PARETO OPTIMAL MATCHINGS

Katarína Cechlárová, UPJŠ Košice, Slovakia

BASIC SETTING

A is the set of m agents,
C is the set of n objects.

Each agents

- consumes at most one object
- has strict preferences over objects.

$I = (A, C, \mathcal{P})$ is an instance of the matching problem.

$P(a_1) : c_4, c_3, c_2, c_7, c_5$
$P(a_2) : c_1, c_3, c_6, c_7$
$P(a_3) : c_2, c_5, c_6, c_4, c_1$
$P(a_4) : c_1, c_3, c_4, c_2$
$P(a_5) : c_4, c_1, c_2$
$P(a_6) : c_4, c_2$
$P(a_7) : c_1, c_3, c_4$

Matching: M_1

$$\left\{ \binom{a_1}{c_2}, \binom{a_2}{c_1}, \binom{a_3}{c_6}, \binom{a_4}{c_4}, \binom{a_7}{c_3} \right\}$$

Possible applications:

- tenants and houses
- workers and positions
- researchers and offices
- students and courses
- ice-hockey teams and players

Which matching is optimal?

A matching M' dominates a matching M if at least one applicant prefers M' to M and no applicant prefers M to M'.

A matching M is Pareto optimal if it is not dominated by any other matching.

SERIAL DICTATORSHIP SD

Agents are ordered into a picking sequence (policy) σ.
Each agent on her turn according to σ picks her most preferred available object.

$P(a_1): c_4, c_3, c_2, c_7, c_5$
$P(a_2): c_1, c_3, c_6, c_7$
$P(a_3): c_2, c_5, c_6, c_4, c_1$
$P(a_4): c_1, c_3, c_4, c_2$
$P(a_5): c_4, c_1, c_2$
$P(a_6): c_4, c_2$
$P(a_7): c_1, c_3, c_4$

Policy $\sigma_1 = a_1, a_2, \ldots, a_7$

Matching: M_{SD1}

$\left\{ \binom{a_1}{c_4}, \binom{a_2}{c_1}, \binom{a_3}{c_2}, \binom{a_4}{c_3} \right\}$

Size of M_{SD1}: 4

$P(a_1): c_4, c_3, c_2, c_7, c_5$
$P(a_2): c_1, c_3, c_6, c_7$
$P(a_3): c_2, c_5, c_6, c_4, c_1$
$P(a_4): c_1, c_3, c_4, c_2$
$P(a_5): c_4, c_1, c_2$
$P(a_6): c_4, c_2$
$P(a_7): c_1, c_3, c_4$

Policy $\sigma_2 = a_7, a_6, \ldots, a_1$

Matching: M_{SD2}

$\left\{ \binom{a_1}{c_7}, \binom{a_2}{c_6}, \binom{a_3}{c_5}, \binom{a_4}{c_3}, \binom{a_5}{c_2}, \binom{a_6}{c_4}, \binom{a_7}{c_1} \right\}$

Size of M_{SD2}: 7

PROPERTIES OF SERIAL DICTATORSHIP

Theorem 1. SD produces a POM for any policy.

Theorem 2. SD is strategy-proof.

Theorem 3. SD can produce any POM.

Proved by:
Svensson 1994, Abdulkadiroğlu & Sönmez 1998, Abraham, KC, Manlove & Mehlhorn 2004, Brams & King 2005

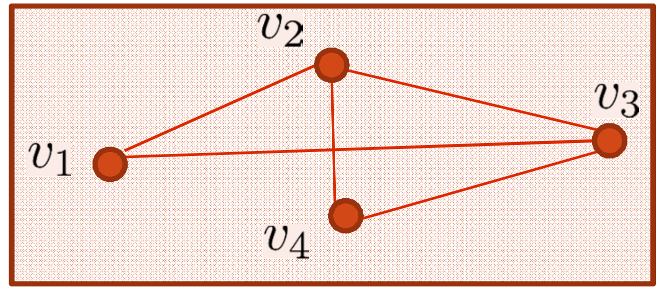
Characterization of POM:

- maximal
- trade-in free
- coalition-free

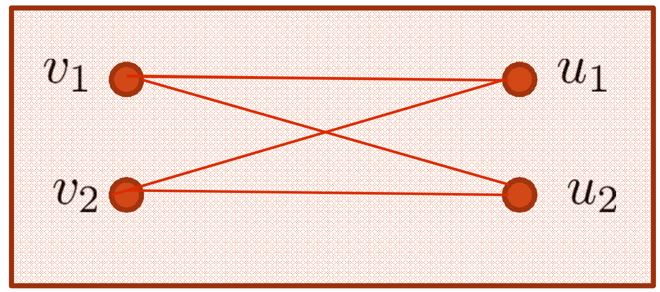
MINITUTORIAL ON GRAPHS 1

Graph is a pair (V, E); V is the set of vertices and E is the set of edges (arcs).

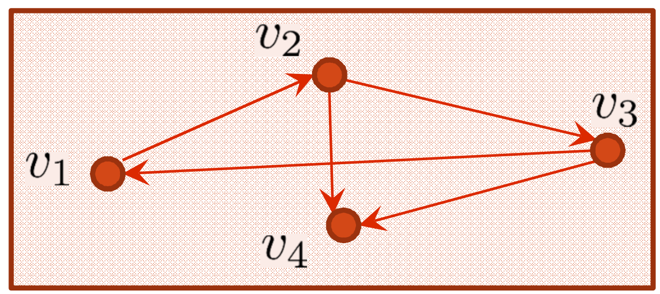
Undirected graph:
edges are unordered pairs of vertices



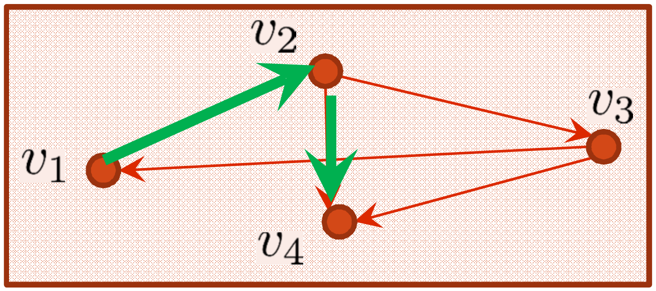
Bipartite graph:
vertices partitioned into sets V, U,
edges are only between V and U

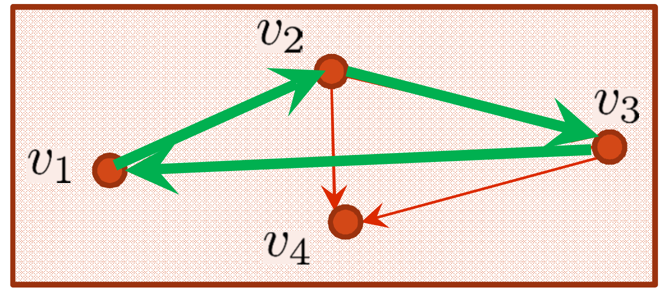


Directed graph:
arcs are ordered pairs of vertices



MINITUTORIAL ON GRAPHS 2

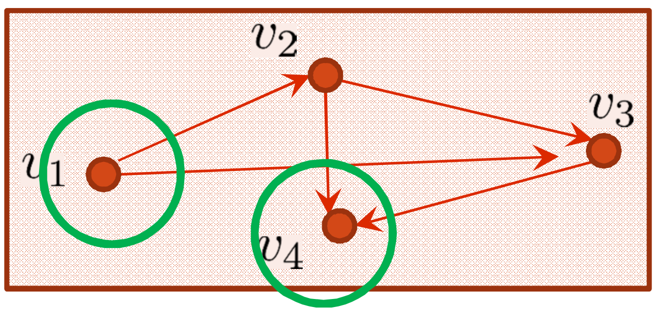




Path

Cycle

A directed graph that contains no cycle is called acyclic



An acyclic graph contains:

a source: vertex with no incomming arcs

a sink: vertex with no outgoing arcs

An acyclic directed graph admits a topological labelling of vertices $\sigma : V \to \mathbb{N}$: if there is an arc $i \to j$ then $\sigma(i) > \sigma(j)$

Algorithm: give a sink v the minimum possible label; delete its incomming arcs and repeat.

CHARACTERIZATION OF POM

- maximal $\Longrightarrow$ no pair can be added
- trade-in free
- coalition-free

Acceptability graph $G(I)$: vertices are agents and objects

Matching: set of edges; no two have a vertex in common

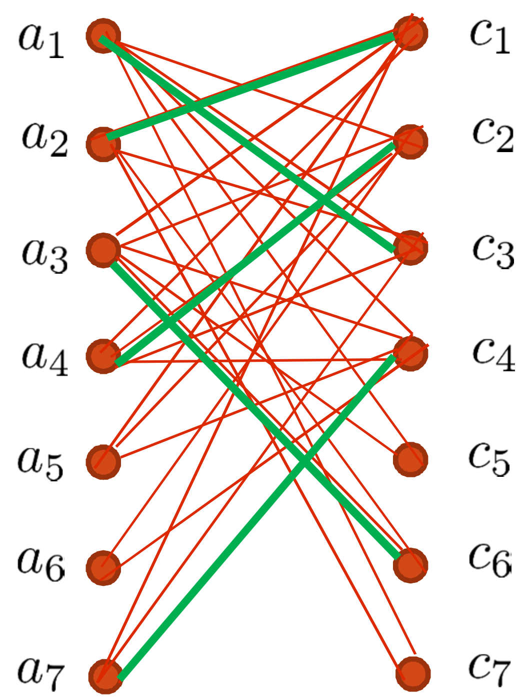
Maximal matching: no edge can be added

Maximum matching: matching with maximum cardinality

$P(a_1) : c_4, c_3, c_2, c_7, c_5$
$P(a_2) : c_1, c_3, c_6, c_7$
$P(a_3) : c_2, c_5, c_6, c_4, c_1$
$P(a_4) : c_1, c_3, c_4, c_2$
$P(a_5) : c_4, c_1, c_2$
$P(a_6) : c_4, c_2$
$P(a_7) : c_1, c_3, c_4$

Matching: M_1

$\left\{ \binom{a_1}{c_3}, \binom{a_2}{c_1}, \binom{a_3}{c_6}, \binom{a_4}{c_2}, \binom{a_7}{c_4} \right\}$



CHARACTERIZATION OF POM

- maximal
- trade-in free $\Longrightarrow$ no matched agent can move to a preferred free object
- coalition-free

$P(a_1) : c_4, c_3, c_2, c_7, c_5$
$P(a_2) : c_1, c_3, c_6, c_7$
$P(a_3) : c_2, c_5, c_6, c_4, c_1$
$P(a_4) : c_1, c_3, c_4, c_2$
$P(a_5) : c_4, c_1, c_2$
$P(a_6) : c_4, c_2$
$P(a_7) : c_1, c_3, c_4$

M_1 is
not trade-in free:
a_3 can move to c_5

Matching: M_2

$$\left\{\binom{a_1}{c_3}, \binom{a_2}{c_1}, \binom{a_3}{c_5}, \binom{a_4}{c_2}, \binom{a_7}{c_4}\right\}$$

CHARACTERIZATION OF POM

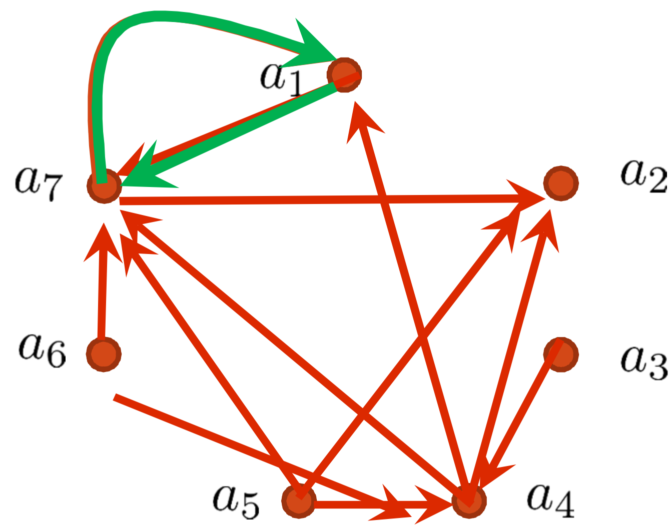
- maximal
- trade-in free
- coalition-free $\Longrightarrow$ no coalition of agents can exchange their objects

Envy graph $G(M_2)$: vertices are agents

Arc $a_i \to a_j$ if a_j has an object that a_i prefers to $M(a_i)$

M admits a coalition if and only if $G(M)$ contains a cycle.

$P(a_1) : c_4, c_3, c_2, c_7, c_5$
$P(a_2) : c_1, c_3, c_6, c_7$
$P(a_3) : c_2, c_5, c_6, c_4, c_1$
$P(a_4) : c_1, c_3, c_4, c_2$
$P(a_5) : c_4, c_1, c_2$
$P(a_6) : c_4, c_2$
$P(a_7) : c_1, c_3, c_4$



Matching: M_2

$\left\{ \binom{a_1}{c_3}, \binom{a_2}{c_1}, \binom{a_3}{c_5}, \binom{a_4}{c_2}, \binom{a_7}{c_4} \right\}$

CHARACTERIZATION OF POM

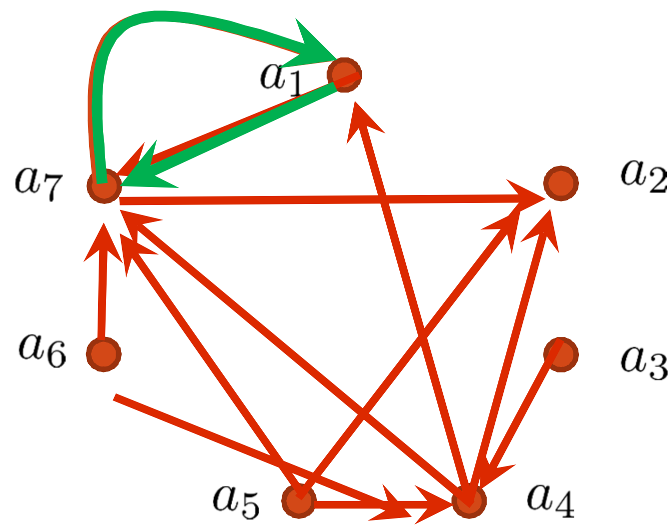
- maximal
- trade-in free
- coalition-free $\Longrightarrow$ no coalition of agents can exchange their objects

Envy graph $G(M_2)$: vertices are agents

Arc $a_i \to a_j$ if a_j has an object that a_i prefers to $M(a_i)$

M admits a coalition if and only if $G(M)$ contains a cycle.

$P(a_1) : c_4, c_3, c_2, c_7, c_5$
$P(a_2) : c_1, c_3, c_6, c_7$
$P(a_3) : c_2, c_5, c_6, c_4, c_1$
$P(a_4) : c_1, c_3, c_4, c_2$
$P(a_5) : c_4, c_1, c_2$
$P(a_6) : c_4, c_2$
$P(a_7) : c_1, c_3, c_4$



Matching: M_3

$\left\{ \binom{a_1}{c_4}, \binom{a_2}{c_1}, \binom{a_3}{c_5}, \binom{a_4}{c_2}, \binom{a_7}{c_3} \right\}$

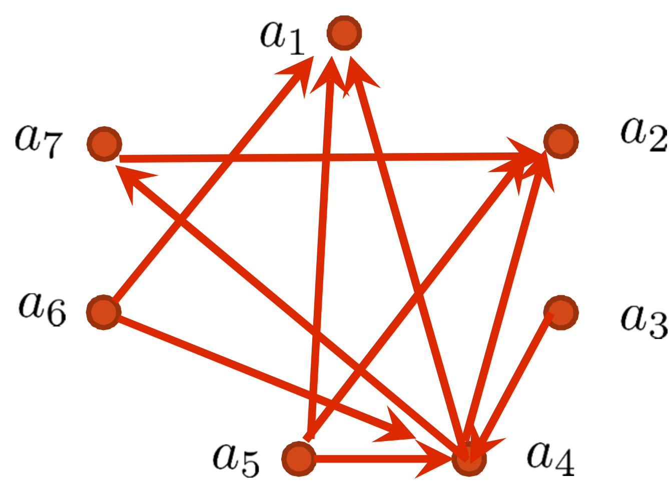
Coalition (a_1, a_7)

CHARACTERIZATION OF POM

- maximal
- trade-in free
- coalition-free $\Longrightarrow$ no coalition can profitably exchange their houses

Envy graph $G(M_3)$ $\Longrightarrow$ is acyclic

$P(a_1) : c_4, c_3, c_2, c_7, c_5$
$P(a_2) : c_1, c_3, c_6, c_7$
$P(a_3) : c_2, c_5, c_6, c_4, c_1$
$P(a_4) : c_1, c_3, c_4, c_2$
$P(a_5) : c_4, c_1, c_2$
$P(a_6) : c_4, c_2$
$P(a_7) : c_1, c_3, c_4$



Matching: M_3

$\left\{ \binom{a_1}{c_4}, \binom{a_2}{c_1}, \binom{a_3}{c_5}, \binom{a_4}{c_2}, \binom{a_7}{c_3} \right\}$

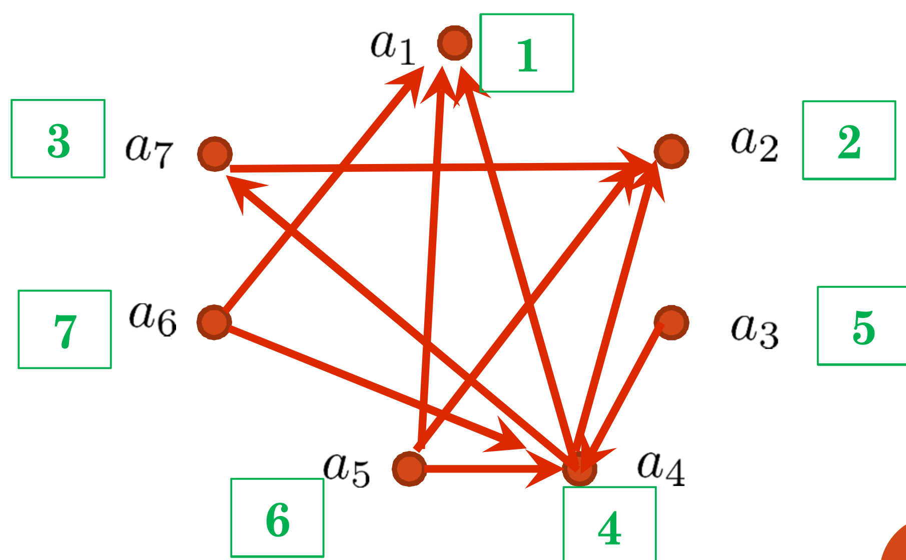
CHARACTERIZATION OF POM

- maximal
- trade-in free
- coalition-free $\Longrightarrow$ no coalition can profitably exchange their houses

Envy graph $G(M_3) \Longrightarrow$ is acyclic

$\Longrightarrow G(M_3)$ admits a *topological ordering* σ

$P(a_1): c_4, c_3, c_2, c_7, c_5$
$P(a_2): c_1, c_3, c_6, c_7$
$P(a_3): c_2, c_5, c_6, c_4, c_1$
$P(a_4): c_1, c_3, c_4, c_2$
$P(a_5): c_4, c_1, c_2$
$P(a_6): c_4, c_2$
$P(a_7): c_1, c_3, c_4$



Matching: M_3

$$\left\{ \binom{a_1}{c_4}, \binom{a_2}{c_1}, \binom{a_3}{c_5}, \binom{a_4}{c_2}, \binom{a_7}{c_3} \right\}$$

$\Longrightarrow \sigma$ gives a policy to obtain M_3

$\sigma = (a_1, a_2, a_7, a_4, a_3, a_5, a_6)$

ALTERNATIVE TESTING FOR COALITIONS

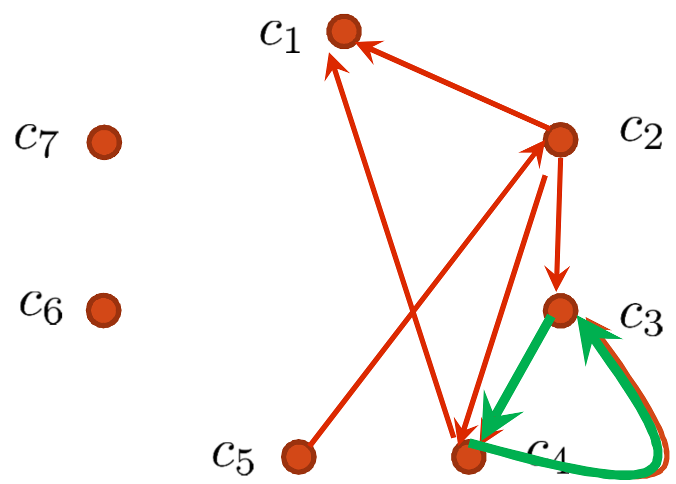
Aziz et al., Optimal Reallocation under Additive and Ordinal Preferences, 2016

Object improvement graph $\bar{G}(M_2)$: vertices are objects

Arc $c_i \to c_j$ if there exists an agent a who prefers c_j to $c_i = M(a)$

M admits a coalition if and only if $\bar{G}(M)$ contains a cycle.

$P(a_1) : c_4, c_3, c_2, c_7, c_5$
$P(a_2) : c_1, c_3, c_6, c_7$
$P(a_3) : c_2, c_5, c_6, c_4, c_1$
$P(a_4) : c_1, c_3, c_4, c_2$
$P(a_5) : c_4, c_1, c_2$
$P(a_6) : c_4, c_2$
$P(a_7) : c_1, c_3, c_4$



Matching: M_2

$\left\{ \binom{a_1}{c_3}, \binom{a_2}{c_1}, \binom{a_3}{c_5}, \binom{a_4}{c_2}, \binom{a_7}{c_4} \right\}$

FINDING A MAXIMUM CARDINALITY POM

1. Find a maximum cardinality matching M_1 in acceptability graph $G(I)$.
2. Use all possible trade-ins to get a matching M_2.
3. Satisfy all coalitions to get a matching M_3.

Steps 2,3:
size of matching cannot decrease
Pareto optimality is ensured

$P(a_1) : c_4, c_3, c_2, c_7, c_5$	$P(a_1) : c_4, c_3, c_2, c_7, c_5$
$P(a_2) : c_1, c_3, c_6, c_7$	$P(a_2) : c_1, c_3, c_6, c_7$
$P(a_3) : c_2, c_5, c_6, c_4, c_1$	$P(a_3) : c_2, c_5, c_6, c_4, c_1$
$P(a_4) : c_1, c_3, c_4, c_2$	$P(a_4) : c_1, c_3, c_4, c_2$
$P(a_5) : c_4, c_1, c_2$	$P(a_5) : c_4, c_1, c_2$
$P(a_6) : c_4, c_2$	$P(a_6) : c_4, c_2$
$P(a_7) : c_1, c_3, c_4$	$P(a_7) : c_1, c_3, c_4$

Given an agent a and object c:
$\textsc{pos}(a,c)$: Does there exist a policy σ such that $M_{SD(\sigma)}(a) = c$?
$\textsc{nec}(a,c)$: Is it true that $M_{SD(\sigma)}(a) = c$ for each policy σ?

Saban & Sethuraman 2013:
$\textsc{pos}(a,c)$ is NP-complete and $\textsc{nec}(a,c)$ is polynomial if each agent finds all objects acceptable.

Further results: Aziz, Brand, Brill, Mestre 2014

MANY-TO-MANY MATCHINGS

Course allocation problem: applicants+courses, various feasibility constraints

EXISTENCE OF A POM

Take the set of all feasible matchings $\mathcal{M}$.

Create a partial order $\succeq$ on $\mathcal{M}$:

$M \succeq M'$ if $M(a) \succeq M'(a)$ each agent a and $M'(a) \succ M(a)$ for no agent a.

As $\mathcal{M}$ is finite, $(\mathcal{M}, \succ)$ has $\succeq$-maximal elements $\Longrightarrow$ they correspond to POM.

TESTING FOR PARETO OPTIMALITY

coNP-complete in many settings

EXTENDING PREFERENCES

Agents have preferences over individual objects, need to compare sets.

Agent a: (strictly) prefers object c to object c': notation $c \succ_a c'$

is indifferent between objects c and c': notation $c \sim_a c'$

weakly prefers object c to object c': notation $c \succeq_a c'$

Minimal requirement for the preference extension: responsiveness

Two most common set preferences:

Additive: agent a has utility $u_a(c)$ for each object $c \in C$

a prefers set S to set T if $\sum_{c \in S} u_a(c) > \sum_{c \in T} u_a(c)$

Lexicographic: agent a prefers set S to set T if the most preferred object in the symmetric difference $S \oplus T$ belongs to S

Characteristic vector χ_a^S: entries ordered according to a' preferences

$$\chi_a^S(c) = \begin{cases} 1 & \text{if } c \in S \\ 0 & \text{otherwise} \end{cases}$$

a prefers set S to set T if χ_a^S is lexicographically greater than χ_a^T.

Example: $P(a): c_1, c_2, c_4, c_5, c_3$ — $S = \{c_1, c_4\}$; $\chi_a^s = (1,0,1,0,0)$

$P(a): c_1, c_2, c_4, c_5, c_3$ — $T = \{c_1, c_5, c_3\}$; $\chi_a^T = (1,0,0,1,1)$

$\Longrightarrow$ a prefers S to T

EXAMPLES OF FEASIBILITY CONSTRAINTS

(i) **Capacity constraints.** A bundle of courses is feasible for applicant a with capacity $q(a)$ if and only if its size does not exceed this capacity.

(ii) **Partition constraints.** Suppose applicant a partitions the set of courses into disjoint classes $C_1^a, C_2^a, \dots, C_r^a$ and applicant a has nonnegative *partial quotas* $q_1(a), \dots, q_r(a)$ that denote the maximum number of courses from each class that she is willing to attend.

(iii) **Conflict-free constraints.** Applicant cannot attend courses scheduled in the same time. This can be modelled by a *conflict* graph: vertices=courses, edge between two courses if their times overlap. Feasible bundles of courses correspond to independent sets of vertices of this graph.

(iv) **Price-budget constraints**. Each course c has a nonnegative price $p(c)$, applicant a has a budget $b(a)$. Set of courses is feasible if its total price does not exceed the budget.

Downward closed feasible sets: a matching with lexicographic preferences is a POM iff it can be obtained by a modified sequential allocation mechanism.

Finding a POM in the price-budget case with additive preferences is NP-hard.

KC, Eirinakis, Fleiner, Magos, Mourtos, Potpinková: Pareto optimality in many-to-many matching problems, Discrete Optimization 14 (2014), 160-169.

INDIFFERENCES

Svenson 1994: Serial dictatorship may output a matching that is not a POM.

$P(a_1) : (c_1, c_2)$ Policy $\sigma = a_1, a_2$

$P(a_2) : c_1$ Matching: $M = \{(a_1, c_1)\}$ is dominated by: $M' = \{(a_1, c_2), (a_2, c_1)\}$

Krysta, Manlove, Rastegari, Zhang, *Size versus truthfulness in the House Allocation problem*, 2015: combination of SD with augmenting paths technique

We shall deal with the many-to-many generalization.

K. C., P. Eirinakis, T. Fleiner, D. Magos, D. Manlove, I. Mourtos, E. Oceľáková, B. Rastegari, Pareto optimal matchings in many-to-many markets with ties, Algorithmic Game Theory, SAGT 2015, LNCS 9347, 27-42, 2015.

An instance of many-to-many matching problem: $I = (A, C, \mathcal{P})$ where
A is the set of agents, each has quota $q(a)$
C is the set of objects, each has quota $q(c)$
$\mathcal{P}$ are the preferences of agents over objects, may contain indifferences

agent	quota	preference list	object	quota
a_1	2	$(c_1, c_2), c_3$	c_1	3
a_2	3	$c_2, (c_1, c_3)$	c_2	1
a_3	2	c_3, c_2, c_1	c_3	1

LEXICOGRAPHIC PREFERENCES

If basic preferences are strict, then so are lexicographic preferences over sets.
What about indifferences?

agent	quota	preference list	object	quota
a_1	2	$(c_1, c_2), c_3$	c_1	3
a_2	3	$c_2, (c_1, c_3)$	c_2	1
a_3	2	c_3, c_2, c_1	c_3	1

Indifference classes (ties)
$P(a) : (C_1^a, C_2^a, \ldots, C_k^a)$

Generalized characteristic vector χ_a^S: entries are $(|C_1^a \cap S|, |C_2^a \cap S|, \ldots, |C_k^a \cap S|)$
Agent a prefers set S to set T if χ_a^S is lexicographically greater than χ_a^T.

Example: $P(a) : (c_1, c_2), (c_4, c_5), c_3$ $S = \{c_2, c_5\};\ \chi_a^S = (1, 1, 0)$

$P(a) : (c_1, c_2), (c_4, c_5), c_3$ $T = \{c_1, c_4, c_5\};\ \chi_a^T = (0, 2, 1)$

$\Longrightarrow$ a prefers set S to set T

Algorithm Generalized Serial Dictatorship with Ties GSDT uses network flows.

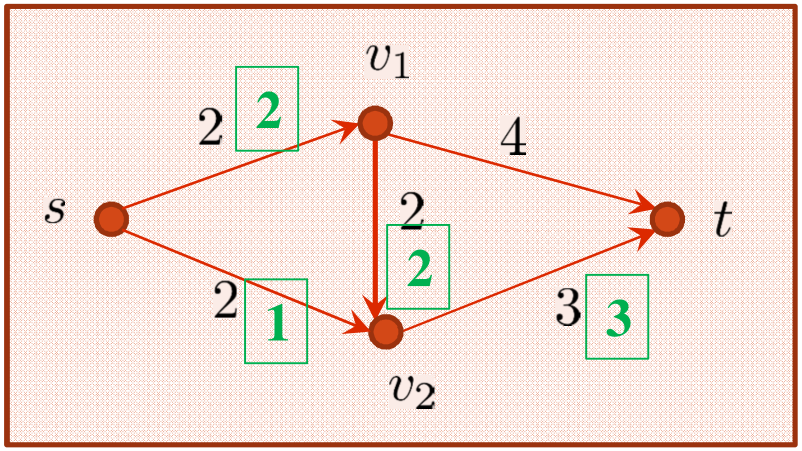
MINITUTORIAL ON NETWORK FLOWS

Network is a pair $N = (G, w)$ where $G = (V, E)$ is a directed graph with a source s and sink t and $w : E \to \mathbb{N}$ are capacities of arcs.

Flow in N:
function $f : E \to \mathbb{R}^+$ that fulfils:

capacity constraints:
$f(e) \leq w(e)$ for each arc e

flow conservation:
inflow=outflow for each vertex $\neq s, t$



MINITUTORIAL ON NETWORK FLOWS

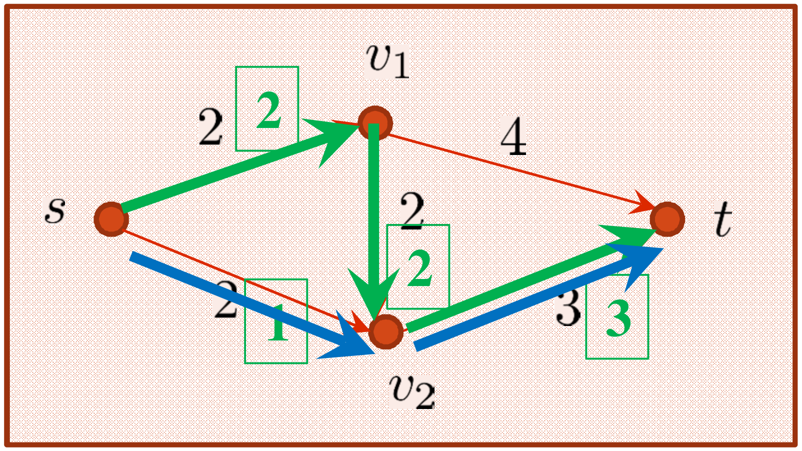
Network is a pair $N = (G, w)$ where $G = (V, E)$ is a directed graph with a source s and sink t and $w : E \to \mathbb{N}$ are capacities of arcs.

Flow in N:
function $f : E \to \mathbb{R}^+$ that fulfils:

capacity constraints:
$f(e) \leq w(e)$ for each arc e

flow conservation:
inflow=outflow for each vertex $\neq s, t$

Size of flow $v(f)$:
sum of outflows from s



Each flow f can be partitioned into $v(f)$ $s-t$ paths

Maximum flow: flow with maximum size

MINITUTORIAL ON NETWORK FLOWS

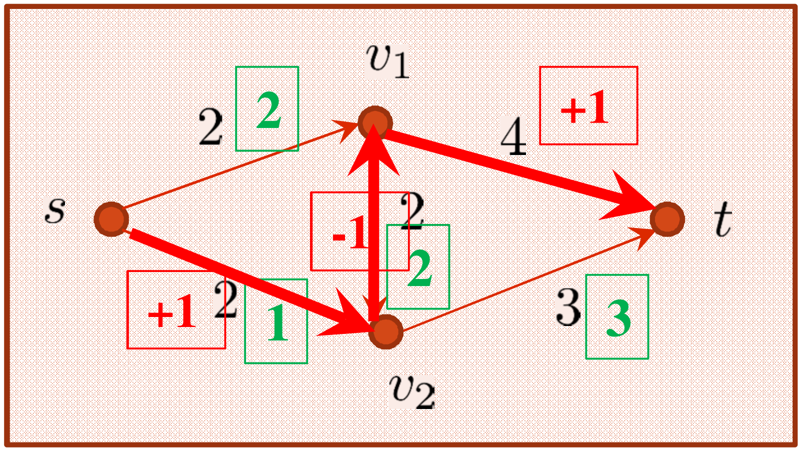
Network is a pair $N = (G, w)$ where $G = (V, E)$ is a directed graph with a source s and sink t and $w : E \to \mathbb{N}$ are capacities of arcs.

Flow in N:
function $f : E \to \mathbb{R}^+$ that fulfils:

capacity constraints:
$f(e) \leq w(e)$ for each arc e

flow conservation:
inflow=outflow for each vertex $\neq s, t$

Size of flow $v(f)$:
sum of outflows from s



Each flow f can be partitioned into $v(f)$ $s - t$ paths

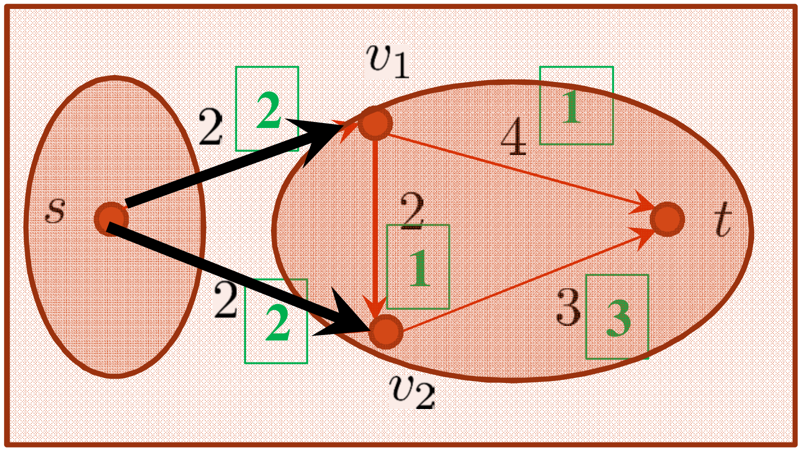
Maximum flow: flow with maximum size

Flow f is maximum if and only if it admits no f-augmenting path.

Forward arcs: $f(e) < w(e)$
Backward arcs: $f(e) > 0$

MINITUTORIAL ON NETWORK FLOWS

Network is a pair $N = (G, w)$ where $G = (V, E)$ is a directed graph with a source s and sink t and $w : E \to \mathbb{N}$ are capacities of arcs.



Flow in N:
function $f : E \to \mathbb{R}^+$ that fulfils:

capacity constraints:
$f(e) \leq w(e)$ for each arc e

flow conservation:
inflow=outflow for each vertex $\neq s, t$

Size of flow $v(f)$:
sum of outflows from s

Each flow f can be partitioned into $v(f)$ $s - t$ paths

Maximum flow: flow with maximum size

Cut in network: partition of vertices into X, Y so that $s \in X$ and $t \in Y$

Capacity of a cut (X, Y): $w(\delta^{out}(X) = \sum\{w(e); e \text{ goes from } X \text{ to } Y\}$

For each flow f and each cut (X, Y): $v(f) \leq w(\delta^{out}(X))$

Theorem (Maxflow-mincut). A flow f is maximum if and only if its size is equal to the capacity of some cut.

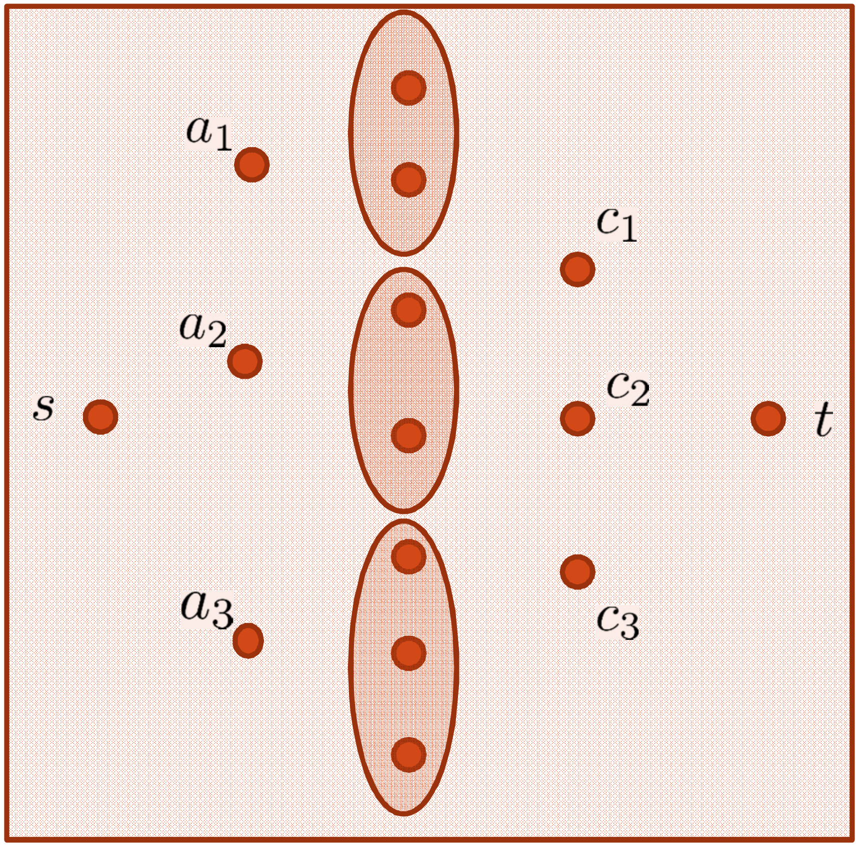
ALGORITHM FOR MANY-TO-MANY MATCHINGS WITH INDIFFERENCES

agent	quota	preference list	object	quota
a_1	2	$(c_1, c_2), c_3$	c_1	3
a_2	3	$c_2, (c_1, c_3)$	c_2	1
a_3	2	c_3, c_2, c_1	c_3	1

Lexicographic preferences

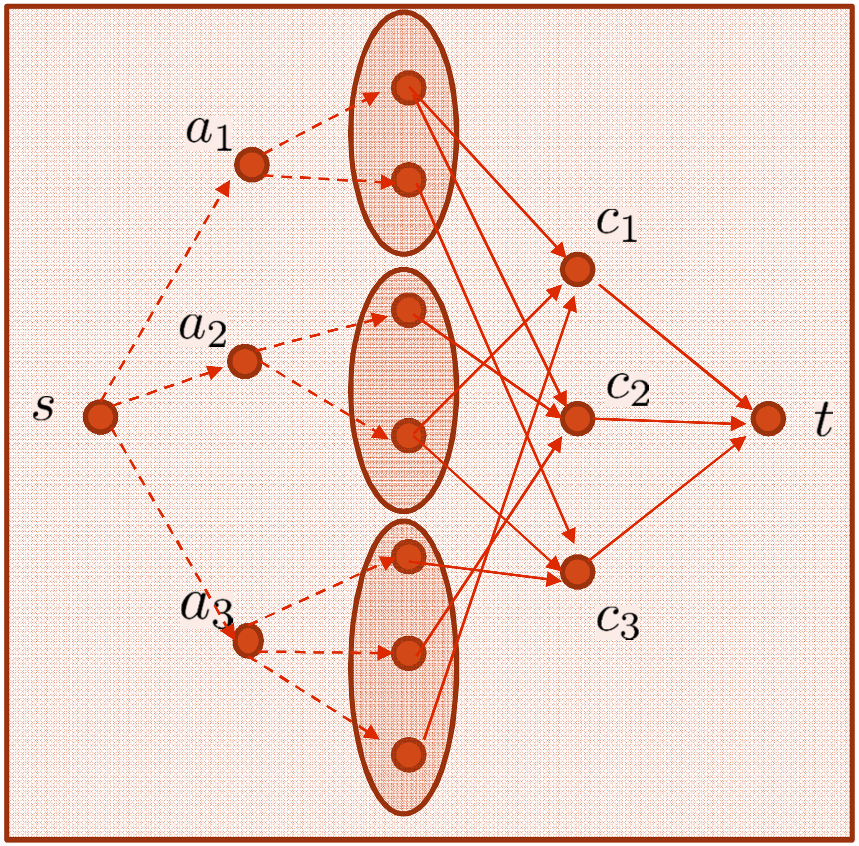
The algorithm uses network $N(I)$.

Vertices: s, t, agents,ties, objects



ALGORITHM FOR MANY-TO-MANY MATCHINGS WITH INDIFFERENCES

agent	quota	preference list	object	quota
a_1	2	$(c_1, c_2), c_3$	c_1	3
a_2	3	$c_2, (c_1, c_3)$	c_2	1
a_3	2	c_3, c_2, c_1	c_3	1



Lexicographic preferences

The algorithm uses network $N(I)$.

Vertices: s, t, agents,ties, objects

Arcs:

(c, t): capacity is $q(c)$

(tie,object): capacity is 1

(s, agent) and (agent,tie):
capacity increases during algorithm

ALGORITHM FOR MANY-TO-MANY MATCHINGS WITH INDIFFERENCES

agent	quota	preference list	object	quota
a_1	2	$(c_1, c_2), c_3$	c_1	3
a_2	3	$c_2, (c_1, c_3)$	c_2	1
a_3	2	c_3, c_2, c_1	c_3	1

Lexicographic preferences

The algorithm uses network $N(I)$.

Vertices: s, t, agents,ties, objects

Arcs:

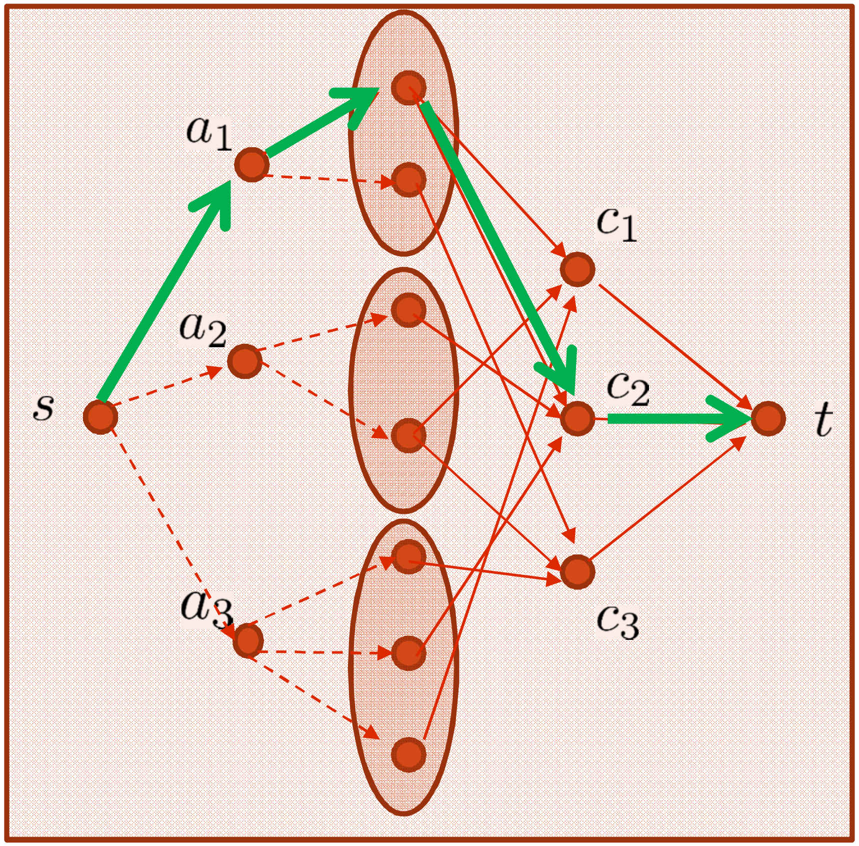
(c, t): capacity is $q(c)$

(tie,object): capacity is 1

(s, agent) and (agent,tie): capacity increases during algorithm

Policy $\sigma = a_1, a_2, a_3, a_2, a_2, a_1, a_3$

The algorithm works in stages.



Stage i: applicant a^i increases her capacity by 1

increases capacity of tie C_j^a

a^i can get an object from tie C_j^a iff network in $N^{i,t}$ admits augmenting path.

ALGORITHM FOR MANY-TO-MANY MATCHINGS WITH INDIFFERENCES

agent	quota	preference list	object	quota
a_1	2	$(c_1, c_2), c_3$	c_1	3
a_2	3	$c_2, (c_1, c_3)$	c_2	1
a_3	2	c_3, c_2, c_1	c_3	1

Lexicographic preferences

The algorithm uses network $N(I)$.

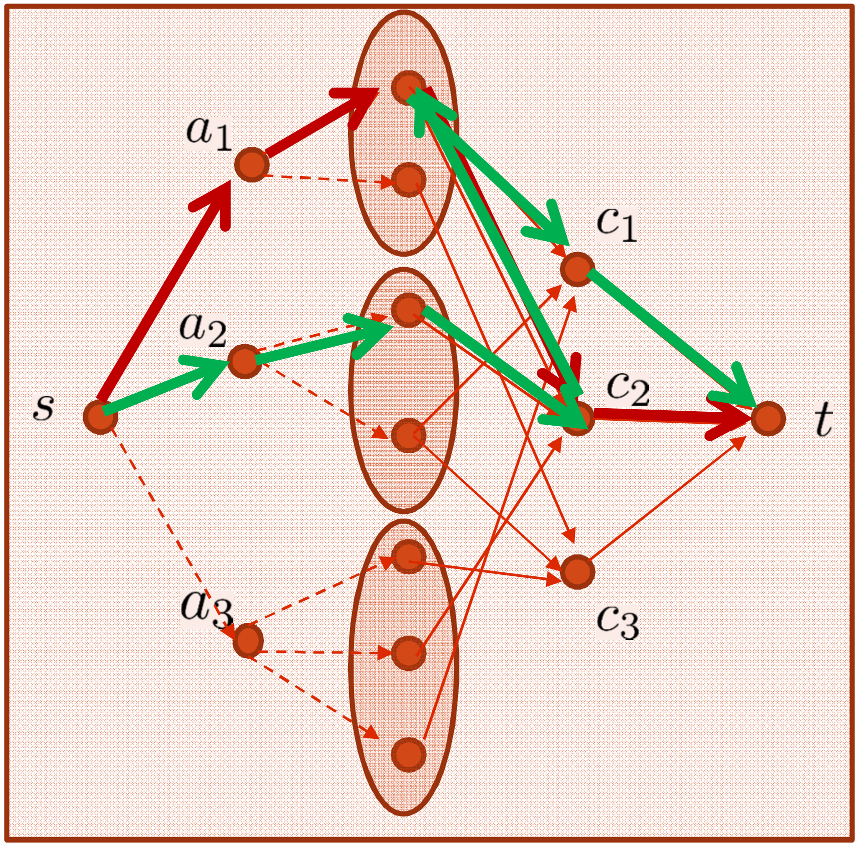
Vertices: s, t, agents,ties, objects

Arcs:

(c, t): capacity is $q(c)$

(tie,object): capacity is 1

(s, agent) and (agent,tie):
capacity increases during algorithm



Stage 2: applicant $a^2 = a_2$ increases her capacity by 1
increases capacity of her first tie

LOWER QUOTAS OF COURSES

Applicant a has capacity $q(a)$;
course c has lower quota $\ell(c)$ and upper quota $u(c)$.

applicant	capacity	preference list	course	lower quota	upper quota
a_1	3	c_1, c_2, c_3	c_1	3	3
a_2	2	c_3, c_1, c_4	c_2	1	1
a_3	1	c_2, c_3	c_3	2	3
a_4	1	c_4, c_1	c_4	2	2

If a course does not achieve its lower quota then it stays closed.

Matchings with project closures:
Monte & Tumenassan 2013,
Kamiyama 2013,
C. & Fleiner 2016

An assignment M is a *matching* if:
(i) $M(a) \subseteq P(a)$, $|M(a)| \leq q(a)$ for each $a \in A$;
(ii) $\ell(c) \leq |M(c)| \leq u(c)$ or $M(c) = \emptyset$ for each $c \in C$.

An assignment M is called a *partial* matching if it fulfils (i) and
(ii') $|M(c)| \leq u(c)$ for each $c \in C$.

A partial matching M has a set $\mathscr{D}(M)$ of demanding courses: $0 < |M(c)| < \ell(c)$

Residual demand of a partial matching M: $RD(M) = \sum_{c \in \mathscr{D}(M)} (\ell(c) - |M(c)|)$.

A partial matching M is a matching iff $RD(M) = 0$.

LOWER QUOTAS: ALGORITHM GSDPC

Applicants' clones are ordered into a picking sequence $\sigma = a^1, a^2, \ldots, a^Q$.

Algorithm GSDPC works in *rounds*. Round k starts with a partial matching M_{k-1}.

applicant	capacity	preference list	course	lower quota	upper quota
a_1	3	c_1, c_2, c_3	c_1	3	3
a_2	2	c_3, c_1, c_4	c_2	1	1
a_3	1	c_2, c_3	c_3	2	3
a_4	1	c_4, c_1	c_4	2	2

Round k: assign applicant a^k the best possible course c on conditions that:

- no course will exceed its upper quota
- all courses from $\mathscr{D}(M_{k-1} \cup (a^k, c)\}$ can still fulfil their lower quotas.

To check these conditions we use network flows.

Network $N(M)$:

- applicant vertices,course vertices, s, t
- capacity of (sa)=residual capacity of aplicant a
- arc $(a_j c_k)$ if $c_k \in P(a_j)$ and a_j has not yet considered c_k
- capacity of arc $(c_k t)$ is $\ell(c_k) - |M(c_k)|$ if $c_k \in \mathscr{D}(M)$

Lemma. There exists a matching μ such that $M_k = M_{k-1} \cup \{(a, c)\} \subseteq \mu$ if and only if $N(M_k)$ admits a flow f_k of value $RD(M_k)$.

LOWER QUOTAS: ALGORITHM GSDPC

Picking sequence $\sigma = a_1, a_4, a_2, a_3, a_2, a_1, a_1$.

applicant	capacity	preference list	course	lower quota	upper quota	$RD(M)$
a_1	3	~~c_1~~, c_2, c_3	c_1	3	3	~~0~~ 2
a_2	2	c_3, c_1, c_4	c_2	1	1	0
a_3	1	c_2, c_3	c_3	2	3	0
a_4	1	c_4, c_1	c_4	2	2	0

Round 1: $M_0 = \emptyset$

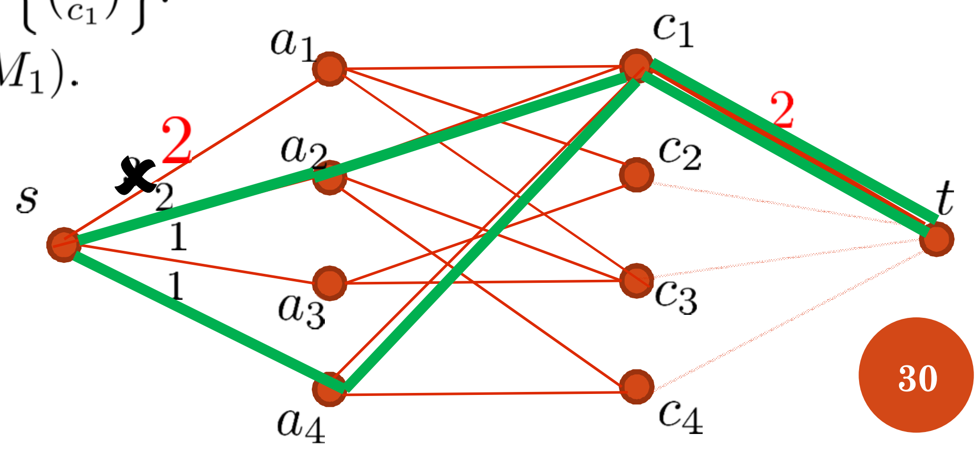
Applicant a_1 is treated, she considers c_1.

Provisional partial matching $M_1 = \left\{ \binom{a_1}{c_1} \right\}$.

Modify the network $N(M_0) \to N(M_1)$.

Flow of value 2 is needed.

$M_1 = \left\{ \binom{a_1}{c_1} \right\}$ becomes fixed.



LOWER QUOTAS: ALGORITHM GSDPC

Picking sequence $\sigma =$ ~~a_1~~, ~~a_4~~, a_2, a_3, a_2, a_1, a_1.

applicant	capacity	preference list	course	lower quota	upper quota
a_1	3	~~c_1~~, c_2, c_3	c_1	3	3
a_2	2	c_3, c_1, c_4	c_2	1	1
a_3	1	c_2, c_3	c_3	2	3
a_4	1	~~c_4~~, c_1	c_4	2	2

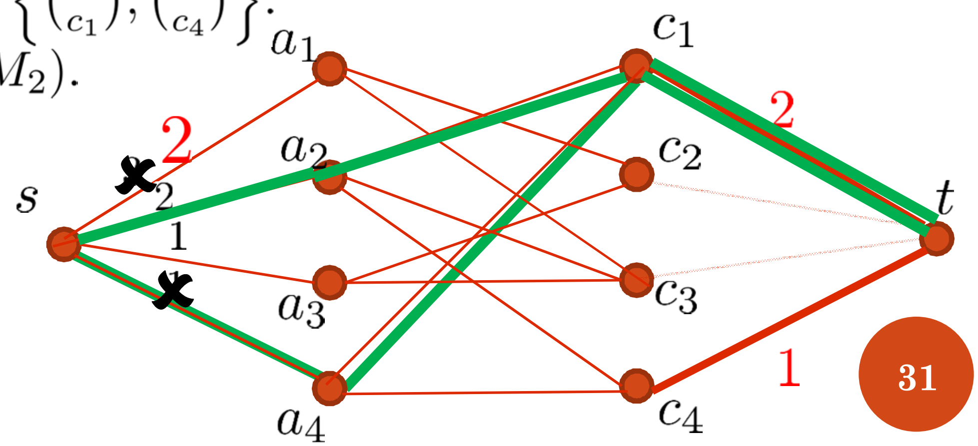
$RD(M)$
~~0~~ 2
0
0
~~0~~ 1

Round 2: $M_1 = \left\{ \binom{a_1}{c_1} \right\}$.

Applicant a_4 is treated, she considers c_4.

Provisional partial matching $M_2 = \left\{ \binom{a_1}{c_1}, \binom{a_4}{c_4} \right\}$.

Modify the network $N(M_1) \to N(M_2)$.



LOWER QUOTAS: ALGORITHM GSDPC

Picking sequence $\sigma = a_1, a_4, a_2, a_3, a_2, a_1, a_1$.

applicant	capacity	preference list	course	lower quota	upper quota
a_1	3	c_1, c_2, c_3	c_1	3	3
a_2	2	c_3, c_1, c_4	c_2	1	1
a_3	1	c_2, c_3	c_3	2	3
a_4	1	c_4, c_1	c_4	2	2

$RD(M)$
0 2
0
0
0 1

Round 2: $M_1 = \left\{ \binom{a_1}{c_1} \right\}$.

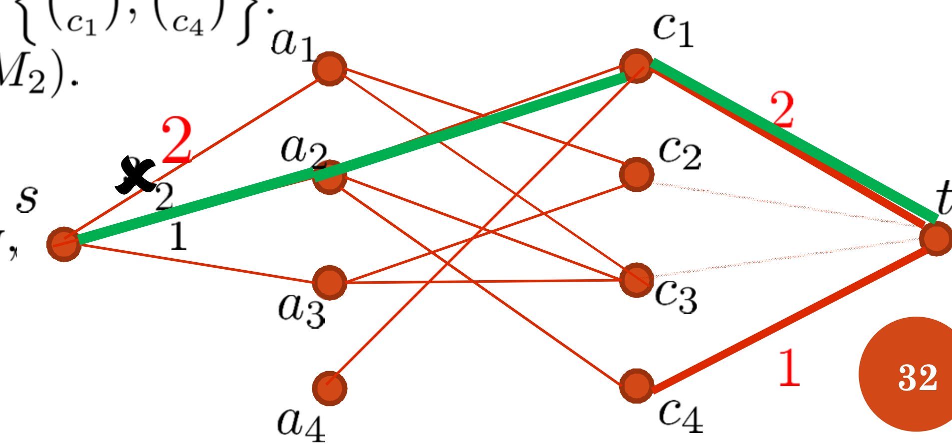
Applicant a_4 is treated, she considers c_4.

Provisional partial matching $M_2 = \left\{ \binom{a_1}{c_1}, \binom{a_4}{c_4} \right\}$.

Modify the network $N(M_1) \to N(M_2)$.

Flow of value 3 is needed.

$N(M_2)$ does not admit such a flow, therefore return to M_1.



LOWER QUOTAS: ALGORITHM GSDPC

Picking sequence $\sigma =$ ✗, ✗, a_2, a_3, a_2, a_1, a_1.

applicant	capacity	preference list	course	lower quota	upper quota
a_1	3	✗, c_2, c_3	c_1	3	3
a_2	2	c_3, c_1, c_4	c_2	1	1
a_3	1	c_2, c_3	c_3	2	3
a_4	1	✗, ✗	c_4	2	2

$RD(M)$
✗ 21
0
0
0

Round 2: $M_1 = \left\{ \binom{a_1}{c_1} \right\}$.

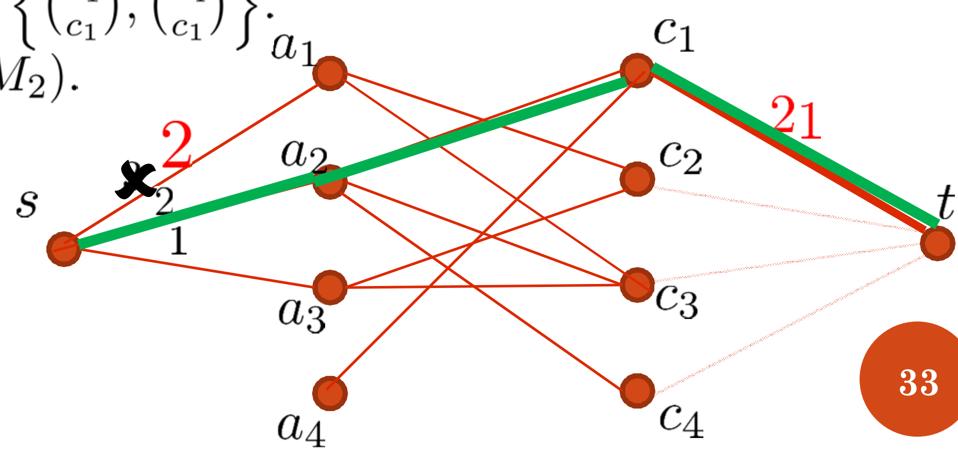
Applicant a_4 is still treated, she considers c_1.

Provisional partial matching $M_2 = \left\{ \binom{a_1}{c_1}, \binom{a_4}{c_1} \right\}$.

Modify the network $N(M_1) \to N(M_2)$.

Flow of value 1 is needed.

$M_2 = \left\{ \binom{a_1}{c_1}, \binom{a_4}{c_1} \right\}$ becomes fixed.



PROPERTIES OF GSDPC

Theorem. GSDPT outputs a Pareto optimal matching.

Proof. By Maxflow - Mincut theorem, in each round k:

$0 \leq$ residual demand of $M_k \leq value(f_k) \leq w(\delta^{out}\{s\}) =$ residual capacity

Last round: residual capacity $0 \Longrightarrow RD(M_r) = 0 \Longrightarrow M_r$ is a matching.
Pareto optimality: induction argument

Computational complexity:

L (applicant,course) pairs in preference lists; each explored at most once

Do not start from zero flow, at most $\ell(c)$ searches in network when exploring c

In total: $O(L^2 max_{c \in C} \ell(c))$

Theorem. CALQ-DOMINANCE is NP-complete even in the case when $q(a) = 1$ for each $a \in A$ and no lower quota of a course exceeds 3.

Theorem. Finding a POM with maximum cardinality in an instance of CALQ is NP-hard, even if no lower quota exceeds 4 and capacities of applicant are 1.

Theorem. Finding a POM in an instance with indifferences is NP-hard, even if each applicant is indifferent between all her acceptable courses.

STRATEGIC ISSUES

Assumption: applicants know the picking sequence and all preferences.

Two types of manipulations:
reordering: changing the order of the entries in the preference list;
dropping: declaring some courses in the preference lists unacceptable

GSDPC is not immune against reodering manipulations

applicant	capacity	preference list	course	lower quota	upper quota
a_1	2	c_1 c_2	c_1	1	2
a_1	1	c_1 c_2	c_2	2	2

Assume picking sequence a_1, a_2, a_1.

Both applicants act truthfully: output $M_1(a_1) = M_1(a_2) = \{c_1\}$.

If a_1 reports c_2, c_1: output $M_2(a_1) = \{c_1, c_2\}; M_2(a_2) = \{c_2\}$.

Theorem. GSDPC with a *contiguous* picking sequence is strategy-proof against reordering manipulations.

STRATEGIC ISSUES

Theorem. There is no Pareto optimal mechanism for CALQ that is strategy-proof against dropping manipulations.

applicant	capacity	preference list	course	lower quota	upper quota
a_1	1	c_1, c_2	c_1	2	2
a_1	1	c_2, c_1	c_2	2	2

Two POMs: $M_1(c_1) = \{a_1, a_2\}$. $M_2(c_2) = \{a_1, a_2\}$.

If a mechanism outputs M_1, a_2 has incentives to drop c_1.

applicant	capacity	preference list	course	lower quota	upper quota
a_1	1	c_1, c_2	c_1	2	2
a_1	1	c_2	c_2	2	2

If a mechanism outputs M_2, a_1 has incentives to drop c_2.

applicant	capacity	preference list	course	lower quota	upper quota
a_1	1	c_1	c_1	2	2
a_1	1	c_2, c_1	c_2	2	2

PARETO OPTIMAL MATCHINGS WITH PREREQUISITES CONSTRAINTS

Prerequisites: a student is allowed to subscribe to a course c only if she subscribes to a set C' of other course(s).

Example:
Optimal Control Theory requires Differential Equations and Linear Algebra
Differential Equations require a Calculus course

For each applicant $a \in A$: a partial order $\rightarrow_a$ on C

Meaning: if $c \in M(a)$ and $c \rightarrow_{a_i} c'$ then $c' \in M(a)$

For lexicographic preferences:
a POM can be found by a modified sequential mechanism

PARETO OPTIMAL MATCHINGS WITH PREREQUISITES CONSTRAINTS

Prerequisites: a student is allowed to subscribe to a course c only if she subscribes to a set C' of other course(s).

Example:
Optimal Control Theory requires Differential Equations and Linear Algebra
Differential Equations require a Calculus course

For each applicant $a \in A$: a partial order $\rightarrow_a$ on C

Meaning: if $c \in M(a)$ and $c \rightarrow_{a_i} c'$ then $c' \in M(a)$

For lexicographic preferences:
a POM can be found by a modified sequential mechanism

Algorithm SM-CAPR:
- always finds a Pareto optimal matching, given any policy
- runs in polynomial time
- may not produce all Pareto optimal matchings
- is not strategy-proof (implied also by (Hosseini and Larson, 2015)

Hard problems:
- Deciding whether a matching is Pareto optimal is co-NP-complete
- Finding a maximum cardinality Pareto optimal matching is NP-hard

COMPULSORY PREREQUISITES

$\sigma = \langle a_1, a_2, a_1, a_2, a_1, a_2, \ldots \rangle$

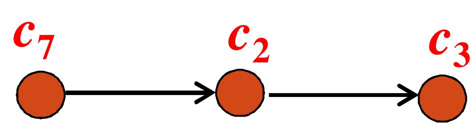
$a_1 : c_1\ c_2\ c_3\ c_4\ c_5\ c_6\ c_7\ c_8$

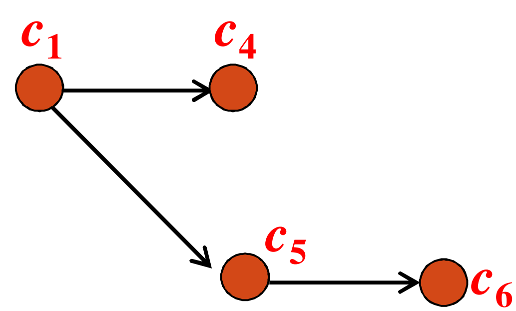
$a_2 : c_1\ c_2\ c_3\ c_4\ c_5\ c_6\ c_7\ c_8$

$q(a_1) =$ ~~6~~ ~~2~~ 1

$q(a_2) =$ ~~4~~ ~~2~~ 0

$q(c_1) =$ ~~2~~ 1
$q(c_2) =$ ~~1~~ 0
$q(c_3) =$ ~~1~~ 0
$q(c_4) =$ ~~1~~ 0
$q(c_5) =$ ~~2~~ ~~1~~ 0
$q(c_6) =$ ~~2~~ ~~1~~ 0
$q(c_7) = 1$
$q(c_8) =$ ~~1~~ 0





PARETO OPTIMAL MATCHINGS WITH ALTERNATING PREREQUISITES

Prerequisites: a student is allowed to subscribe to a course c only if she subscribes to at least one course from a given set C'

Example:
Mathematical modeling requires some course on mathematical sofware (MATHEMATICA,MATLAB, MAPLE ...)

For each applicant $a \in A$ there is a mapping $\mapsto_a : C \to 2^C$

Meaning:
if $c \in M(a)$ and $c \mapsto_a \{c_{i_1}, c_{i_2}, \ldots, c_{i_k}\}$ then $c_{i_j} \in M(a)$ for some $j = 1, \ldots, k$

Bad news: finding a Pareto optimal matching is NP-hard under either additive or lexicographic preferences

PARETO OPTIMAL MATCHINGS WITH COPREREQUISITES

For each applicant $a \in A$ there is an equivalence relation $\leftrightarrow_a$ on C

Meaning: $M(a)$ contains either all courses from an equivalence class or none

Algorithm for lexicographic preferences:

1. replace each course $d \in C$ by its equivalence class D:
   - size of the 'supercourse' is the number of courses in the equivalence class
   - position of the 'supercourse' in the preference list is the position of the best course of the equivalence class
2. Run the sequential mechanism (take care of sizes)

> Theorem. MAX POM CACR is NP-hard and not approximable within a factor of $N^{1-\varepsilon}$, for any $\varepsilon > 0$, unless P=NP, where N is the total capacity of the applicants.

EMPIRICAL STUDY

- Assignment of students to bachelor projects
- 53 students, 64 offered topics
- Distributed maket, we had results of real outcome
- We elicitated students´ preferences
- What are the preferences of teachers?
- Serial dictatorship: policy decreasing in students´ grades
- 7 students improved compared to the real outcome

Thank you for your attention!