts). Each agent v has a choice function $\mathcal{C}_v$: if the set of available options for agent v is X

One sided markets with choice functions



Vertices of the graph are agents edges are possible partnerships (contracts). Each agent v has a choice function $\mathcal{C}_v$: if the set of available options for agent v is X then v selects subset $\mathcal{C}_v(X)$.

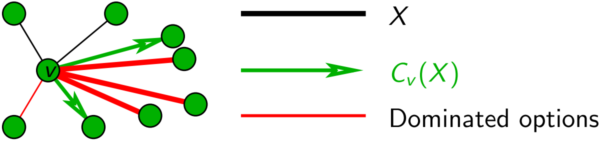
One sided markets with choice functions



Vertices of the graph are agents edges are possible partnerships (contracts). Each agent v has a choice function $\mathcal{C}_v$: if the set of available options for agent v is X then v selects subset $\mathcal{C}_v(X)$.

Def. Option e is dominated by set of options X

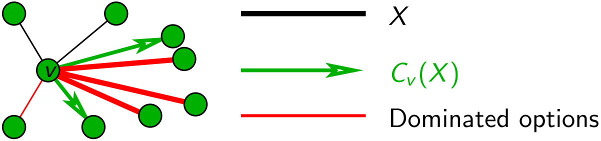
One sided markets with choice functions



Vertices of the graph are agents edges are possible partnerships (contracts). Each agent v has a choice function $\mathcal{C}_v$: if the set of available options for agent v is X then v selects subset $\mathcal{C}_v(X)$.

Def. Option e is dominated by set of options X if option e is ignored when all options in X are available:

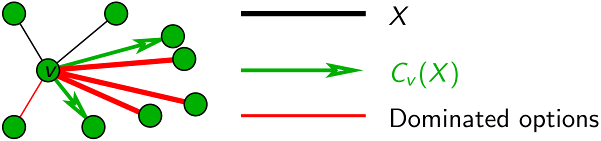
One sided markets with choice functions



Vertices of the graph are agents edges are possible partnerships (contracts). Each agent v has a choice function $\mathcal{C}_v$: if the set of available options for agent v is X then v selects subset $\mathcal{C}_v(X)$.

Def. Option e is dominated by set of options X if option e is ignored when all options in X are available: $e \notin \mathcal{C}_v(X \cup \{e\})$.

One sided markets with choice functions

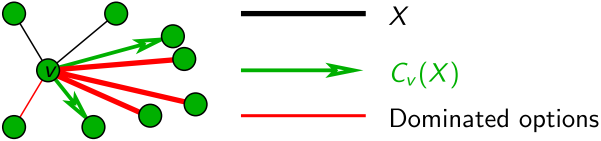


Vertices of the graph are agents edges are possible partnerships (contracts). Each agent v has a choice function $\mathcal{C}_v$: if the set of available options for agent v is X then v selects subset $\mathcal{C}_v(X)$.

Def. Option e is dominated by set of options X if option e is ignored when all options in X are available: $e \notin \mathcal{C}_v(X \cup \{e\})$.

Notation: $\mathcal{D}_v(X)$ is the set of options dominated by X.

One sided markets with choice functions



Vertices of the graph are agents edges are possible partnerships (contracts). Each agent v has a choice function $\mathcal{C}_v$: if the set of available options for agent v is X then v selects subset $\mathcal{C}_v(X)$.

Def. Option e is dominated by set of options X if option e is ignored when all options in X are available: $e \notin \mathcal{C}_v(X \cup \{e\})$.

Notation: $\mathcal{D}_v(X)$ is the set of options dominated by X.

We assume that agents' choice functions $\mathcal{C}_v$ are substitutable. This means that dominance functions $\mathcal{D}_v$ are monotone:

$$X \subseteq Y \Rightarrow \mathcal{D}_v(X) \subseteq \mathcal{D}_v(Y) \ .$$

(Extra choices do not make an ignored option more attractive.)

Stable matchings with choice functions

So $\mathcal{C}_v(X)$ is the best set of options from X, according to the preference order of v. For the stable b-matching problem, $\mathcal{C}_v(X)$ denotes the best $b(v)$ options of X. We assume that all choice functions $\mathcal{C}_v$ are substitutable.
A stable (b-)matching can be defined as a set S of contracts such that

- No contract of S is dominated by other contracts of S.
- S dominates each contract outside S (according to some $\mathcal{D}_v$)

The stable partnership problem is given by a graph G and substitutable choice functions $\mathcal{C}_v$ on the stars.

Stable matchings with choice functions

So $\mathcal{C}_v(X)$ is the best set of options from X, according to the preference order of v. For the stable b-matching problem, $\mathcal{C}_v(X)$ denotes the best $b(v)$ options of X. We assume that all choice functions $\mathcal{C}_v$ are substitutable.

A stable (b-)matching can be defined as a set S of contracts such that

- No contract of S is dominated by other contracts of S.
- S dominates each contract outside S (according to some $\mathcal{D}_v$)

The stable partnership problem is given by a graph G and substitutable choice functions $\mathcal{C}_v$ on the stars. Aim: find a stable partnership, i.e. a subset S of $E(G)$ with the above two properties.

Stable matchings with choice functions

So $\mathcal{C}_v(X)$ is the best set of options from X, according to the preference order of v. For the stable b-matching problem, $\mathcal{C}_v(X)$ denotes the best $b(v)$ options of X. We assume that all choice functions $\mathcal{C}_v$ are substitutable.
A stable (b-)matching can be defined as a set S of contracts such that

- No contract of S is dominated by other contracts of S.
- S dominates each contract outside S (according to some $\mathcal{D}_v$)

The stable partnership problem is given by a graph G and substitutable choice functions $\mathcal{C}_v$ on the stars. Aim: find a stable partnership, i.e. a subset S of $E(G)$ with the above two properties.
Fact: for bipartite graphs, the Gale-Shapley algorithm works.

Stable matchings with choice functions

So $\mathcal{C}_v(X)$ is the best set of options from X, according to the preference order of v. For the stable b-matching problem, $\mathcal{C}_v(X)$ denotes the best $b(v)$ options of X. We assume that all choice functions $\mathcal{C}_v$ are substitutable.
A stable (b-)matching can be defined as a set S of contracts such that

- No contract of S is dominated by other contracts of S.
- S dominates each contract outside S (according to some $\mathcal{D}_v$)

The stable partnership problem is given by a graph G and substitutable choice functions $\mathcal{C}_v$ on the stars. Aim: find a stable partnership, i.e. a subset S of $E(G)$ with the above two properties.
Fact: for bipartite graphs, the Gale-Shapley algorithm works.
For nonbipartite graphs, we can solve only a special case: we assume that each choice function $\mathcal{C}_v$ is increasing, i.e.

$$X \subseteq Y \Rightarrow |\mathcal{C}_v(X)| \leq |\mathcal{C}_v(Y)| \ .$$

(Greater choice set means more choices selected.)

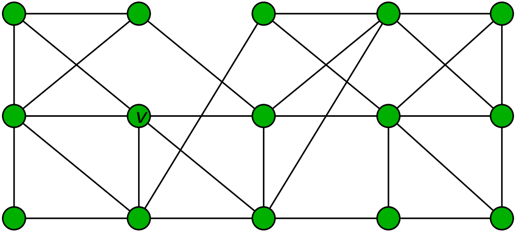
Finding a stable partnership

Generalization of Irving's algorithm: we keep on deleting edges such that

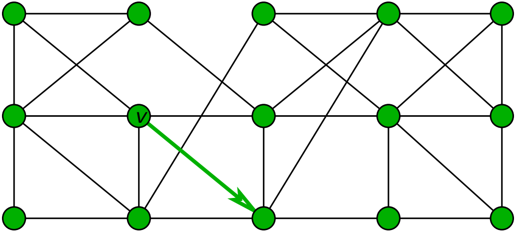
- no new stable partnership is created
- not all stable partnerships are killed

until a single stable partnership remains.
For an ordinary stable roommates problem, the extended algorithm is doing the same as Irving's.

1st phase of the algorithm

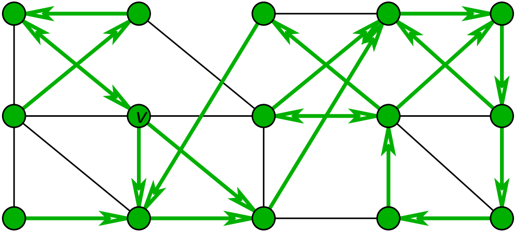


1st phase of the algorithm



Def. Oriented edge $e = vu$ is a **first choice of** v and called a **1-arc** if $e \in \mathcal{C}_v(E(v))$. ($E(v)$ is the set of edges incident with v.)

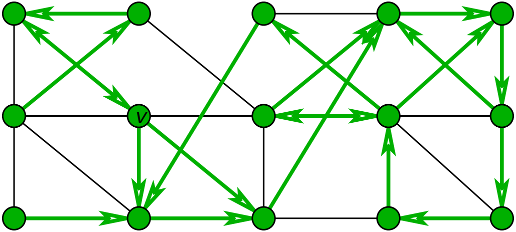
1st phase of the algorithm



Def. Oriented edge $e = vu$ is a **first choice of** v and called a **1-arc** if $e \in \mathcal{C}_v(E(v))$. ($E(v)$ is the set of edges incident with v.)

Proposal step: Find all 1-arcs.

1st phase of the algorithm

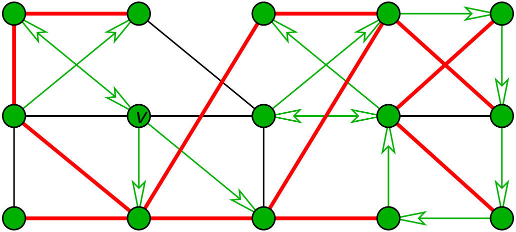


Def. Oriented edge $e = vu$ is a **first choice of** v and called a **1-arc** if $e \in \mathcal{C}_v(E(v))$. ($E(v)$ is the set of edges incident with v.)

Proposal step: Find all 1-arcs.

The set of stable partnerships does not change.

1st phase of the algorithm

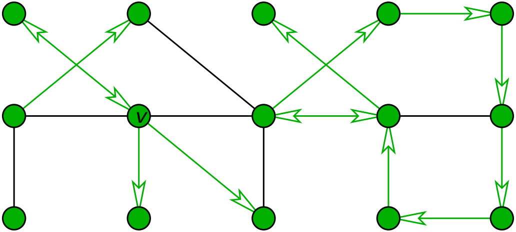


Def. Oriented edge $e = vu$ is a **first choice of** v and called a **1-arc** if $e \in \mathcal{C}_v(E(v))$. ($E(v)$ is the set of edges incident with v.)
Proposal step: Find all 1-arcs.

The set of stable partnerships does not change.

Refusal step: If X is the set of 1-arcs pointing to u, delete $\mathcal{D}_u(X)$.

1st phase of the algorithm



Def. Oriented edge $e = vu$ is a **first choice of v** and called a **1-arc** if $e \in \mathcal{C}_v(E(v))$. ($E(v)$ is the set of edges incident with v.)

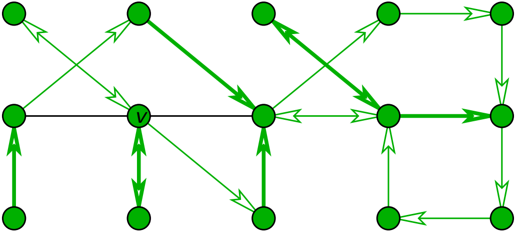
Proposal step: Find all 1-arcs.

The set of stable partnerships does not change.

Refusal step: If X is the set of 1-arcs pointing to u, delete $\mathcal{D}_u(X)$.

The set of stable partnerships does not change.

1st phase of the algorithm



Def. Oriented edge $e = vu$ is a **first choice of** v and called a 1-**arc** if $e \in \mathcal{C}_v(E(v))$. $(E(v)$ is the set of edges incident with v.)

Proposal step: Find all 1-arcs.

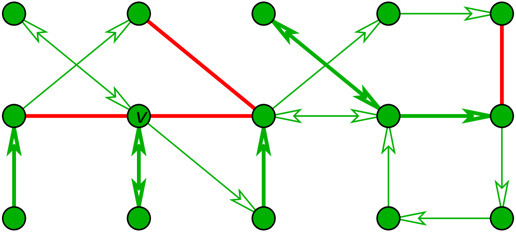
The set of stable partnerships does not change.

Refusal step: If X is the set of 1-arcs pointing to u, delete $\mathcal{D}_u(X)$.

The set of stable partnerships does not change.

Execute proposal and refusal steps alternatingly

1st phase of the algorithm



Def. Oriented edge $e = vu$ is a **first choice of** v and called a 1-**arc** if $e \in \mathcal{C}_v(E(v))$. ($E(v)$ is the set of edges incident with v.)

Proposal step: Find all 1-arcs.

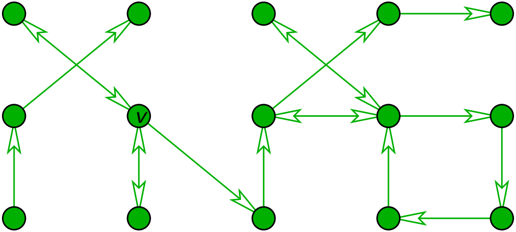
The set of stable partnerships does not change.

Refusal step: If X is the set of 1-arcs pointing to u, delete $\mathcal{D}_u(X)$.

The set of stable partnerships does not change.

Execute proposal and refusal steps alternatingly

1st phase of the algorithm



Def. Oriented edge $e = vu$ is a **first choice of** v and called a 1-**arc** if $e \in \mathcal{C}_v(E(v))$. $(E(v)$ is the set of edges incident with v.)

Proposal step: Find all 1-arcs.

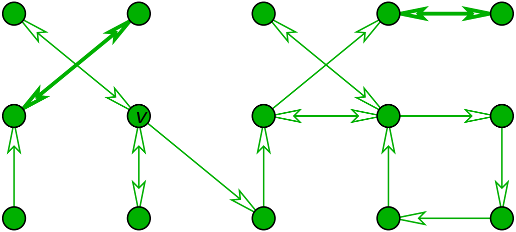
The set of stable partnerships does not change.

Refusal step: If X is the set of 1-arcs pointing to u, delete $\mathcal{D}_u(X)$.

The set of stable partnerships does not change.

Execute proposal and refusal steps alternatingly

1st phase of the algorithm



Def. Oriented edge $e = vu$ is a **first choice of v** and called a 1-**arc** if $e \in \mathcal{C}_v(E(v))$. ($E(v)$ is the set of edges incident with v.)

Proposal step: Find all 1-arcs.

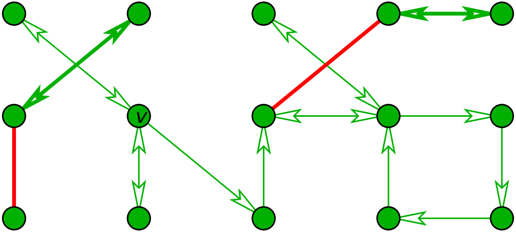
The set of stable partnerships does not change.

Refusal step: If X is the set of 1-arcs pointing to u, delete $\mathcal{D}_u(X)$.

The set of stable partnerships does not change.

Execute proposal and refusal steps alternatingly

1st phase of the algorithm



Def. Oriented edge $e = vu$ is a **first choice of** v and called a 1-**arc** if $e \in \mathcal{C}_v(E(v))$. $(E(v)$ is the set of edges incident with v.)

Proposal step: Find all 1-arcs.

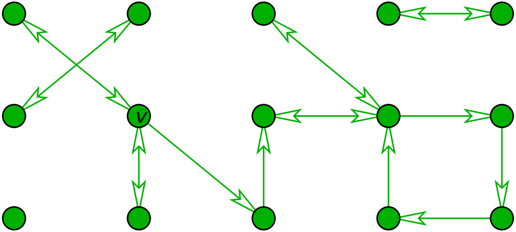
The set of stable partnerships does not change.

Refusal step: If X is the set of 1-arcs pointing to u, delete $\mathcal{D}_u(X)$.

The set of stable partnerships does not change.

Execute proposal and refusal steps alternatingly

1st phase of the algorithm



Def. Oriented edge $e = vu$ is a **first choice of** v and called a **1-arc** if $e \in \mathcal{C}_v(E(v))$. ($E(v)$ is the set of edges incident with v.)

Proposal step: Find all 1-arcs.

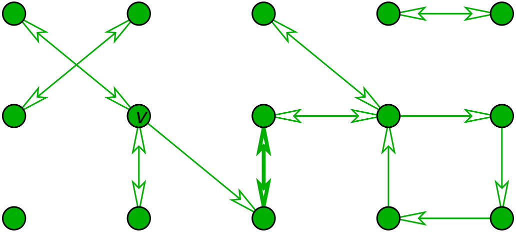
The set of stable partnerships does not change.

Refusal step: If X is the set of 1-arcs pointing to u, delete $\mathcal{D}_u(X)$.

The set of stable partnerships does not change.

Execute proposal and refusal steps alternatingly

1st phase of the algorithm



Def. Oriented edge $e = vu$ is a **first choice of** v and called a 1-**arc** if $e \in \mathcal{C}_v(E(v))$. ($E(v)$ is the set of edges incident with v.)

Proposal step: Find all 1-arcs.

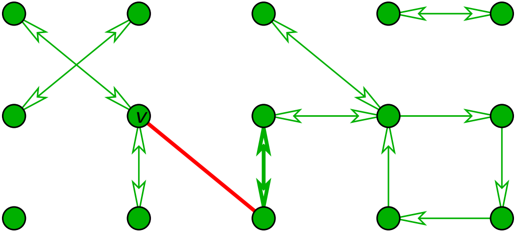
The set of stable partnerships does not change.

Refusal step: If X is the set of 1-arcs pointing to u, delete $\mathcal{D}_u(X)$.

The set of stable partnerships does not change.

Execute proposal and refusal steps alternatingly

1st phase of the algorithm



Def. Oriented edge $e = vu$ is a **first choice of** v and called a 1-**arc** if $e \in \mathcal{C}_v(E(v))$. ($E(v)$ is the set of edges incident with v.)

Proposal step: Find all 1-arcs.

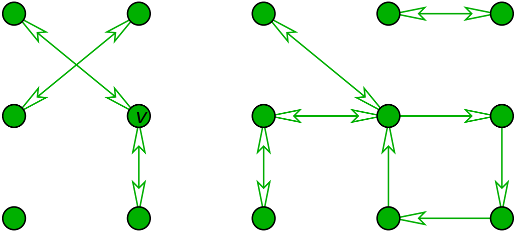
The set of stable partnerships does not change.

Refusal step: If X is the set of 1-arcs pointing to u, delete $\mathcal{D}_u(X)$.

The set of stable partnerships does not change.

Execute proposal and refusal steps alternatingly

1st phase of the algorithm



Def. Oriented edge $e = vu$ is a **first choice of** v and called a 1-**arc** if $e \in \mathcal{C}_v(E(v))$. ($E(v)$ is the set of edges incident with v.)

Proposal step: Find all 1-arcs.

The set of stable partnerships does not change.

Refusal step: If X is the set of 1-arcs pointing to u, delete $\mathcal{D}_u(X)$.

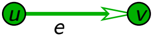
The set of stable partnerships does not change.

Execute proposal and refusal steps alternatingly until no further step is possible. Now 1-arcs form an Eulerian graph.

If all 1-arcs are bidirected then it is the only stable partnership.

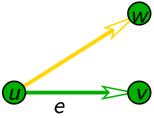
Otherwise we move on to the 2nd phase of the algorithm.

Finding a rotation from replacements



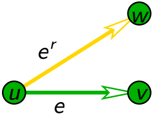
Let $e = uv$ be a 1-arc.

Finding a rotation from replacements



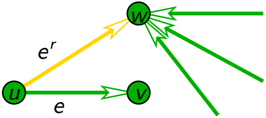
Let $e = uv$ be a 1-arc. The replacement of e is the contract that u selects instead of e if e is not available any more,

Finding a rotation from replacements



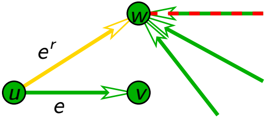
Let $e = uv$ be a 1-arc. The replacement of e is the contract that u selects instead of e if e is not available any more, that is, $e^r = C_u(E(u) \setminus \{e\}) \setminus C_u(E(u))$.

Finding a rotation from replacements



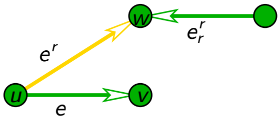
Let $e = uv$ be a 1-arc. The replacement of e is the contract that u selects instead of e if e is not available any more, that is,
$e^r = C_u(E(u) \setminus \{e\}) \setminus C_u(E(u))$.
Let $e^r = uw$ and A^- be the set of 1-arcs that point to w.

Finding a rotation from replacements



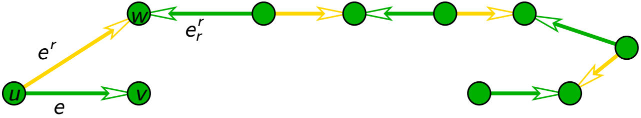
Let $e = uv$ be a 1-arc. The replacement of e is the contract that u selects instead of e if e is not available any more, that is, $e^r = C_u(E(u) \setminus \{e\}) \setminus C_u(E(u))$.
Let $e^r = uw$ and A^- be the set of 1-arcs that point to w.
$A^- \cup \{e^r\}$ dominates a unique 1-arc e_r^r.

Finding a rotation from replacements



Let $e = uv$ be a 1-arc. The replacement of e is the contract that u selects instead of e if e is not available any more, that is,
$e^r = C_u(E(u) \setminus \{e\}) \setminus C_u(E(u))$.
Let $e^r = uw$ and A^- be the set of 1-arcs that point to w.
$A^- \cup \{e^r\}$ dominates a unique 1-arc e_r^r.

Finding a rotation from replacements

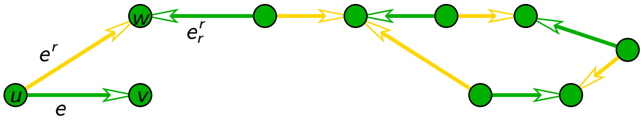


Let $e = uv$ be a 1-arc. The replacement of e is the contract that u selects instead of e if e is not available any more, that is, $e^r = C_u(E(u) \setminus \{e\}) \setminus C_u(E(u))$.
Let $e^r = uw$ and A^- be the set of 1-arcs that point to w.
$A^- \cup \{e^r\}$ dominates a unique 1-arc e_r^r.
Alternating sequence

$$e, e^r, e_r^r, (e_r^r)^r, (e_r^r)_r^r, \ldots$$

of 1-arcs and replacements

Finding a rotation from replacements

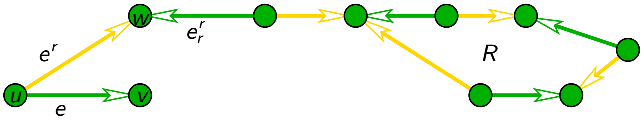


Let $e = uv$ be a 1-arc. The replacement of e is the contract that u selects instead of e if e is not available any more, that is, $e^r = C_u(E(u) \setminus \{e\}) \setminus C_u(E(u))$.
Let $e^r = uw$ and A^- be the set of 1-arcs that point to w.
$A^- \cup \{e^r\}$ dominates a unique 1-arc e_r^r.
Alternating sequence

$$e, e^r, e_r^r, (e_r^r)^r, (e_r^r)_r^r, \ldots$$

of 1-arcs and replacements sooner or later repeats a 1-arc.

Finding a rotation from replacements



Let $e = uv$ be a 1-arc. The replacement of e is the contract that u selects instead of e if e is not available any more, that is, $e^r = C_u(E(u) \setminus \{e\}) \setminus C_u(E(u))$.
Let $e^r = uw$ and A^- be the set of 1-arcs that point to w.
$A^- \cup \{e^r\}$ dominates a unique 1-arc e_r^r.
Alternating sequence

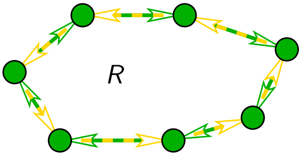
$$e, e^r, e_r^r, (e_r^r)^r, (e_r^r)_r^r, \ldots$$

of 1-arcs and replacements sooner or later repeats a 1-arc.
Hence we find a rotation R.

Rotation elimination

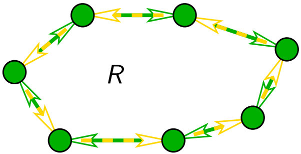
Case 1. 1-arcs and their replacements in R form identical sets.

Rotation elimination



Case 1. 1-arcs and their replacements in R form identical sets.
Rotation R is an odd cycle, and the algorithm stops:

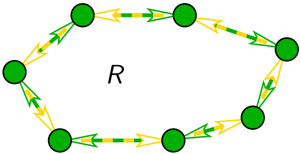
Rotation elimination



Case 1. 1-arcs and their replacements in R form identical sets.
Rotation R is an odd cycle, and the algorithm stops:

No stable partnership exists.

Rotation elimination

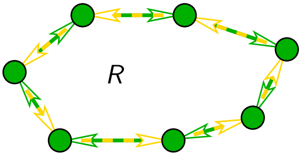


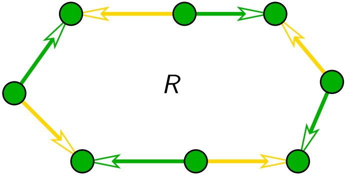
Case 1. 1-arcs and their replacements in R form identical sets.
Rotation R is an odd cycle, and the algorithm stops:

No stable partnership exists.

Case 2. 1-arcs and their replacements in R are distinct.

Rotation elimination



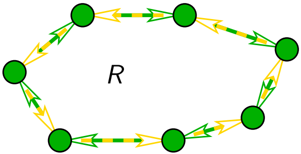


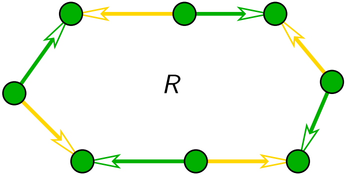
Case 1. 1-arcs and their replacements in R form identical sets.
Rotation R is an odd cycle, and the algorithm stops:

No stable partnership exists.

Case 2. 1-arcs and their replacements in R are distinct.
R is an even alternating cycle.

Rotation elimination



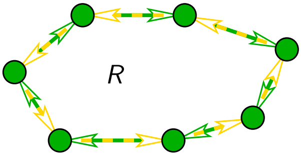


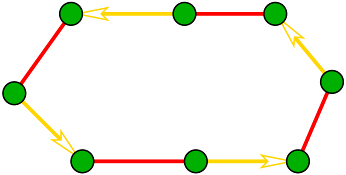
Case 1. 1-arcs and their replacements in R form identical sets.
Rotation R is an odd cycle, and the algorithm stops:

No stable partnership exists.

Case 2. 1-arcs and their replacements in R are distinct.
R is an even alternating cycle.
We eliminate the rotation:

Rotation elimination





Case 1. 1-arcs and their replacements in R form identical sets.
Rotation R is an odd cycle, and the algorithm stops:

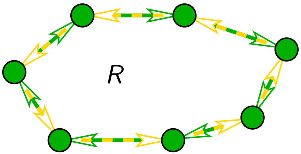
No stable partnership exists.

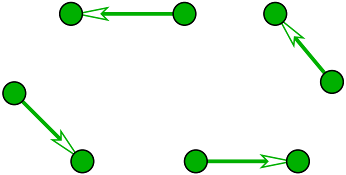
Case 2. 1-arcs and their replacements in R are distinct.
R is an even alternating cycle.

We eliminate the rotation:

- Delete all 1-arcs of the rotation

Rotation elimination





Case 1. 1-arcs and their replacements in R form identical sets.
Rotation R is an odd cycle, and the algorithm stops:

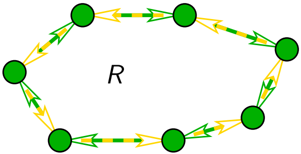
No stable partnership exists.

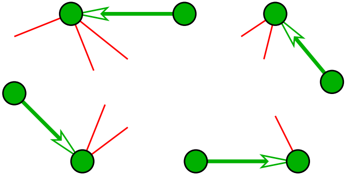
Case 2. 1-arcs and their replacements in R are distinct.
R is an even alternating cycle.

We eliminate the rotation:

- Delete all 1-arcs of the rotation
- Replacements become 1-arcs.

Rotation elimination





Case 1. 1-arcs and their replacements in R form identical sets.
Rotation R is an odd cycle, and the algorithm stops:

No stable partnership exists.

Case 2. 1-arcs and their replacements in R are distinct.
R is an even alternating cycle.

We eliminate the rotation:

- Delete all 1-arcs of the rotation
- Replacements become 1-arcs.
- Execute a refusal step at each terminal of the new 1-arcs.

2nd phase of the algorithm

Theorem

After a rotation elimination, no new stable partnership is created and not **all** *stable partnerships are killed.*

How does the algorithm terminate?

Theorem

If there are no more rotations then all edges are bidirected 1*-arcs, hence the graph itself is a stable partnership.*

2nd phase of the algorithm

Theorem

After a rotation elimination, no new stable partnership is created and not **all** *stable partnerships are killed.*

How does the algorithm terminate?

Theorem

If there are no more rotations then all edges are bidirected 1*-arcs, hence the graph itself is a stable partnership.*

Complexity?

Theorem

The generalization of Irving's algorithm needs $O(n+m)$ $\mathcal{C}$*-calls and* $O(n+m)$ $\mathcal{D}$*-calls.*

Thank you for the attention!