Assignments, Arborescences, and Popularity

Kavitha Telikepalli

(TIFR, Mumbai)

NESTiD Seminar — Jan 16, 2026

Joint work with:

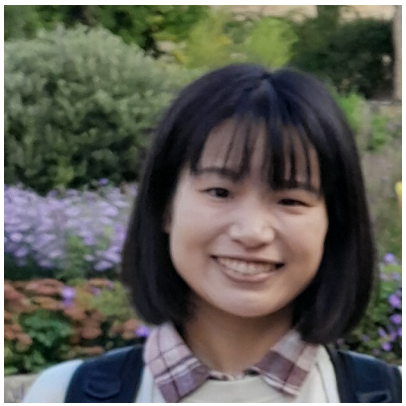
Tamás Király

Kazuhisa Makino

Jannik Matuschke

Ildikó Schlotter

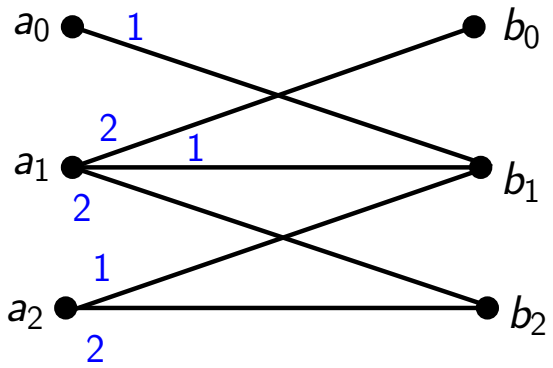
Ulrike Schmidt-Kraepelin



Yu Yokoi

Our model

We are given a bipartite graph G with one-sided preferences.

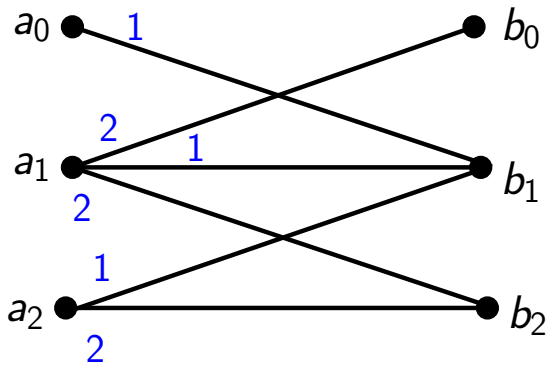


Vertices on the left are agents and on the right are houses.

- Every agent has a ranking over houses it is interested in.

Our model

We are given a bipartite graph G with one-sided preferences.



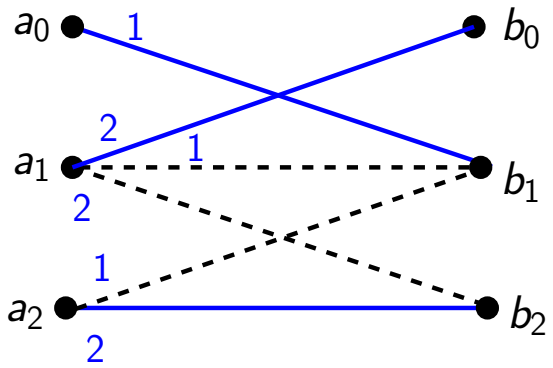
Vertices on the left are agents and on the right are houses.

- Every agent has a ranking over houses it is interested in.

This is called a *house allocation* instance.

Our model

We are given a bipartite graph G with one-sided preferences.



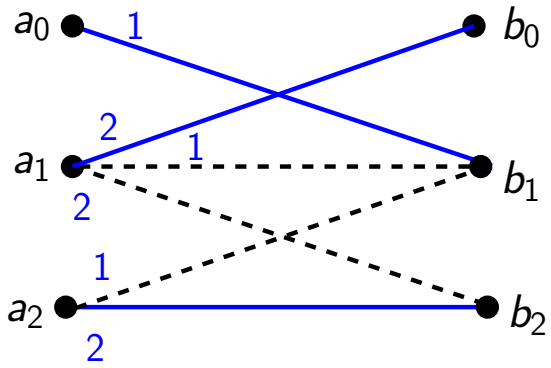
Vertices on the left are agents and on the right are houses.

- Every agent has a ranking over houses it is interested in.

This is called a *house allocation* instance.

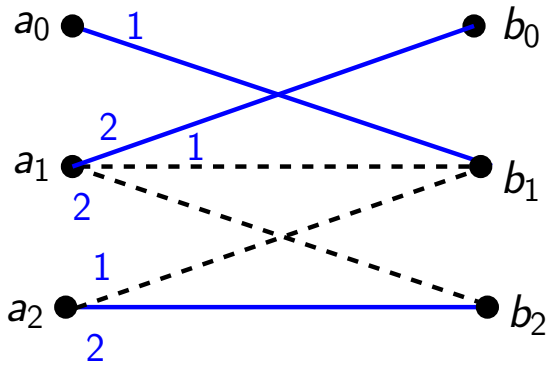
Optimal matchings

Is this matching M good?



Optimal matchings

Is this matching M good?

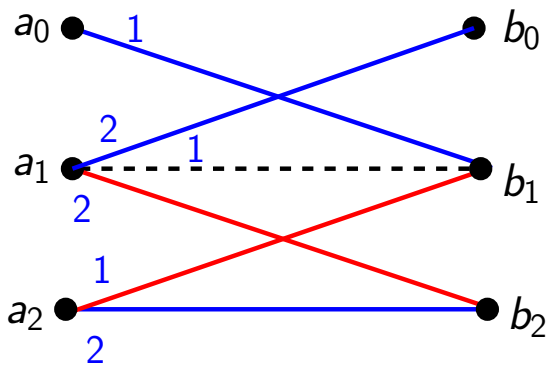


M can be compared with any matching N:

- by holding a head-to-head election between them;
- each agent a_i is a voter.

Optimal matchings

Is this matching M good?



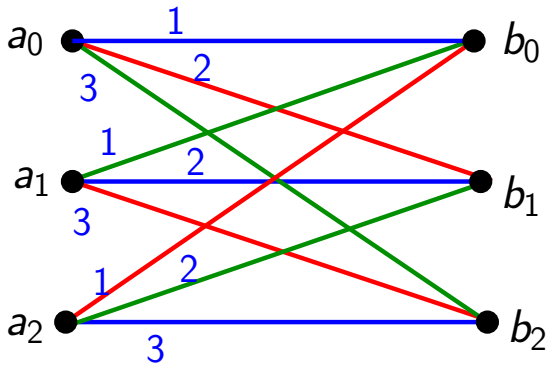
Let us compare the blue matching M with the red matching N:

- a_0 votes for M and a_2 votes for N.
- a_1 abstains from voting.

The "more popular than" relation

Consider the following instance with 3 agents and 3 houses.

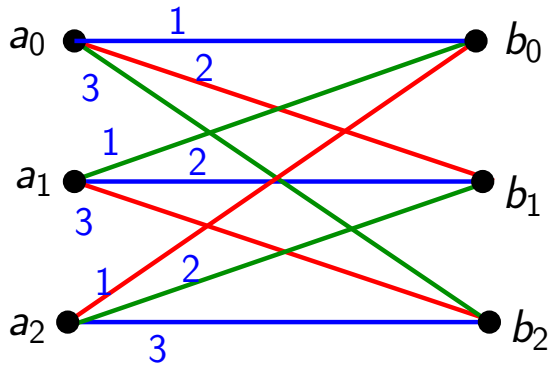
- All agents have the same ranking: $b_0 \succ b_1 \succ b_2$.



The "more popular than" relation

Consider the following instance with 3 agents and 3 houses.

- All agents have the same ranking: $b_0 \succ b_1 \succ b_2$.



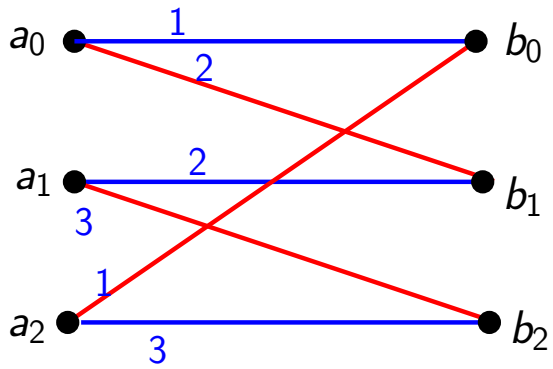
Let us hold elections between some pairs of matchings here.

- Say, between the blue matching and the red matching.

The "more popular than" relation

Consider the following instance with 3 agents and 3 houses.

- All agents have the same ranking: $b_0 \succ b_1 \succ b_2$.



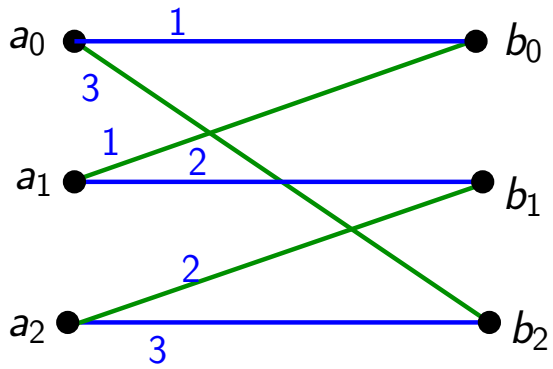
The blue matching is more popular than the red matching.

- In the blue vs red election: blue gets 2 votes (a_0 and a_1) and red gets only 1 (a_2).

The "more popular than" relation

Consider the following instance with 3 agents and 3 houses.

- All agents have the same ranking: $b_0 \succ b_1 \succ b_2$.



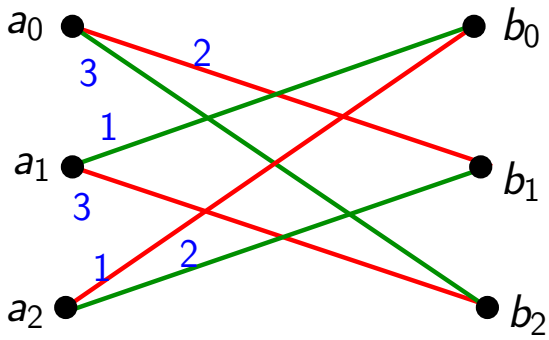
The green matching is more popular than the blue matching.

- In the green vs blue election: green gets 2 votes (a_1 and a_2) and blue gets 1 (a_0).

The "more popular than" relation

Consider the following instance with 3 agents and 3 houses.

- All agents have the same ranking: $b_0 \succ b_1 \succ b_2$.



The red matching is more popular than the green matching.

- In the red vs green election: red gets 2 votes (a_0 and a_2) and green gets 1 (a_1).

Popular matchings

So we have blue $\succ$ red $\succ$ green $\succ$ blue.

- Thus the "more popular than" relation is not transitive.

Popular matchings

So we have blue $\succ$ red $\succ$ green $\succ$ blue.

- Thus the "more popular than" relation is not transitive.

Call a matching M popular if there is *no* matching more popular than M.

- So M never loses any head-to-head election.
- Popular matchings are *weak Condorcet winners*.

Popular matchings

So we have blue $\succ$ red $\succ$ green $\succ$ blue.

- Thus the "more popular than" relation is not transitive.

Call a matching M popular if there is *no* matching more popular than M.

- So M never loses any head-to-head election.
- Popular matchings are *weak Condorcet winners*.

PROBLEM. Given an instance $G = (A \cup B, E)$, does G admit a popular matching?

Popular matchings

So we have blue $\succ$ red $\succ$ green $\succ$ blue.

- Thus the "more popular than" relation is not transitive.

Call a matching M popular if there is *no* matching more popular than M.

- So M never loses any head-to-head election.
- Popular matchings are *weak Condorcet winners*.

PROBLEM. Given an instance $G = (A \cup B, E)$, does G admit a popular matching?

- A combinatorial characterization of popular matchings is known.

  [Joint work with Abraham, Irving, and Mehlhorn, 2007]

- This leads to an efficient algorithm for this problem.

Characterizing popular matchings

For every $a \in A$, let us add a dummy house $d(a)$ as a's *worst* house.

- Henceforth, only A-perfect matchings are interesting.

Characterizing popular matchings

For every $a \in A$, let us add a dummy house $d(a)$ as a's *worst* house.

- Henceforth, only A-perfect matchings are interesting.

Assume strict rankings. For any $a \in A$:

- let $f(a) = \{a$'s top choice house$\}$;
- let $s(a) = \{a$'s most preferred house that is nobody's top choice$\}$.

Characterizing popular matchings

For every $a \in A$, let us add a dummy house $d(a)$ as a's *worst* house.

- Henceforth, only A-perfect matchings are interesting.

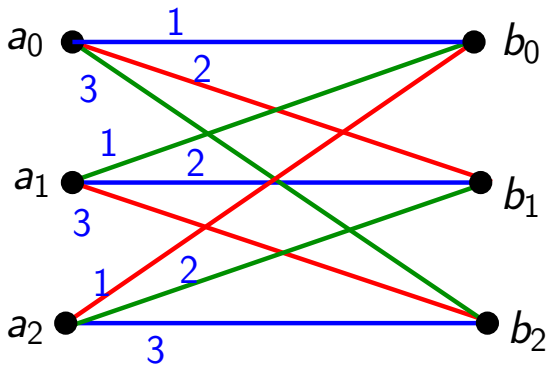
Assume strict rankings. For any $a \in A$:

- let $f(a) = \{a$'s top choice house$\}$;
- let $s(a) = \{a$'s most preferred house that is nobody's top choice$\}$.

Claim. For any $a \in A$ and any popular matching M:

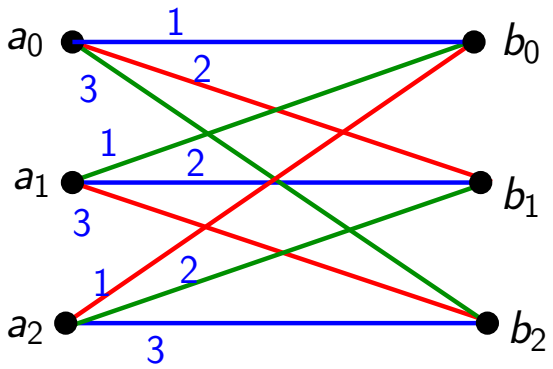
- $M(a) \in f(a) \cup s(a)$.

Structure of popular matchings for strict rankings



- Here $f(a_0) = f(a_1) = f(a_2) = \{b_0\}$.
- And $s(a_0) = s(a_1) = s(a_2) = \{b_1\}$.

Structure of popular matchings for strict rankings



- Here $f(a_0) = f(a_1) = f(a_2) = \{b_0\}$.
- And $s(a_0) = s(a_1) = s(a_2) = \{b_1\}$.

M is popular $\Rightarrow$ $M(a) \in \{b_0, b_1\}$ for all $a \in A$.

- There is no such A-perfect matching.
- Hence there is no popular matching.

For weak rankings

Let $E_0 = \{\text{top edges in } G\}$ i.e., $ab \in E_0 \iff b$ is a top choice house for a.

- The subgraph $G_0 = (A \cup B, E_0)$ will be useful.

For weak rankings

Let $E_0 = \{\text{top edges in } G\}$ i.e., $ab \in E_0 \iff b$ is a top choice house for a.

- The subgraph $G_0 = (A \cup B, E_0)$ will be useful.

For each $a \in A$:

- let $f(a) = \{a\text{'s top choice houses}\}$;
- let $s(a) = \{a\text{'s most preferred houses among } \underbrace{\text{non-critical vertices}}_{\downarrow \text{ those left unmatched in some maximum matching}} \text{ in } G_0\}$.

For weak rankings

Let $E_0 = \{\text{top edges in } G\}$ i.e., $ab \in E_0 \iff b$ is a top choice house for a.

- The subgraph $G_0 = (A \cup B, E_0)$ will be useful.

For each $a \in A$:

- let $f(a) = \{a\text{'s top choice houses}\}$;
- let $s(a) = \{a\text{'s most preferred houses among } \underbrace{\text{non-critical vertices}}_{\downarrow \text{ those left unmatched in some maximum matching}} \text{ in } G_0\}$.

Claim. M is popular if and only if:

- $M(a) \in f(a) \cup s(a)$ for all $a \in A$.
- $M \cap E_0$ is a maximum matching in G_0.

The popular matching algorithm

G has a popular matching $\Leftrightarrow$ $G' = (A \cup B, E')$ has an A-perfect matching.

- $E' = \{ab : a \in A$ and $b \in f(a) \cup s(a)\}$.

The popular matching algorithm

G has a popular matching $\Leftrightarrow$ $G' = (A \cup B, E')$ has an A-perfect matching.

- $E' = \{ab : a \in A$ and $b \in f(a) \cup s(a)\}$.

Our algorithm checks if G' admits an A-perfect matching.

- If not, it says "no popular matching";
- else it returns an A-perfect matching M in G' that maximizes $|M \cap E_0|$.

The popular matching algorithm

G has a popular matching $\Leftrightarrow$ $G' = (A \cup B, E')$ has an A-perfect matching.

- $E' = \{ab : a \in A \text{ and } b \in f(a) \cup s(a)\}$.

Our algorithm checks if G' admits an A-perfect matching.

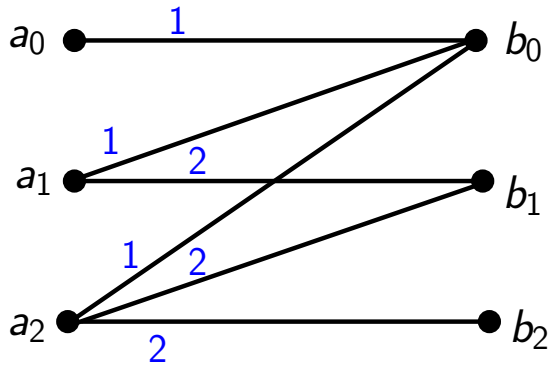
- If not, it says "no popular matching";
- else it returns an A-perfect matching M in G' that maximizes $|M \cap E_0|$.

Algorithm says "no popular matching" $\Rightarrow$ there is no A-perfect matching M in G such that $M(a) \in f(a) \cup s(a)$ for all $a \in A$.

- So G indeed has no popular matching.

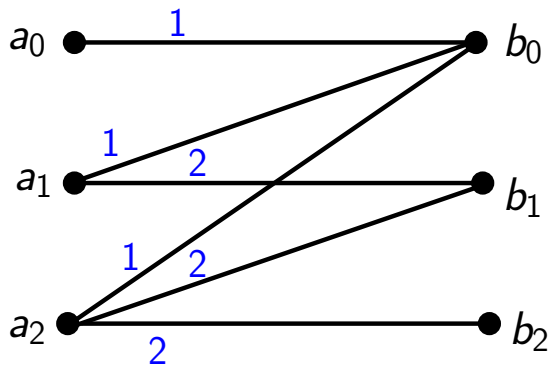
When cardinality is more important than popularity

Suppose the most important attribute of a matching is its cardinality.



When cardinality is more important than popularity

Suppose the most important attribute of a matching is its cardinality.



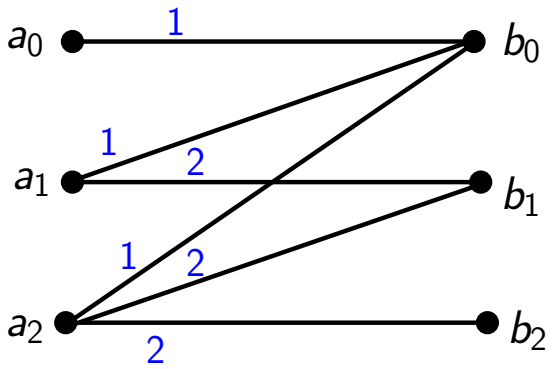
The cardinality of the matching is important in many applications:

- campus housing to students;
- training positions to trainees;
- staff to hospitals in emergencies such as a pandemic;
- allocation problems for humanitarian organizations.

When cardinality is more important than popularity

We seek a maximum matching in these applications.

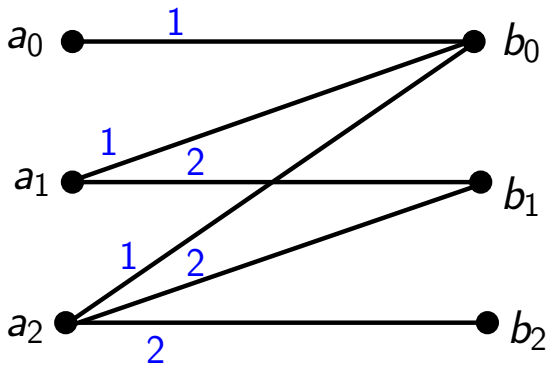
- We can assume wlog that G has a perfect matching, i.e., an assignment.



When cardinality is more important than popularity

We seek a maximum matching in these applications.

- We can assume wlog that G has a perfect matching, i.e., an assignment.



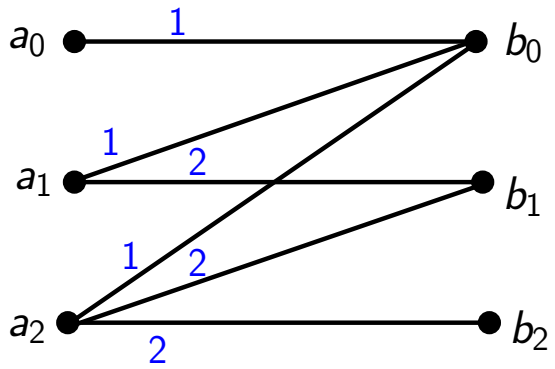
Popular assignment M:

- No assignment defeats M in their head-to-head election;
- smaller matchings may defeat M.

When cardinality is more important than popularity

We seek a maximum matching in these applications.

- We can assume wlog that G has a perfect matching, i.e., an assignment.



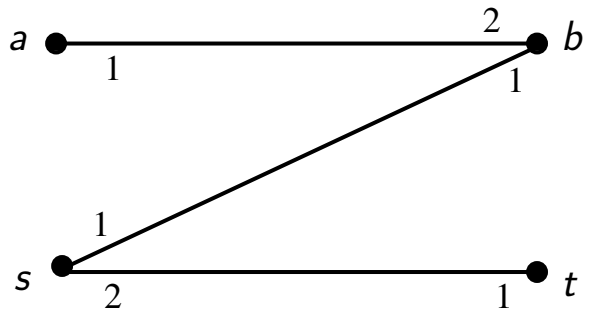
Popular assignment M:

- No assignment defeats M in their head-to-head election;
- smaller matchings may defeat M.

Popular assignment problem. Does $G = (A \cup B, E)$ admit a popular assignment?

A detour

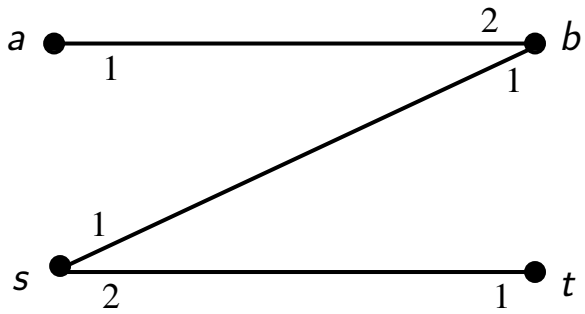
Popular matchings were first studied in the *stable marriage* setting.



- Here preferences are two-sided and strict.

A detour

Popular matchings were first studied in the *stable marriage* setting.



- Here preferences are two-sided and strict.

To compare a pair of matchings M and N:

- hold a head-to-head election between them;
- now *all* vertices are voters.

A matching is popular if it does not lose any election.

Stable matchings

A matching M is stable if there is no edge ab such that:

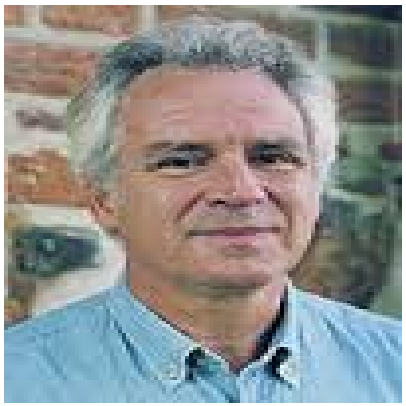
- a and b prefer each other to their respective assignments in M.

Stable matchings

A matching M is stable if there is no edge ab such that:

- a and b prefer each other to their respective assignments in M.

Every stable matching is popular [Gärdenfors 1975].



Comparing a stable matching S with any matching N:

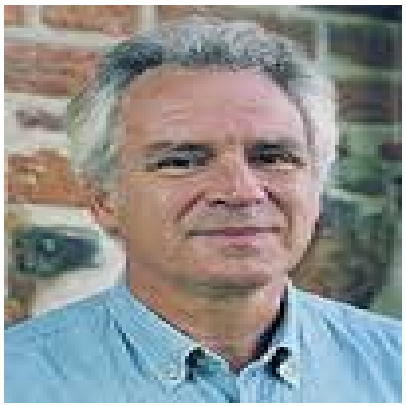
- u prefers N to $S \implies N(u)$ has to prefer S to N;
- so the number of votes for $N \leq$ the number of votes for S;
- hence S is popular.

Stable matchings

A matching M is stable if there is no edge ab such that:

- a and b prefer each other to their respective assignments in M.

Every stable matching is popular [Gärdenfors 1975].



Comparing a stable matching S with any matching N:

- u prefers N to $S \implies N(u)$ has to prefer S to N;
- so the number of votes for $N \leq$ the number of votes for S;
- hence S is popular.

So popular matchings always exist in this model.

The 2-sided strict preferences model

- Does G always admit a popular maximum matching?

  ↓

  a maximum matching undefeated by any maximum matching

The 2-sided strict preferences model

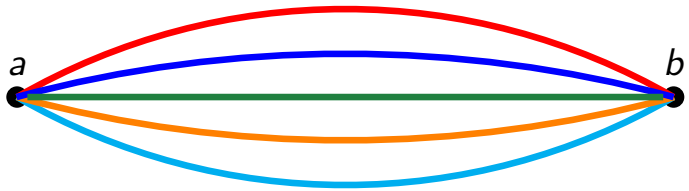
- Does G always admit a popular maximum matching?

  ↓

  a maximum matching undefeated by any maximum matching

A new stable marriage instance G^*:

- for every edge ab in G, there are n parallel copies $ab, ab, ab, \ldots, ab, ab$ in G^*.



The 2-sided strict preferences model

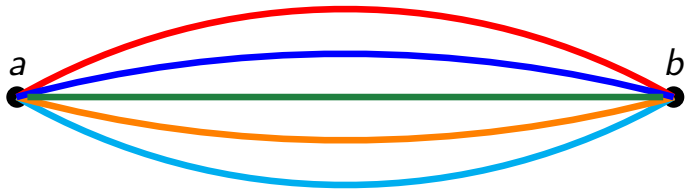
- Does G always admit a popular maximum matching?

  ↓

  a maximum matching undefeated by any maximum matching

A new stable marriage instance G^*:

- for every edge ab in G, there are n parallel copies $ab, ab, ab, \ldots, ab, ab$ in G^*.



  - For any $a \in A$:

    $$red \succ blue \succ green \succ \cdots \succ orange \succ cyan.$$

  - For any $b \in B$:

    $$cyan \succ orange \succ \cdots \succ green \succ blue \succ red.$$

The 2-sided strict preferences model

Gale-Shapley algorithm with colors

- Construct the colorful graph $G^* = (A \cup B, E^*)$.
- Run Gale-Shapley algorithm in G^* to compute M^*.
- Return the colorless matching M in G.

THEOREM [K 2014]. M is a popular maximum matching in G.

The 2-sided strict preferences model

Gale-Shapley algorithm with colors

- Construct the colorful graph $G^* = (A \cup B, E^*)$.
- Run Gale-Shapley algorithm in G^* to compute M^*.
- Return the colorless matching M in G.

THEOREM [K 2014]. M is a popular maximum matching in G.

- IDEA: Can we adapt this idea to the one-sided preferences model?

Back to one-sided preferences

Instead of coloring edges, let us color houses.

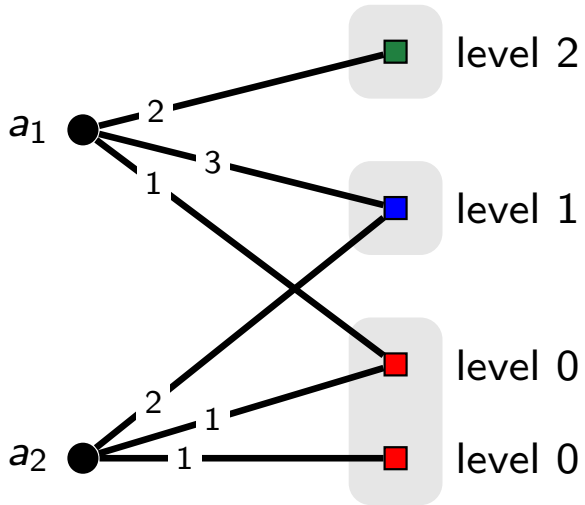
- Suppose we have a color/level function $\ell : B \to \{0, 1, 2, \ldots, n-2, n-1\}$.

Back to one-sided preferences

Instead of coloring edges, let us color houses.

- Suppose we have a color/level function $\ell : B \to \{0, 1, 2, \ldots, n-2, n-1\}$.

For each $a \in A$: let $\ell^*(a) = \underbrace{\max\{\ell(b) : b \in \text{Nbr}(a)\}}_{\text{highest level among } a\text{'s neighbors}}$.



- $\ell^*(a_1) = 2$ and $\ell^*(a_2) = 1$.

The subgraph G_ℓ

- Let $\ell : B \to \{0, 1, 2, \ldots, n-2, n-1\}$ be any function.
- We define a corresponding subgraph $G_\ell = (A \cup B, E_\ell)$ as follows.

The subgraph G_ℓ

- Let $\ell : B \to \{0, 1, 2, \ldots, n-2, n-1\}$ be any function.
- We define a corresponding subgraph $G_\ell = (A \cup B, E_\ell)$ as follows.

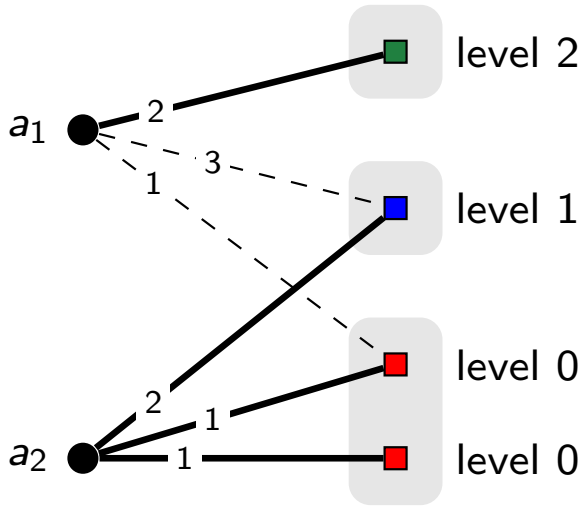
1. Each $a \in A$ keeps edges to its favorite level $\ell^*(a)$ neighbors.

The subgraph G_ℓ

- Let $\ell : B \to \{0, 1, 2, \ldots, n-2, n-1\}$ be any function.
- We define a corresponding subgraph $G_\ell = (A \cup B, E_\ell)$ as follows.
  1. Each $a \in A$ keeps edges to its favorite level $\ell^*(a)$ neighbors.
  2. Furthermore, a keeps edges to its favorite level $\ell^*(a)-1$ neighbors if a prefers them to all level $\ell^*(a)$ neighbors.

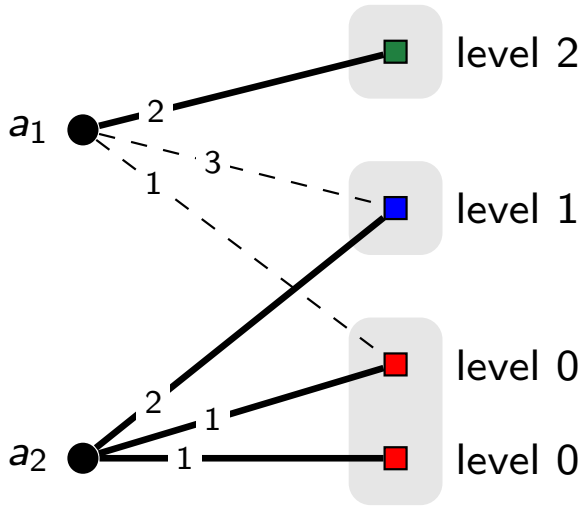
The subgraph G_ℓ

Only the bold edges are in G_ℓ because $\ell^*(a_1) = 2$ and $\ell^*(a_2) = 1$.



The subgraph G_ℓ

Only the bold edges are in G_ℓ because $\ell^*(a_1) = 2$ and $\ell^*(a_2) = 1$.



THEOREM. G has a popular assignment if and only if

- $\exists \ell : B \to \{0, 1, 2, \ldots, n-1\}$ s.t. G_ℓ admits a perfect matching M.

This matching M will be a popular assignment in G.

Popular assignment algorithm [KKMSS 2022]

PROBLEM. Find the right function $\ell : B \to \{0, 1, 2, \ldots, n-2, n-1\}$.

Popular assignment algorithm [KKMSS 2022]

PROBLEM. Find the right function $\ell : B \to \{0, 1, 2, \ldots, n-2, n-1\}$.

---

1. Initialize $\ell(b) = 0$ for every $b \in B$.
2. Compute a maximum matching M in the subgraph G_ℓ.
3. If M is perfect then return M.

Popular assignment algorithm [KKMSS 2022]

PROBLEM. Find the right function $\ell : B \to \{0, 1, 2, \ldots, n-2, n-1\}$.

---

1. Initialize $\ell(b) = 0$ for every $b \in B$.
2. Compute a maximum matching M in the subgraph G_ℓ.
3. If M is perfect then return M.
4. For every unmatched $b \in B$ do: $\ell(b) = \ell(b) + 1$.

Popular assignment algorithm [KKMSS 2022]

PROBLEM. Find the right function $\ell : B \to \{0, 1, 2, \ldots, n-2, n-1\}$.

---

1. Initialize $\ell(b) = 0$ for every $b \in B$.
2. Compute a maximum matching M in the subgraph G_ℓ.
3. If M is perfect then return M.
4. For every unmatched $b \in B$ do: $\ell(b) = \ell(b) + 1$.
5. If $\ell(b) = n$ for any $b \in B$ then return "no"; else go back to Step 2.

Popular assignment algorithm [KKMSS 2022]

PROBLEM. Find the right function $\ell : B \to \{0, 1, 2, \ldots, n-2, n-1\}$.

---

1. Initialize $\ell(b) = 0$ for every $b \in B$.
2. Compute a maximum matching M in the subgraph G_ℓ.
3. If M is perfect then return M.
4. For every unmatched $b \in B$ do: $\ell(b) = \ell(b) + 1$.
5. If $\ell(b) = n$ for any $b \in B$ then return "no"; else go back to Step 2.

The above algorithm works for partial order preferences and even *cyclic* preferences.

- Its correctness can be shown by the LP-method.

The LP-method

Given an assignment M, define edge weights in G as follows. For any edge ab:

$$\mathrm{wt}_M(ab) = \begin{cases} 1 & \text{if } a \text{ prefers } b \text{ to } M(a); \\ -1 & \text{if } a \text{ prefers } M(a) \text{ to } b; \\ 0 & \text{otherwise.} \end{cases}$$

The LP-method

Given an assignment M, define edge weights in G as follows. For any edge ab:

$$\mathrm{wt}_M(ab) = \begin{cases} 1 & \text{if } a \text{ prefers } b \text{ to } M(a); \\ -1 & \text{if } a \text{ prefers } M(a) \text{ to } b; \\ 0 & \text{otherwise.} \end{cases}$$

Observation. For any assignment N:

$$\mathrm{wt}_M(N) = \sum_{e \in N} \mathrm{wt}_M(e) = \#\text{ of votes for } N - \#\text{ of votes for } M.$$

The LP-method

Given an assignment M, define edge weights in G as follows. For any edge ab:

$$\mathrm{wt}_M(ab) = \begin{cases} 1 & \text{if } a \text{ prefers } b \text{ to } M(a); \\ -1 & \text{if } a \text{ prefers } M(a) \text{ to } b; \\ 0 & \text{otherwise.} \end{cases}$$

OBSERVATION. For any assignment N:

$$\mathrm{wt}_M(N) = \sum_{e \in N} \mathrm{wt}_M(e) = \#\text{ of votes for } N - \#\text{ of votes for } M.$$

Thus M is popular $\iff$ $\mathrm{wt}_M(N) \leq 0$ for all assignments N in G.

Since $\mathrm{wt}_M(M) = 0$:

- M is popular $\iff$ M is a max-weight assignment under wt_M.

The LP-method

LP for max-weight assignment:

$$\max \sum_{e \in E} \mathrm{wt}_M(e) \cdot x_e$$

$$\sum_{e \in \delta(v)} x_e \quad = \quad 1 \quad \forall v \in A \cup B \quad \text{and} \quad x_e \geq 0 \quad \forall e \in E.$$

M is popular $\iff$ the optimal value of this LP is 0.

The LP-method

LP for max-weight assignment:

$$\max \sum_{e \in E} \mathrm{wt}_M(e) \cdot x_e$$

$$\sum_{e \in \delta(v)} x_e \quad = \quad 1 \quad \forall v \in A \cup B \quad \text{and} \quad x_e \geq 0 \quad \forall e \in E.$$

M is popular $\iff$ the optimal value of this LP is 0.

The dual LP:

$$\min \sum_{v} y_v$$

$$y_a + y_b \quad \geq \quad \mathrm{wt}_M(ab) \quad \forall ab \in E.$$

M is popular $\iff$ the optimal value of the dual LP is 0.

Dual certificates

CLAIM. M is a popular assignment $\iff$ $\exists$ dual feasible $\vec{\alpha}$ such that $\sum_v \alpha_v = 0$ and

- $\alpha_a \in \{0, 1, 2, \ldots, n-2, n-1\}$ for all $a \in A$;
- $\alpha_b \in \{0, -1, -2, \ldots, n-2, -(n-1)\}$ for all $b \in B$.

Dual certificates

CLAIM. M is a popular assignment $\iff$ $\exists$ dual feasible $\vec{\alpha}$ such that $\sum_v \alpha_v = 0$ and

- $\alpha_a \in \{0, 1, 2, \ldots, n-2, n-1\}$ for all $a \in A$;
- $\alpha_b \in \{0, -1, -2, \ldots, n-2, -(n-1)\}$ for all $b \in B$.

Suppose $\exists \ell : B \to \{0, 1, 2, \ldots, n-2, n-1\}$ s.t. G_ℓ admits a perfect matching M

Dual certificates

CLAIM. M is a popular assignment $\iff$ $\exists$ dual feasible $\vec{\alpha}$ such that $\sum_v \alpha_v = 0$ and

- $\alpha_a \in \{0, 1, 2, \ldots, n-2, n-1\}$ for all $a \in A$;
- $\alpha_b \in \{0, -1, -2, \ldots, n-2, -(n-1)\}$ for all $b \in B$.

Suppose $\exists \ell : B \to \{0, 1, 2, \ldots, n-2, n-1\}$ s.t. G_ℓ admits a perfect matching M

$\Rightarrow$ M is a popular assignment in G.

Dual certificates

CLAIM. M is a popular assignment $\iff$ $\exists$ dual feasible $\vec{\alpha}$ such that $\sum_v \alpha_v = 0$ and

- $\alpha_a \in \{0, 1, 2, \ldots, n-2, n-1\}$ for all $a \in A$;
- $\alpha_b \in \{0, -1, -2, \ldots, n-2, -(n-1)\}$ for all $b \in B$.

Suppose $\exists \ell : B \to \{0, 1, 2, \ldots, n-2, n-1\}$ s.t. G_ℓ admits a perfect matching M

$\Rightarrow$ M is a popular assignment in G.

- $\alpha_b = -\ell(b)$ for $b \in B$ and $\alpha_a = \ell(M(a))$ for $a \in A$ is M's dual certificate.

Dual certificates

CLAIM. M is a popular assignment $\iff$ $\exists$ dual feasible $\vec{\alpha}$ such that $\sum_v \alpha_v = 0$ and

- $\alpha_a \in \{0, 1, 2, \ldots, n-2, n-1\}$ for all $a \in A$;
- $\alpha_b \in \{0, -1, -2, \ldots, n-2, -(n-1)\}$ for all $b \in B$.

Suppose $\exists \ell : B \to \{0, 1, 2, \ldots, n-2, n-1\}$ s.t. G_ℓ admits a perfect matching M

$\Rightarrow$ M is a popular assignment in G.

- $\alpha_b = -\ell(b)$ for $b \in B$ and $\alpha_a = \ell(M(a))$ for $a \in A$ is M's dual certificate.

Conversely, G has a popular assignment M, i.e., $\exists$ a dual certificate $\vec{\alpha}$ for M

Dual certificates

CLAIM. M is a popular assignment $\iff$ $\exists$ dual feasible $\vec{\alpha}$ such that $\sum_v \alpha_v = 0$ and

- $\alpha_a \in \{0, 1, 2, \ldots, n-2, n-1\}$ for all $a \in A$;
- $\alpha_b \in \{0, -1, -2, \ldots, n-2, -(n-1)\}$ for all $b \in B$.

Suppose $\exists \ell : B \to \{0, 1, 2, \ldots, n-2, n-1\}$ s.t. G_ℓ admits a perfect matching M

$\Rightarrow$ M is a popular assignment in G.

- $\alpha_b = -\ell(b)$ for $b \in B$ and $\alpha_a = \ell(M(a))$ for $a \in A$ is M's dual certificate.

Conversely, G has a popular assignment M, i.e., $\exists$ a dual certificate $\vec{\alpha}$ for M

$\Rightarrow$ M is a perfect matching in G_ℓ where $\ell(b) = -\alpha_b$ for all $b \in B$.

Dual certificates

CLAIM. M is a popular assignment $\iff$ $\exists$ dual feasible $\vec{\alpha}$ such that $\sum_v \alpha_v = 0$ and

- $\alpha_a \in \{0, 1, 2, \ldots, n-2, n-1\}$ for all $a \in A$;
- $\alpha_b \in \{0, -1, -2, \ldots, n-2, -(n-1)\}$ for all $b \in B$.

Suppose $\exists \ell : B \to \{0, 1, 2, \ldots, n-2, n-1\}$ s.t. G_ℓ admits a perfect matching M

$\Rightarrow$ M is a popular assignment in G.

- $\alpha_b = -\ell(b)$ for $b \in B$ and $\alpha_a = \ell(M(a))$ for $a \in A$ is M's dual certificate.

Conversely, G has a popular assignment M, i.e., $\exists$ a dual certificate $\vec{\alpha}$ for M

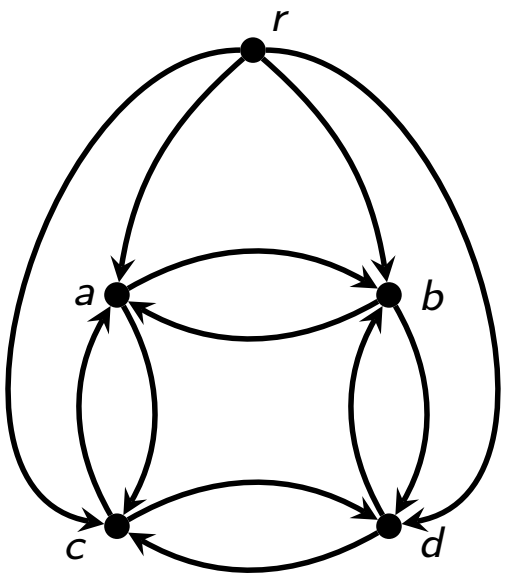
$\Rightarrow$ M is a perfect matching in G_ℓ where $\ell(b) = -\alpha_b$ for all $b \in B$.

- So $\exists\ \ell : B \to \{0, 1, 2, \ldots, n-2, n-1\}$ as desired.
- Our algorithm finds the minimal such ℓ-function.

From assignments to arborescences

Input to the arborescence problem: A directed graph $G = (V \cup \{r\}, E)$.

- The vertex r has no incoming edge—it is the root.

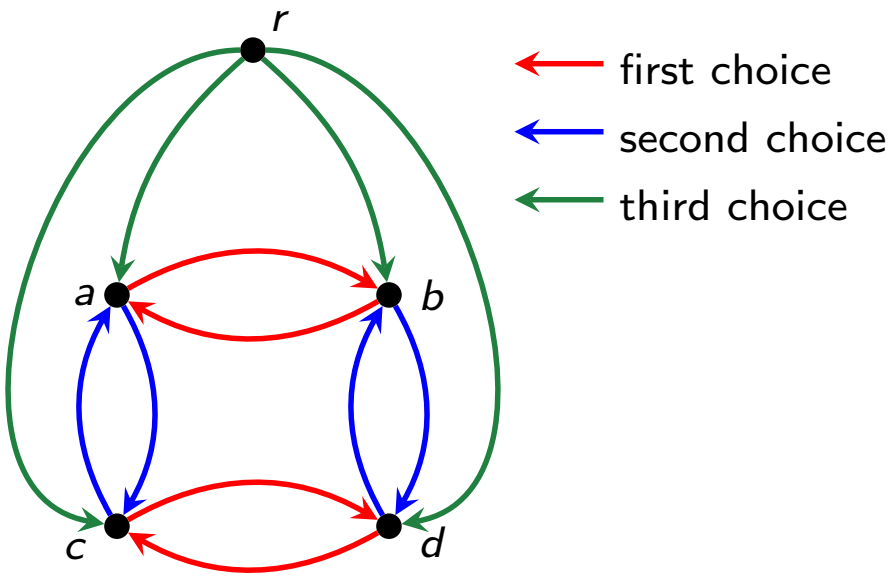


- Every vertex in V has a preference order on its incoming edges.

From assignments to arborescences

Input to the arborescence problem: A directed graph $G = (V \cup \{r\}, E)$.

- The vertex r has no incoming edge—it is the root.

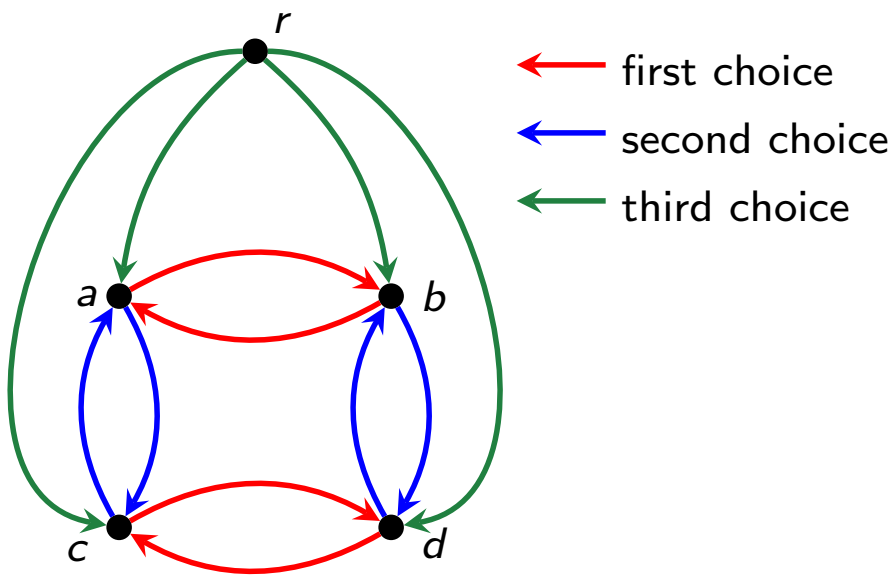


- Every vertex in V has a preference order on its incoming edges.

From assignments to arborescences

Input to the arborescence problem: A directed graph $G = (V \cup \{r\}, E)$.

- The vertex r has no incoming edge—it is the root.



What we seek in G:

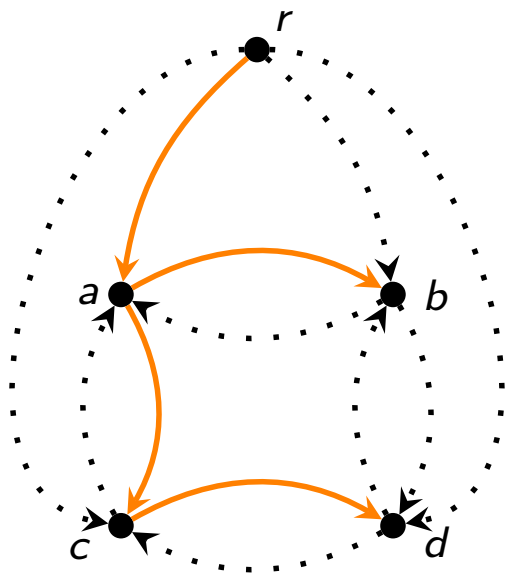
- an optimal arborescence.

- Every vertex in V has a preference order on its incoming edges.

Arborescences

An arborescence A is a directed tree rooted at r.

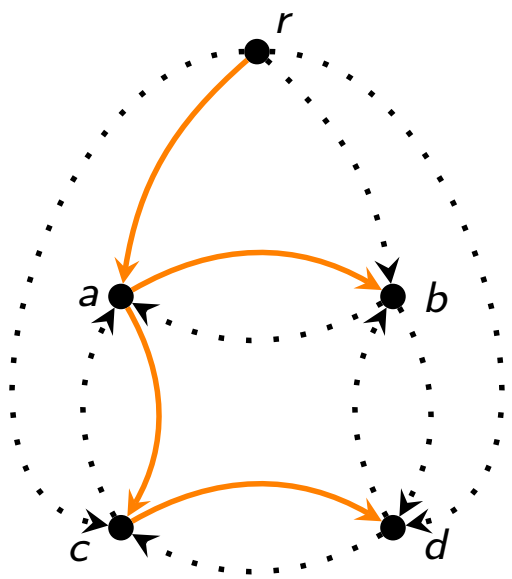
- Every $v \in V$ has a unique incoming edge in A.



Arborescences

An arborescence A is a directed tree rooted at r.

- Every $v \in V$ has a unique incoming edge in A.



- Every $v \in V$ cares only about its incoming edge in A.
  - Better the incoming edge of $v \Rightarrow$ more satisfied is v with A.

Comparing any two arborescences

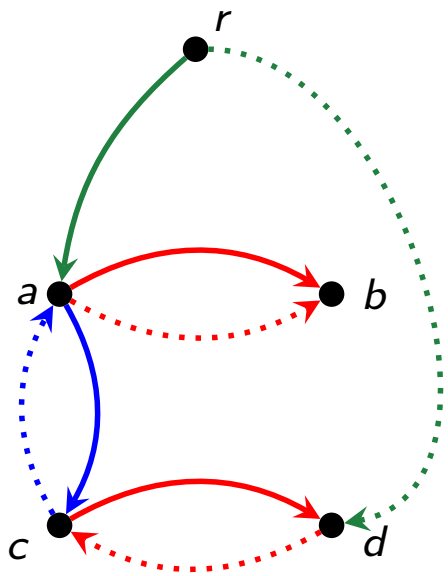
Any pair of arborescences A and A' can be compared as follows:

- each vertex v prefers the arborescence where it has a better incoming edge.

Comparing any two arborescences

Any pair of arborescences A and A' can be compared as follows:

- each vertex v prefers the arborescence where it has a better incoming edge.

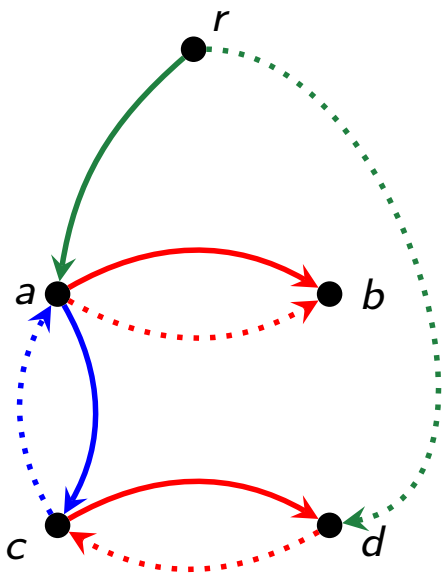


- A: solid arborescence vs A': dotted arborescence.
- a prefers A' to A since it prefers ca to ra;
- b is indifferent between A and A';
- c prefers A' to A since it prefers dc to ac;
- d prefers A to A' since it prefers cd to rd;

Comparing any two arborescences

Any pair of arborescences A and A' can be compared as follows:

- each vertex v prefers the arborescence where it has a better incoming edge.



- A: solid arborescence vs A': dotted arborescence.
- a prefers A' to A since it prefers ca to ra;
- b is indifferent between A and A';
- c prefers A' to A since it prefers dc to ac;
- d prefers A to A' since it prefers cd to rd;

A' *is more popular than* A or $A' \succ A$ if

$$\#\text{ of vertices that prefer } A' > \#\text{ of vertices that prefer } A.$$

Popular Arborescences

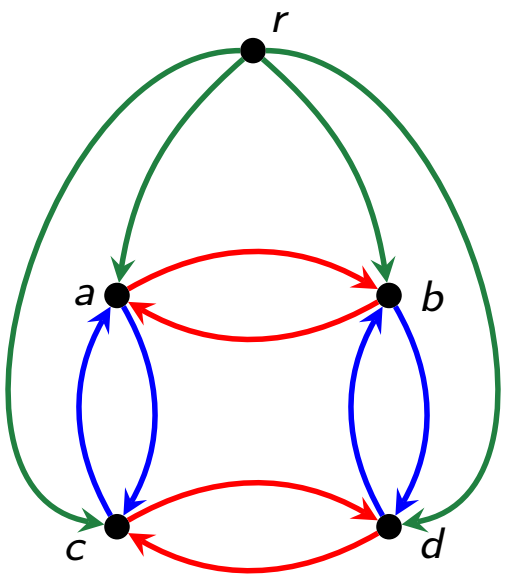
Arborescence A is popular if there is no arborescence A' such that $A' \succ A$.

- What we seek is a popular arborescence in G.

Popular Arborescences

Arborescence A is popular if there is no arborescence A' such that $A' \succ A$.

- What we seek is a popular arborescence in G.
- A motivating application is *liquid democracy* where in an election:
  - each voter may delegate its vote to another voter;
  - delegations are transitive.



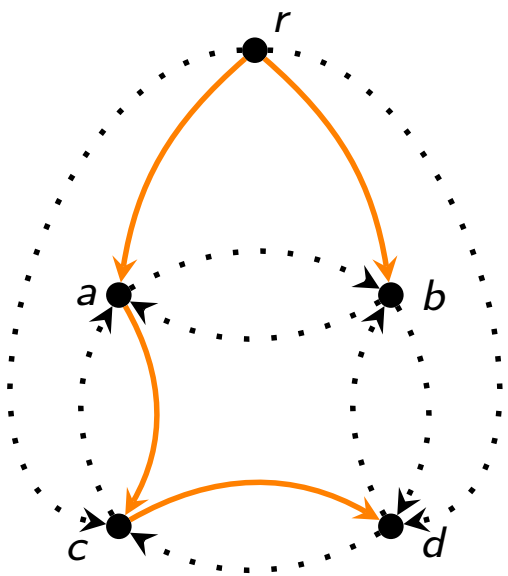
first choice
second choice
third choice

- r is not a voter.
- Each voter declares a set of acceptable representatives.
- Each voter has a preference order among its representatives.
- Delegation cycles are forbidden.

Motivation: Liquid Democracy

An arborescence A captures the entire sequence of delegations.

- Children of r in A cast weighted votes on behalf of all their descendants.

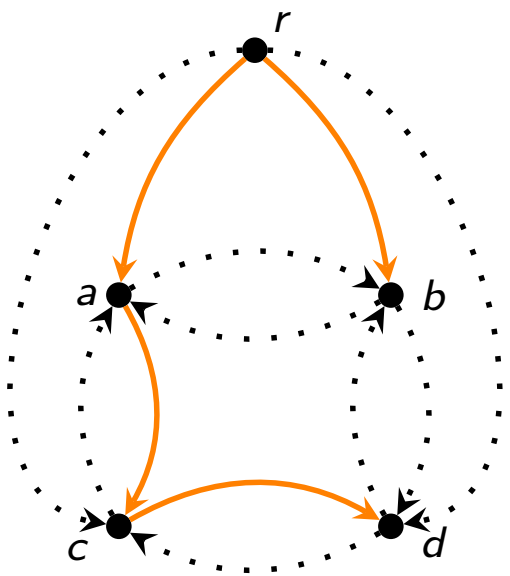


- a casts a vote of weight 3 on behalf of a, c, d.
- b casts its own vote.

Motivation: Liquid Democracy

An arborescence A captures the entire sequence of delegations.

- Children of r in A cast weighted votes on behalf of all their descendants.

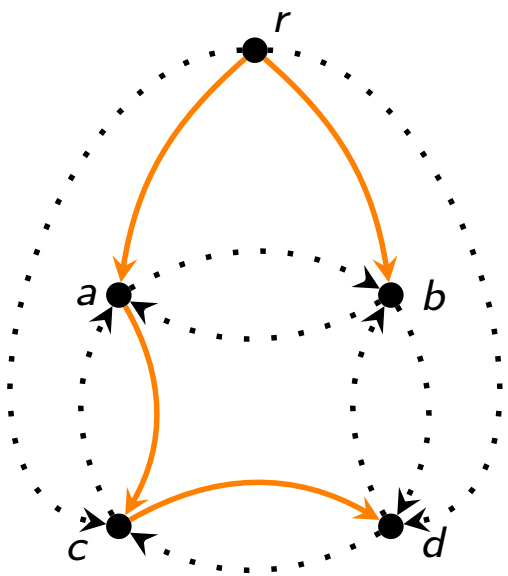


- a casts a vote of weight 3 on behalf of a, c, d.
- b casts its own vote.
- Used for internal decision making at Google.
- Also in some political parties in Europe, e.g., the German Pirate Party.

Motivation: Liquid Democracy

An arborescence A captures the entire sequence of delegations.

- Children of r in A cast weighted votes on behalf of all their descendants.



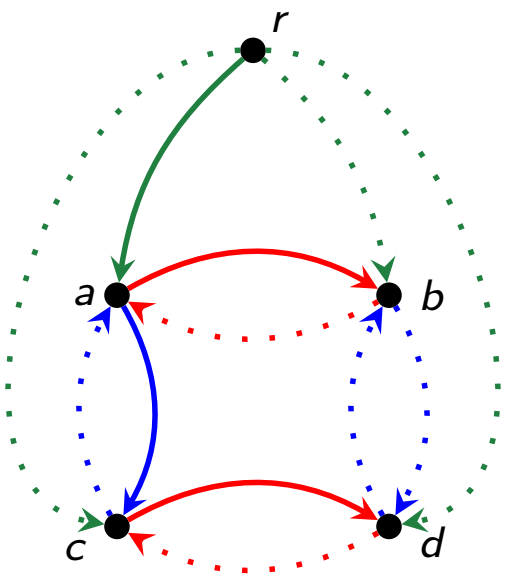
- a casts a vote of weight 3 on behalf of a, c, d.
- b casts its own vote.
- Used for internal decision making at Google.
- Also in some political parties in Europe, e.g., the German Pirate Party.

A popular arborescence represents a stable way of delegating votes.

- Does every instance $G = (V \cup \{r\}, E)$ admit a popular arborescence?

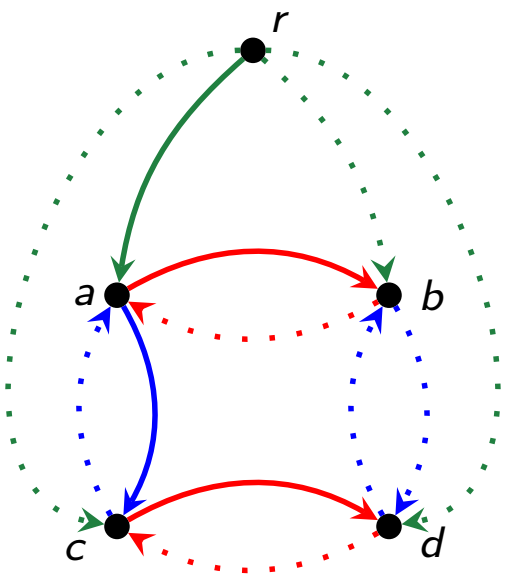
Our example

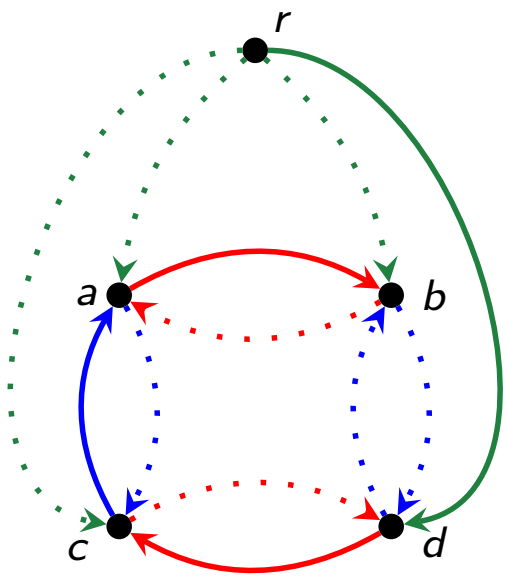
We know $A = \{ra, ab, ac, cd\}$ is not a popular arborescence.



Our example

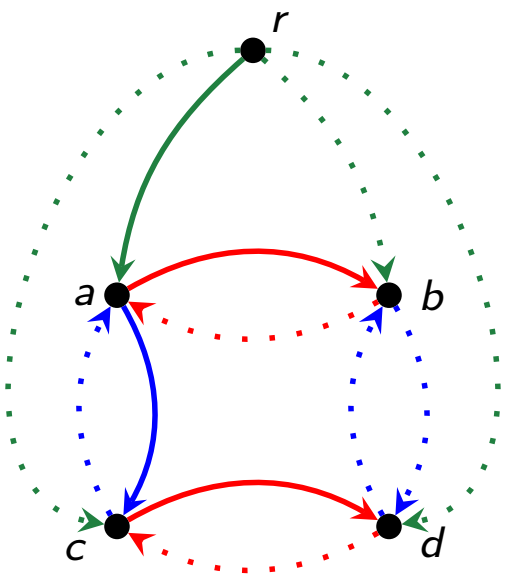
We know $A = \{ra, ab, ac, cd\}$ is not a popular arborescence.

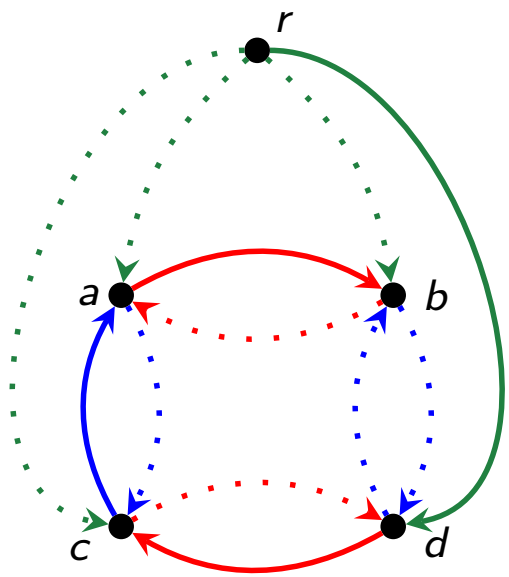


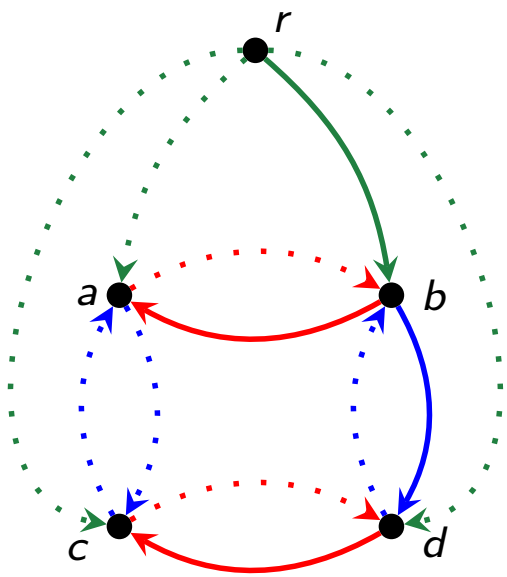


Our example

We know $A = \{ra, ab, ac, cd\}$ is not a popular arborescence.

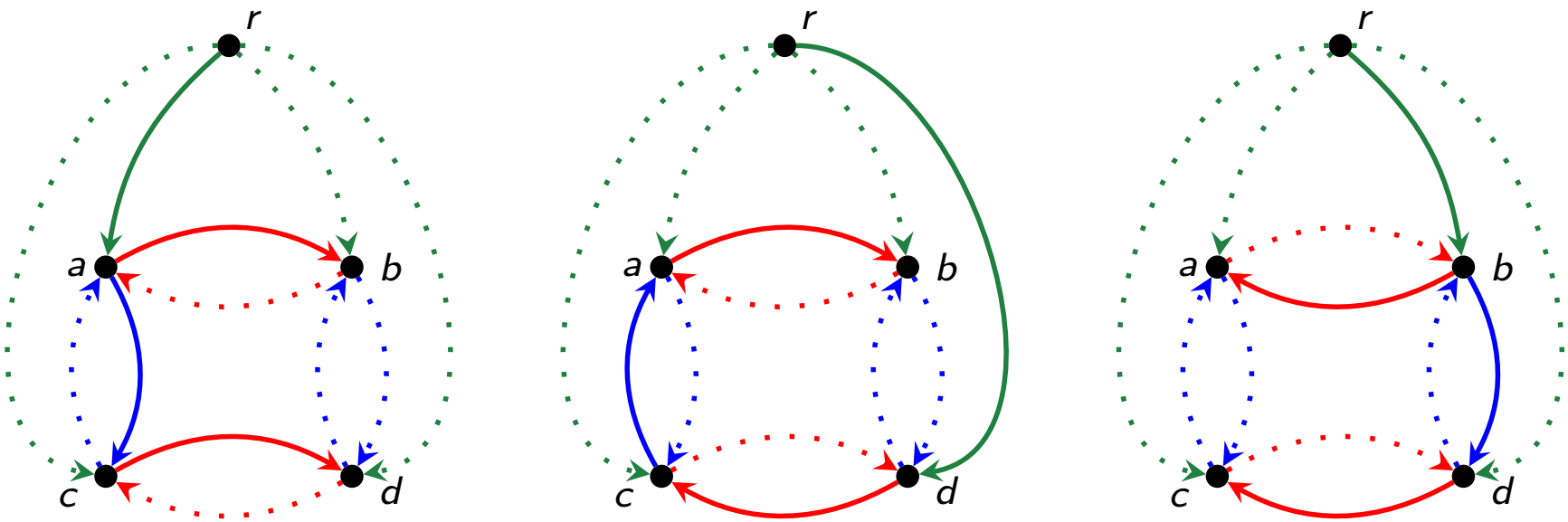






Our example

We know $A = \{ra, ab, ac, cd\}$ is not a popular arborescence.

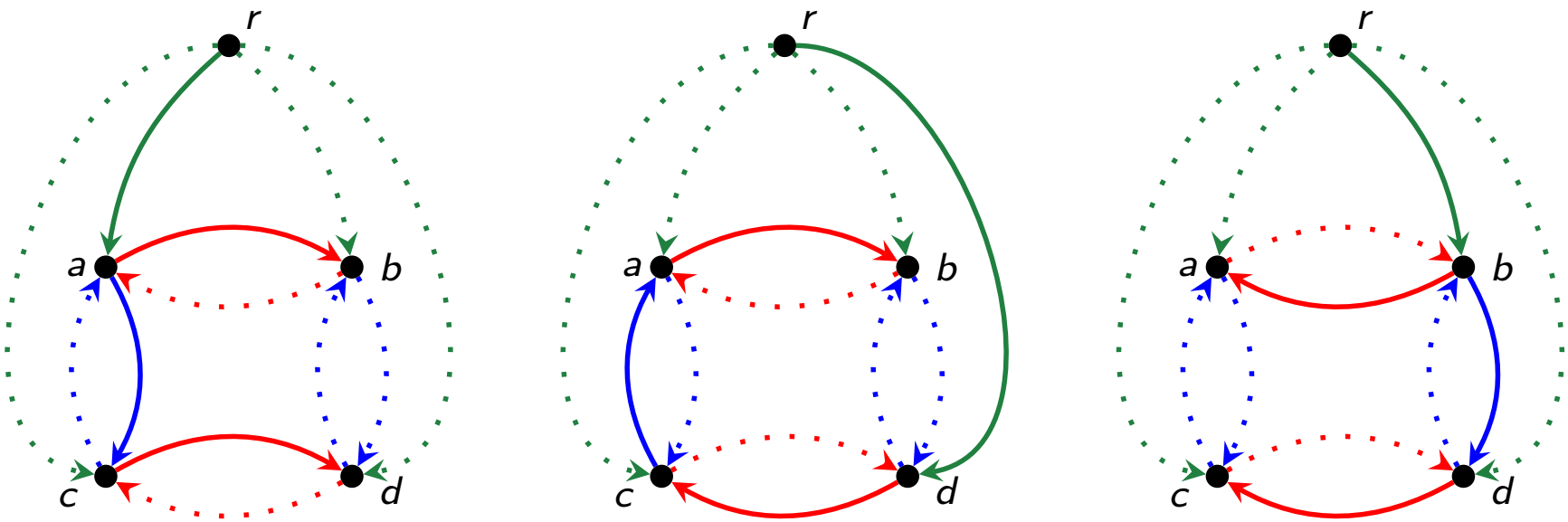


For every arborescence here: there is a more popular arborescence.

- This instance has no popular arborescence.

Our example

We know $A = \{ra, ab, ac, cd\}$ is not a popular arborescence.



For every arborescence here: there is a more popular arborescence.

- This instance has no popular arborescence.

Our problem: Decide if the given instance admits a popular arborescence; if so, find it.

Borrowing ideas from the popular assignment algorithm

Let us use a level function $\ell : E \to \{0, 1, 2, \ldots, n-1\}$

- For each $v \in V$: let $\ell^*(v) = \underbrace{\max\{\ell(e) : e \in \delta^-(v)\}}_{\text{highest level among } v\text{'s incoming edges}}$

Borrowing ideas from the popular assignment algorithm

Let us use a level function $\ell : E \to \{0, 1, 2, \ldots, n-1\}$

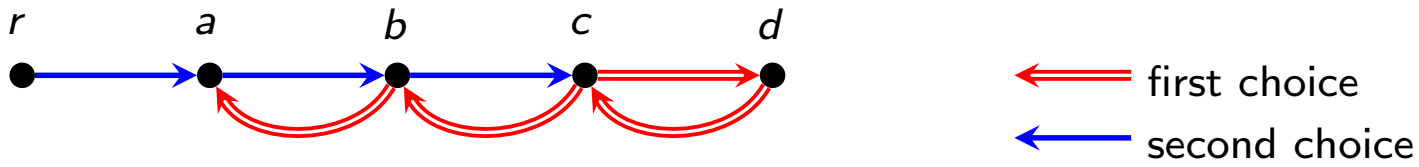
- For each $v \in V$: let $\ell^*(v) = \underbrace{\max\{\ell(e) : e \in \delta^-(v)\}}_{\text{highest level among } v\text{'s incoming edges}}$

Define a subgraph $G_\ell = (V \cup \{r\}, E_\ell)$ of G as follows.

- Incident to each $v \in V$ are (i) its favorite incoming edges in level $\ell^*(v)$ and
- (ii) its favorite incoming edges in level $\ell^*(v)-1$ if v prefers these to all its incoming edges in level $\ell^*(v)$.

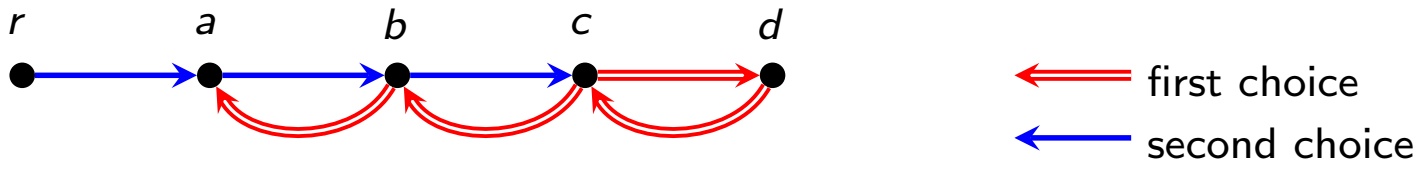
The subgraph G_ℓ

An example:



The subgraph G_ℓ

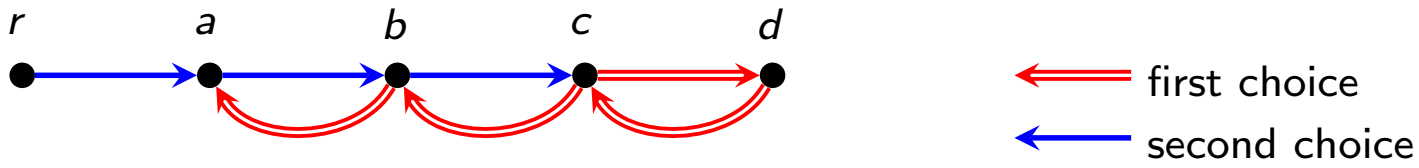
An example:



- Suppose $\ell(ra) = 3, \ell(ab) = \ell(ba) = 2$,
- $\ell(bc) = \ell(cb) = 1$ and $\ell(cd) = \ell(dc) = 0$.

The subgraph G_ℓ

An example:



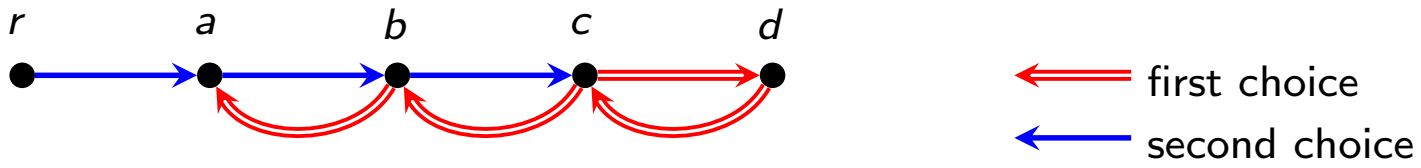
- Suppose $\ell(ra) = 3, \ell(ab) = \ell(ba) = 2$,
- $\ell(bc) = \ell(cb) = 1$ and $\ell(cd) = \ell(dc) = 0$.

Then $\ell^*(a) = 3, \ell^*(b) = 2, \ell^*(c) = 1$, and $\ell^*(d) = 0$.

- The subgraph $G_\ell = G$.

The subgraph G_ℓ

An example:



- Suppose $\ell(ra) = 3, \ell(ab) = \ell(ba) = 2$,
- $\ell(bc) = \ell(cb) = 1$ and $\ell(cd) = \ell(dc) = 0$.

Then $\ell^*(a) = 3, \ell^*(b) = 2, \ell^*(c) = 1$, and $\ell^*(d) = 0$.

- The subgraph $G_\ell = G$.

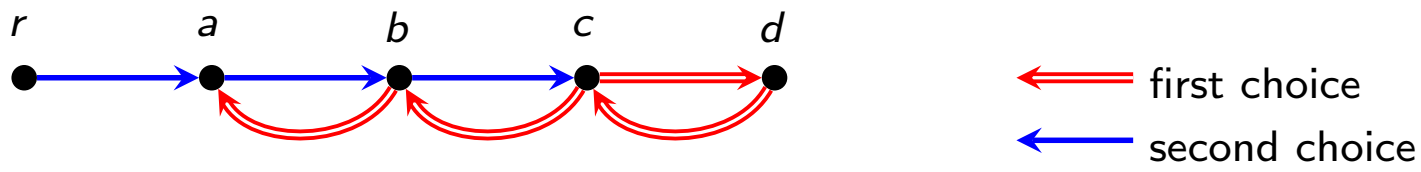
THEOREM. G admits a popular arborescence if and only if

- $\exists \ell : E \to \{0, 1, 2, \ldots, n-1\}$ such that G_ℓ admits an arborescence A where:

undir(A) is a MST in the underlying undirected graph of G where cost(e) $= \ell(e)$.

The subgraph G_ℓ

An example:



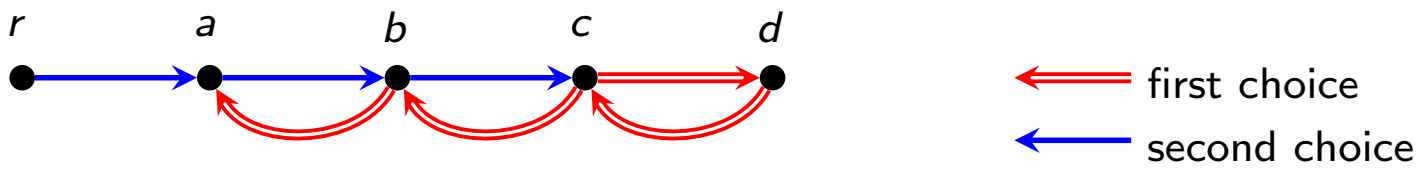
- Suppose $\ell(ra) = 3, \ell(ab) = \ell(ba) = 2$,
- $\ell(bc) = \ell(cb) = 1$ and $\ell(cd) = \ell(dc) = 0$.

Then $\ell^*(a) = 3, \ell^*(b) = 2, \ell^*(c) = 1$, and $\ell^*(d) = 0$.

- The subgraph $G_\ell = G$.

The subgraph G_ℓ

An example:

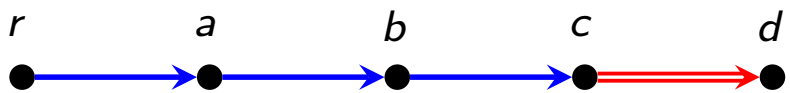


- Suppose $\ell(ra) = 3, \ell(ab) = \ell(ba) = 2$,
- $\ell(bc) = \ell(cb) = 1$ and $\ell(cd) = \ell(dc) = 0$.

Then $\ell^*(a) = 3, \ell^*(b) = 2, \ell^*(c) = 1$, and $\ell^*(d) = 0$.

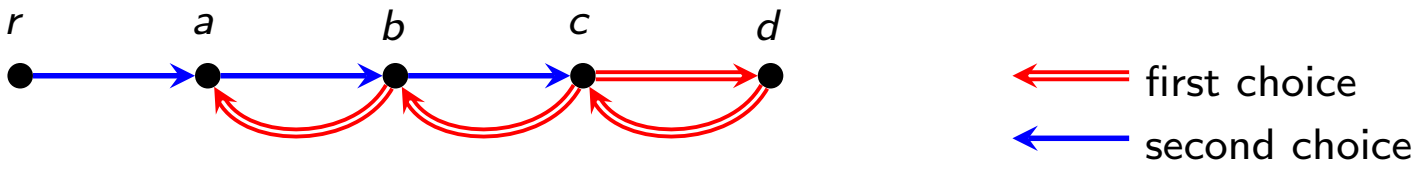
- The subgraph $G_\ell = G$.

- $A = \{ra, ab, bc, cd\}$ satisfies our desired condition.



The subgraph G_ℓ

An example:

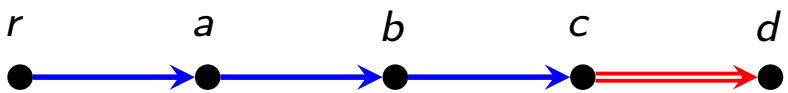


- Suppose $\ell(ra) = 3, \ell(ab) = \ell(ba) = 2$,
- $\ell(bc) = \ell(cb) = 1$ and $\ell(cd) = \ell(dc) = 0$.

Then $\ell^*(a) = 3, \ell^*(b) = 2, \ell^*(c) = 1$, and $\ell^*(d) = 0$.

- The subgraph $G_\ell = G$.

- $A = \{ra, ab, bc, cd\}$ satisfies our desired condition.



- Find the right function ℓ on E s.t.

  G_ℓ admits a desired arborescence

- or decide no such ℓ-function exists.

Our algorithm [KMSY 2025]

1. Initialize $\ell(e) = 0$ for all $e \in E$.
2. Construct the subgraph G_ℓ.
3. Find an ℓ-greedy branching B in G_ℓ.

> *B has as many level 0 edges as possible, subject to this, as many level 1 edges as possible, and so on.*

Our algorithm [KMSY 2025]

1. Initialize $\ell(e) = 0$ for all $e \in E$.
2. Construct the subgraph G_ℓ.
3. Find an ℓ-greedy branching B in G_ℓ.

> *B has as many level 0 edges as possible, subject to this, as many level 1 edges as possible, and so on.*

4. If undir(B) is a MST in the underlying undirected graph of G then return B.

Our algorithm [KMSY 2025]

1. Initialize $\ell(e) = 0$ for all $e \in E$.
2. Construct the subgraph G_ℓ.
3. Find an ℓ-greedy branching B in G_ℓ.

> *B has as many level 0 edges as possible, subject to this, as many level 1 edges as possible, and so on.*

4. If undir(B) is a MST in the underlying undirected graph of G then return B.
5. Let k be the least level with an edge e that violates the "MST-ness" of undir(B).

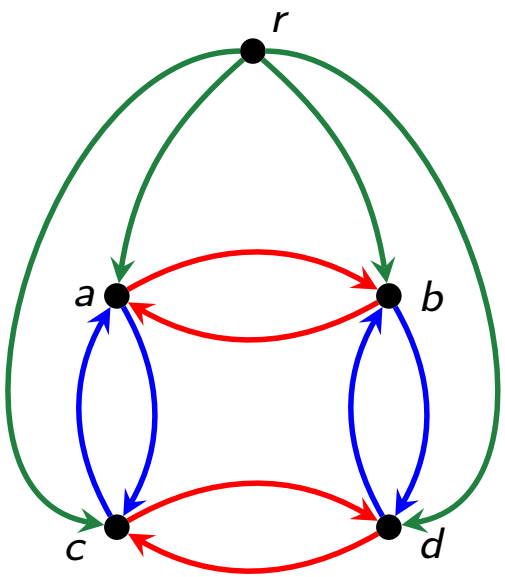
Our algorithm [KMSY 2025]

1. Initialize $\ell(e) = 0$ for all $e \in E$.
2. Construct the subgraph G_ℓ.
3. Find an ℓ-greedy branching B in G_ℓ.

> *B has as many level 0 edges as possible, subject to this, as many level 1 edges as possible, and so on.*

4. If undir(B) is a MST in the underlying undirected graph of G then return B.
5. Let k be the least level with an edge e that violates the "MST-ness" of undir(B).
   - For every violating edge e in level k do: $\ell(e) = \ell(e) + 1$.
   - If $\ell(e) = n$ for any $e \in E$ then return "no"; else go back to Step 2.

The algorithm on our starting example



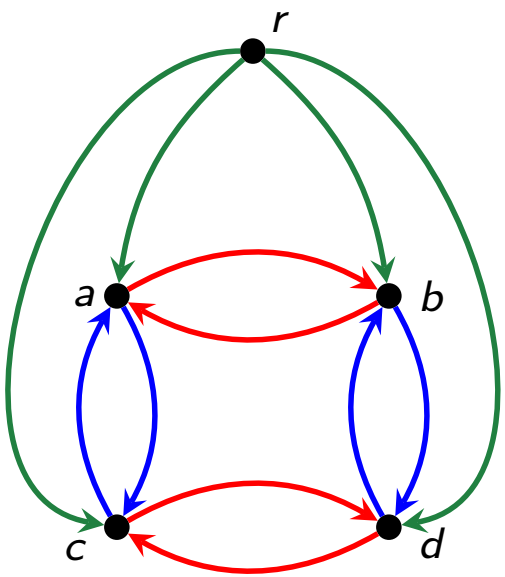
first choice

second choice

third choice

- Initially $\ell(e) = 0$ for all edges e.
- G_ℓ consists of only red edges.

The algorithm on our starting example

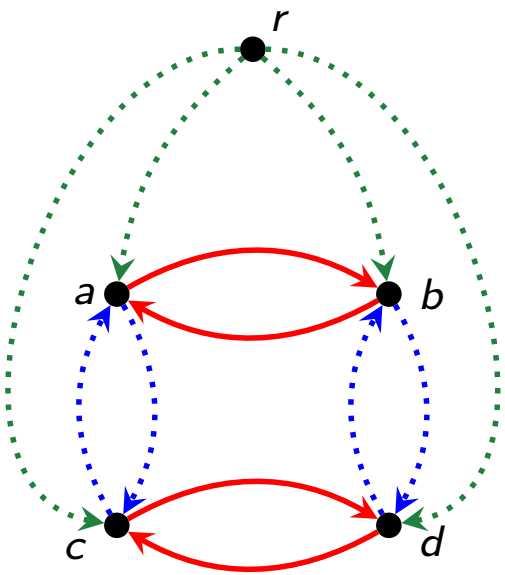


first choice

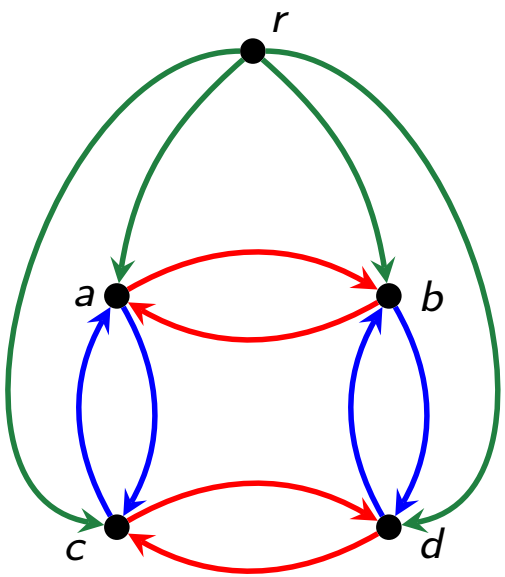
second choice

third choice

- Initially $\ell(e) = 0$ for all edges e.
- G_ℓ consists of only red edges.



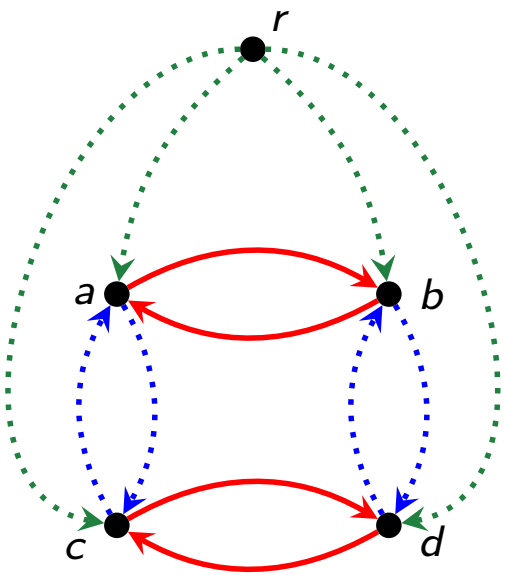
The algorithm on our starting example



first choice

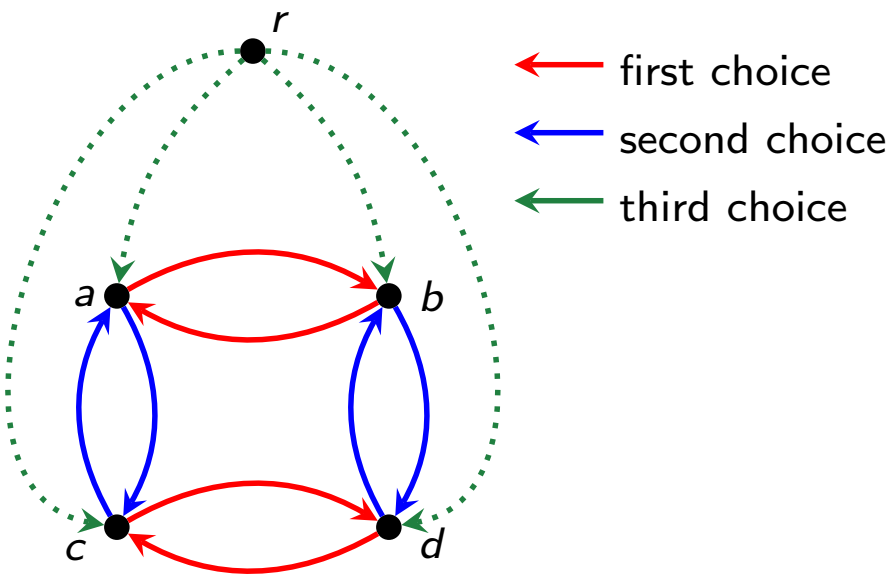
second choice

third choice



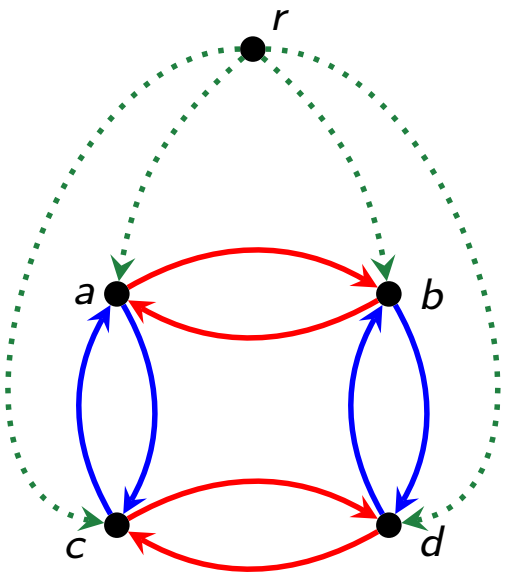
- Initially $\ell(e) = 0$ for all edges e.
- G_ℓ consists of only red edges.
- Say $B = \{ab, cd\}$.
- All blue/green edges are violating.
- So $\ell(e) = 1 \ \forall$ blue/green edges e.

The algorithm on our starting example



- Now $\ell(e) = 0$ for all red edges e and
- $\ell(e) = 1$ for all blue/green edges e.
- G_ℓ has all red and blue edges.

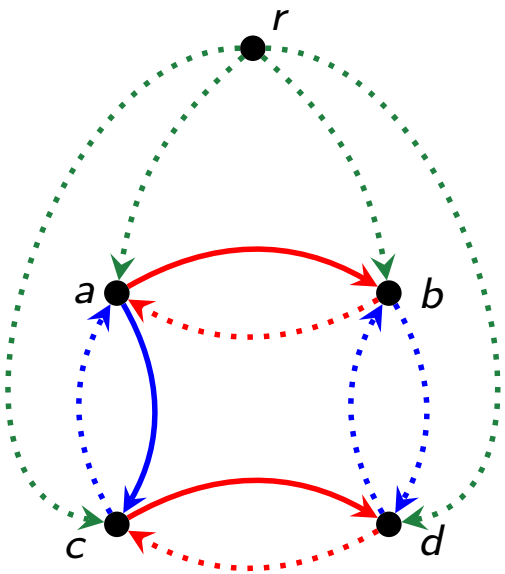
The algorithm on our starting example



first choice

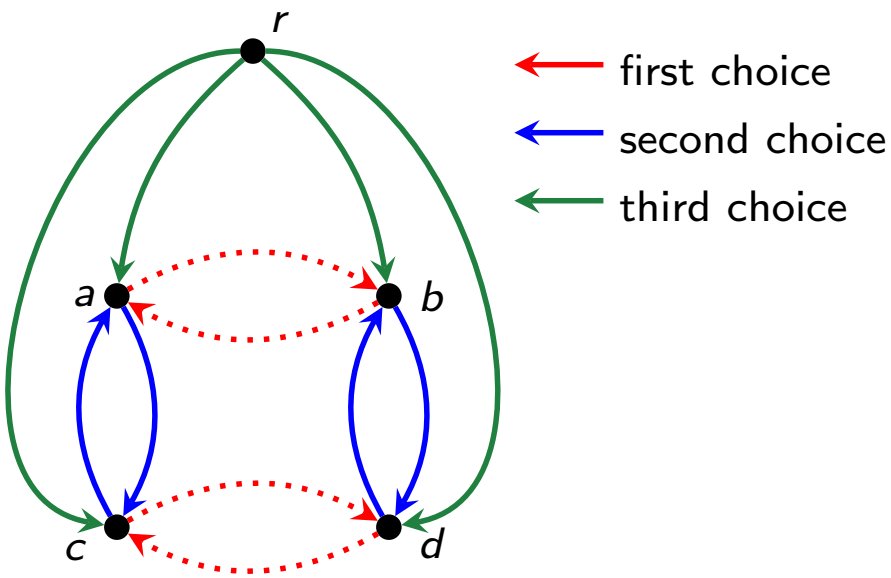
second choice

third choice



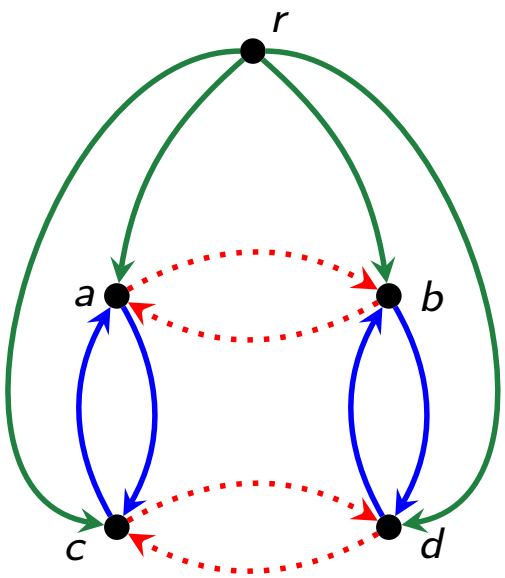
- Now $\ell(e) = 0$ for all red edges e and
- $\ell(e) = 1$ for all blue/green edges e.
- G_ℓ has all red and blue edges.
- Say $B = \{ab, ac, cd\}$.
- All green edges are violating.
- So $\ell(e) = 2$ $\forall$ green edges e.

The algorithm on our starting example



- Now $\ell(e) = 0$ for all red edges e and
- $\ell(e) = 1$ for all blue edges e and
- $\ell(e) = 2$ for all green edges e.
- G_ℓ has all blue and green edges.

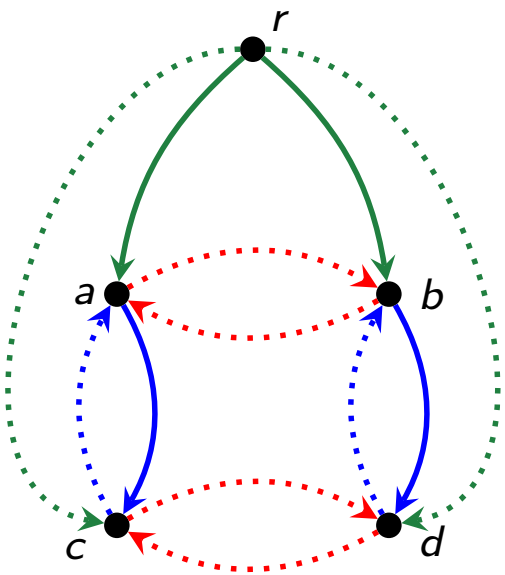
The algorithm on our starting example



first choice

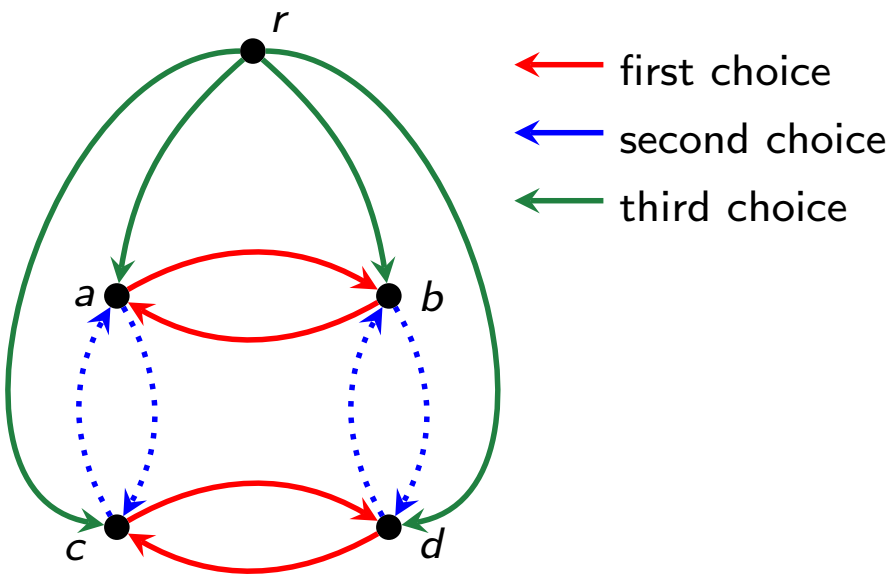
second choice

third choice



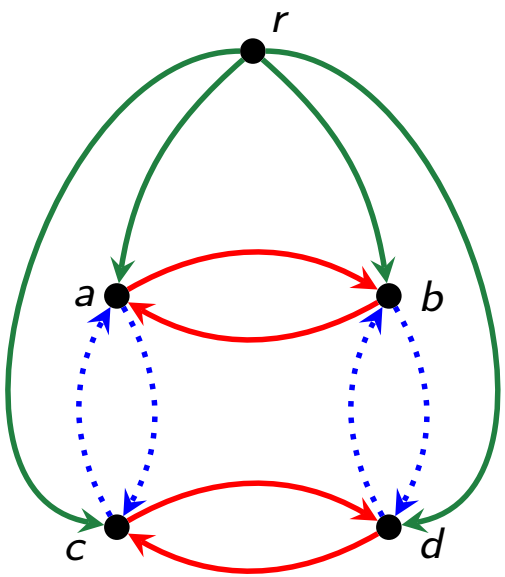
- Now $\ell(e) = 0$ for all red edges e and
- $\ell(e) = 1$ for all blue edges e and
- $\ell(e) = 2$ for all green edges e.
- G_ℓ has all blue and green edges.
- Say $B = \{ra, rb, ac, bd\}$.
- All red edges are violating.
- So $\ell(e) = 1$ $\forall$ red edges e.

The algorithm on our starting example



- Now $\ell(e) = 1$ for all red/blue edges e
- and $\ell(e) = 2$ for all green edges e.
- G_ℓ has all red and green edges.

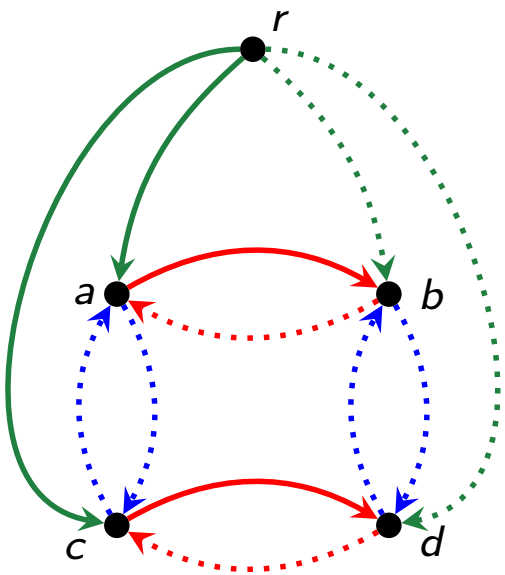
The algorithm on our starting example



first choice

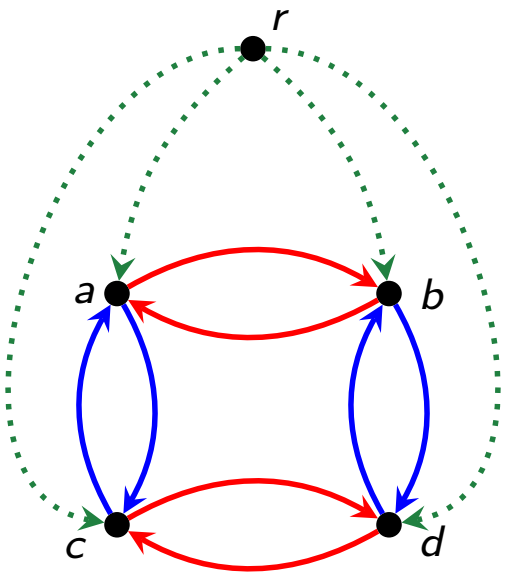
second choice

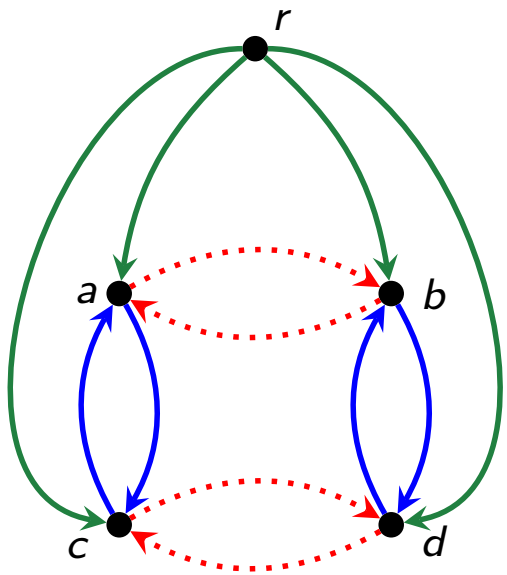
third choice

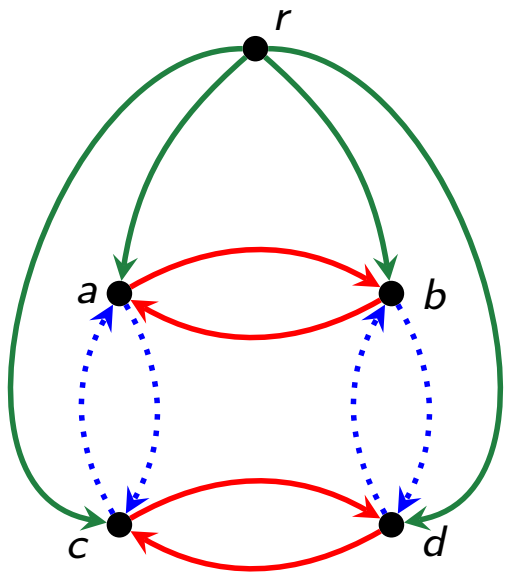


- Now $\ell(e) = 1$ for all red/blue edges e
- and $\ell(e) = 2$ for all green edges e.
- G_ℓ has all red and green edges.
- Say $B = \{ra, rc, ab, cd\}$.
- All blue edges are violating.
- So $\ell(e) = 2$ $\forall$ blue edges e.

The algorithm on our starting example



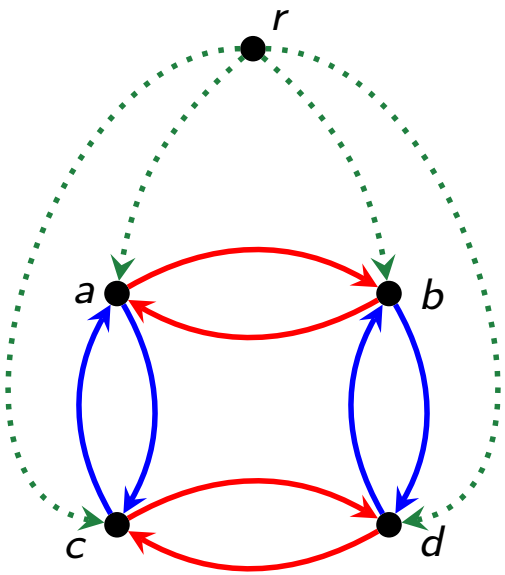


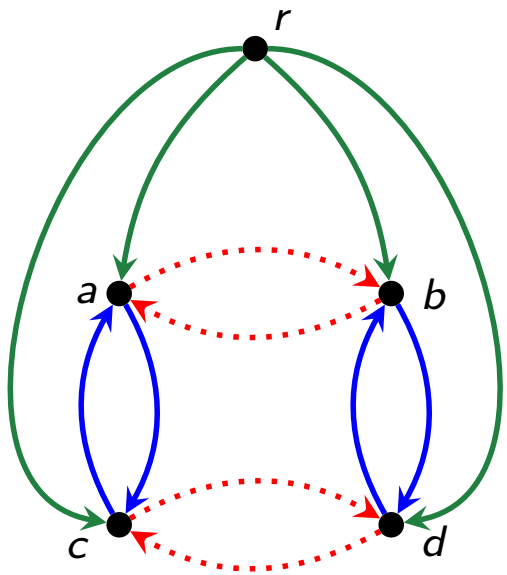


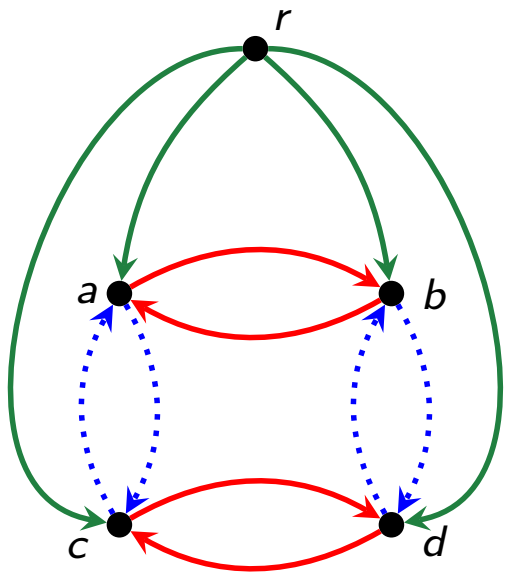
Now G_ℓ goes back to being the red-blue subgraph.

- Then to being the blue-green subgraph.
- And yet again to being the red-green subgraph.

The algorithm on our starting example







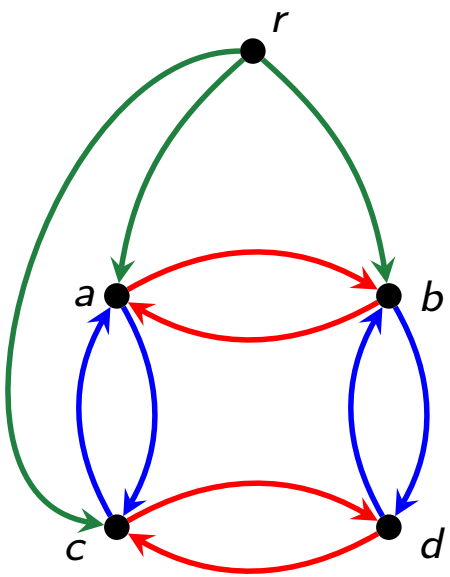
Now G_ℓ goes back to being the red-blue subgraph.

- Then to being the blue-green subgraph.
- And yet again to being the red-green subgraph.

Eventually $\ell(e) = 4$ for green edges e.

- Then the algorithm says "no".

Another example



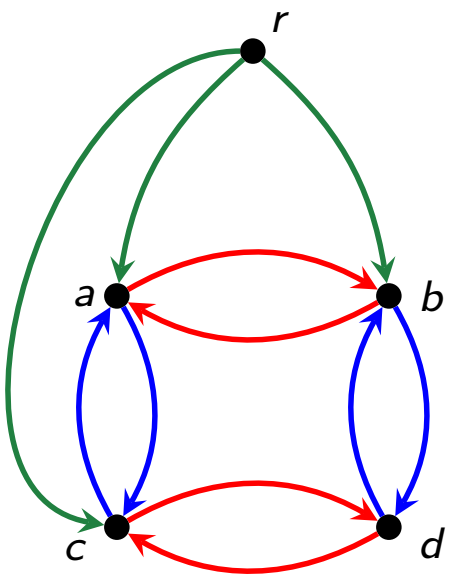
first choice

second choice

third choice

- Initially $\ell(e) = 0$ for all edges e.
- G_ℓ consists of only red edges.

Another example

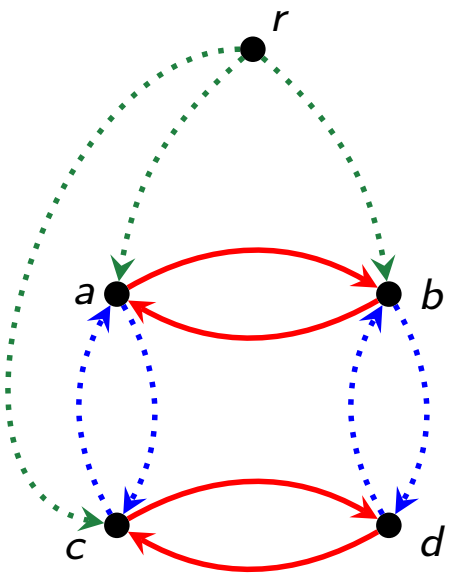


first choice

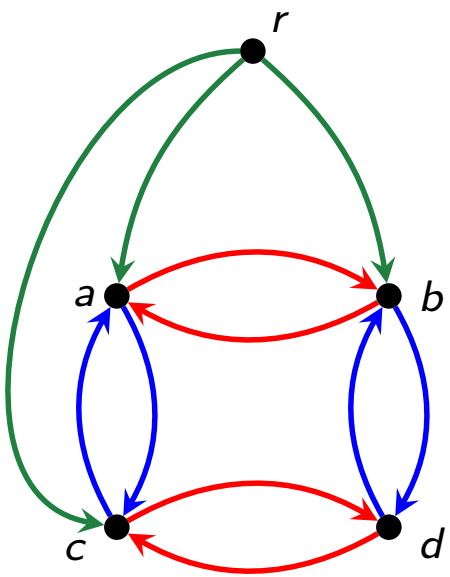
second choice

third choice

- Initially $\ell(e) = 0$ for all edges e.
- G_ℓ consists of only red edges.



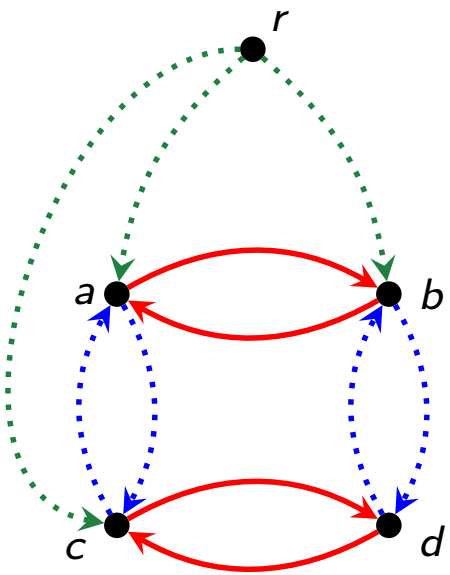
Another example



first choice

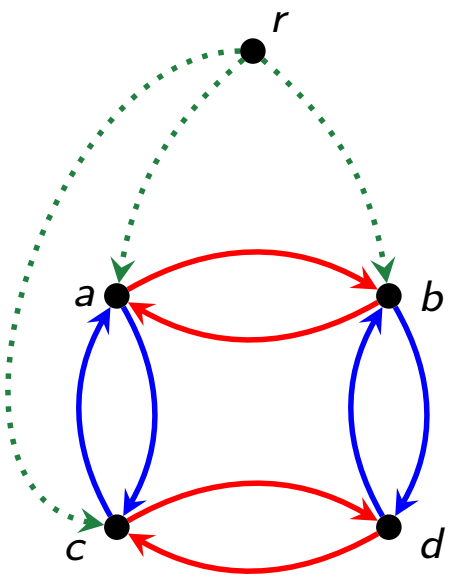
second choice

third choice



- Initially $\ell(e) = 0$ for all edges e.
- G_ℓ consists of only red edges.
- Say $B = \{ab, cd\}$.
- All blue/green edges are violating.
- So $\ell(e) = 1 \ \forall$ blue/green edges e.

Another example



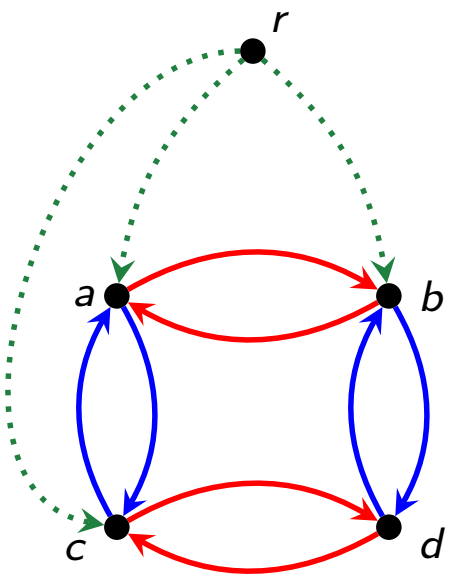
first choice

second choice

third choice

- Now $\ell(e) = 0$ for all red edges e and
- $\ell(e) = 1$ for all blue/green edges e.
- G_ℓ has all red and blue edges.

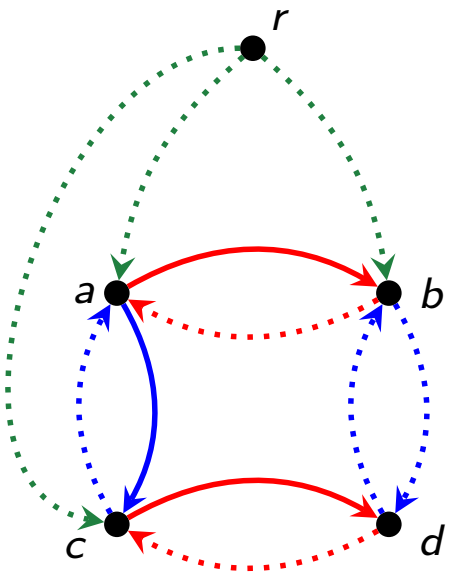
Another example



first choice

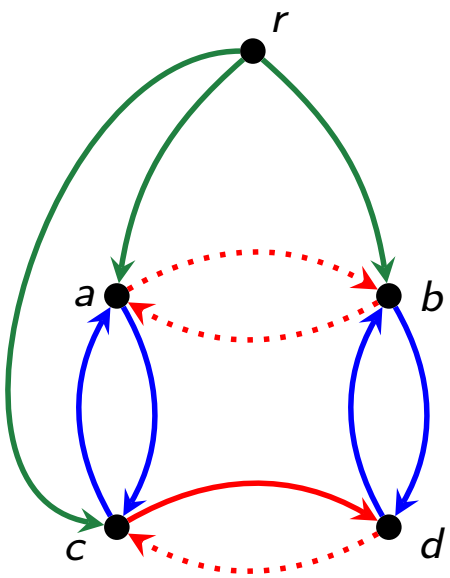
second choice

third choice



- Now $\ell(e) = 0$ for all red edges e and
- $\ell(e) = 1$ for all blue/green edges e.
- G_ℓ has all red and blue edges.
- Say $B = \{ab, ac, cd)\}$.
- All green edges are violating.
- So $\ell(e) = 2$ $\forall$ green edges e.

Another example



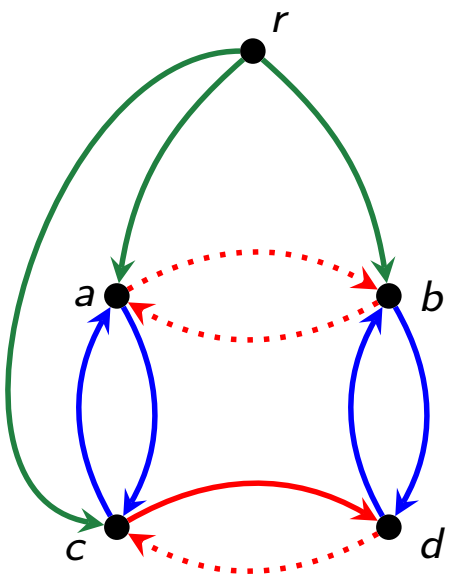
first choice

second choice

third choice

- Now $\ell(e) = 0$ for all red edges e and
- $\ell(e) = 1$ for all blue edges e and
- $\ell(e) = 2$ for all green edges e.
- G_ℓ has all blue/green edges $+$ cd.

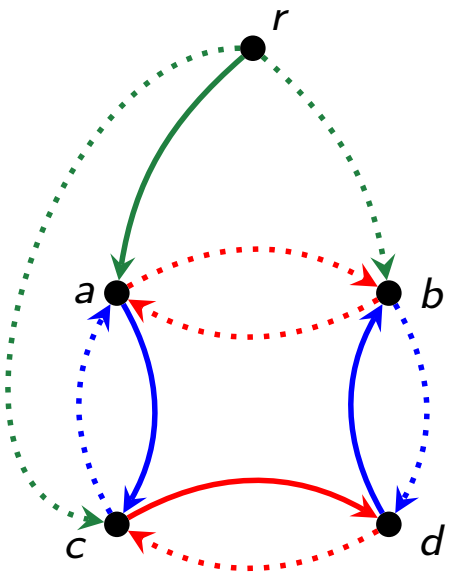
Another example



first choice

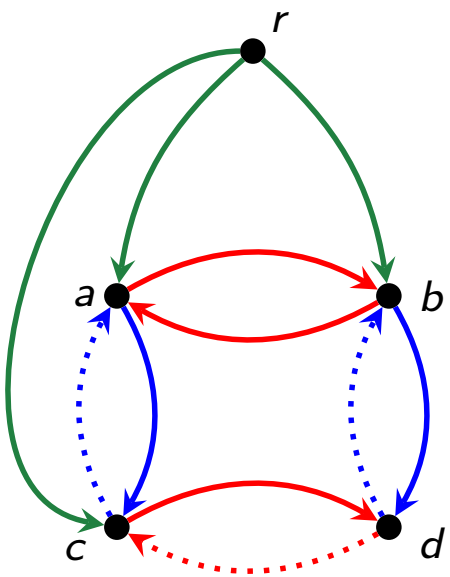
second choice

third choice



- Now $\ell(e) = 0$ for all red edges e and
- $\ell(e) = 1$ for all blue edges e and
- $\ell(e) = 2$ for all green edges e.
- G_ℓ has all blue/green edges + cd.
- Say $B = \{ra, db, ac, cd\}$.
- The edges ab and ba are violating.
- So $\ell(ab) = \ell(ba) = 1$.

Another example



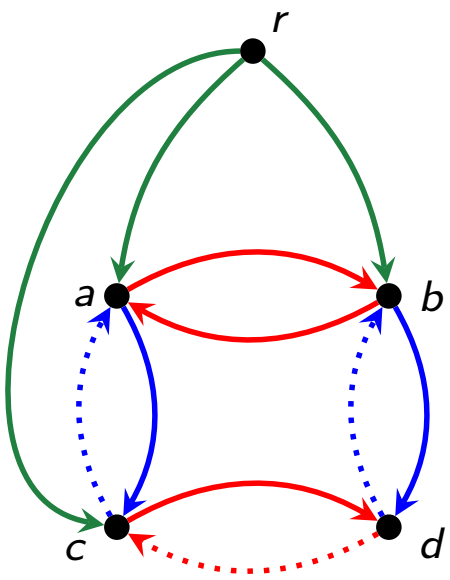
first choice

second choice

third choice

- G_ℓ has all edges except ca, db, dc.

Another example

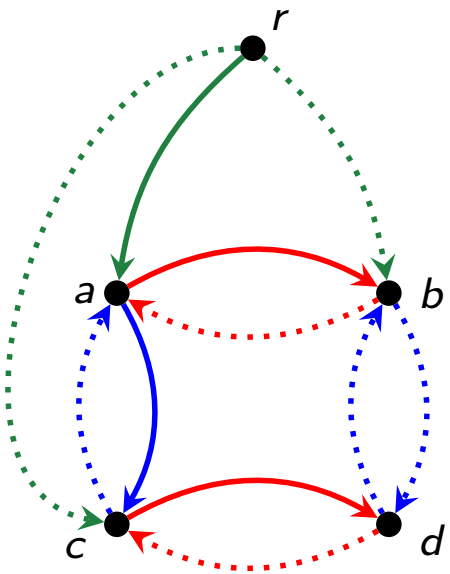


first choice

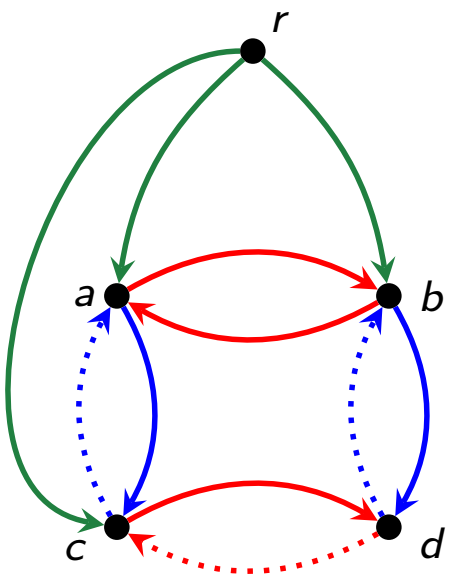
second choice

third choice

- G_ℓ has all edges except ca, db, dc.
- Say $B = \{ra, ab, ac, cd\}$.
- undir(B) is a MST in undir(G).
- So B is returned.



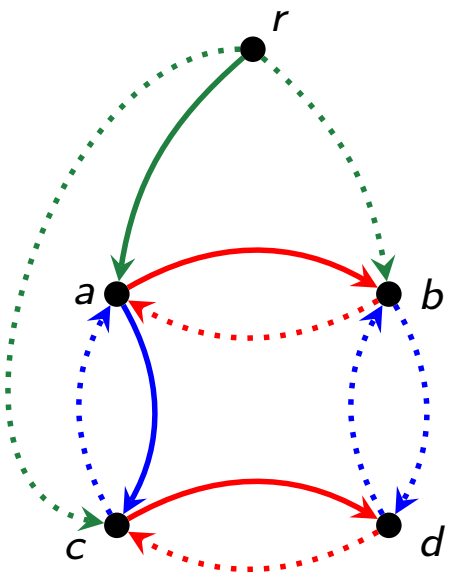
Another example



first choice

second choice

third choice



- G_ℓ has all edges except ca, db, dc.
- Say $B = \{ra, ab, ac, cd\}$.
- undir(B) is a MST in undir(G).
- So B is returned.
- Another possible output:

  $$B' = \{rb, ba, ac, cd\}.$$

- Both B and B' are popular arborescences in G.

The LP-method

Given an arborescence A, define edge weights in G as follows.

- For any $v \in V$, let $A(v)$ be the unique edge in $A \cap \delta^-(v)$; for any $e \in \delta^-(v)$:

$$\mathrm{wt}_A(e) = \begin{cases} 1 & \text{if } v \text{ prefers } e \text{ to } A(v); \\ -1 & \text{if } v \text{ prefers } A(v) \text{ to } e; \\ 0 & \text{otherwise.} \end{cases}$$

The LP-method

Given an arborescence A, define edge weights in G as follows.

- For any $v \in V$, let $A(v)$ be the unique edge in $A \cap \delta^-(v)$; for any $e \in \delta^-(v)$:

$$\mathrm{wt}_A(e) = \begin{cases} 1 & \text{if } v \text{ prefers } e \text{ to } A(v); \\ -1 & \text{if } v \text{ prefers } A(v) \text{ to } e; \\ 0 & \text{otherwise.} \end{cases}$$

OBSERVATION. For any arborescence A':

$$\mathrm{wt}_A(A') = \sum_{e \in A'} \mathrm{wt}_A(e)$$

$$= \#\text{ of vertices that prefer } A' - \#\text{ of vertices that prefer } A.$$

The LP-method

Given an arborescence A, define edge weights in G as follows.

- For any $v \in V$, let $A(v)$ be the unique edge in $A \cap \delta^-(v)$; for any $e \in \delta^-(v)$:

$$\mathrm{wt}_A(e) = \begin{cases} 1 & \text{if } v \text{ prefers } e \text{ to } A(v); \\ -1 & \text{if } v \text{ prefers } A(v) \text{ to } e; \\ 0 & \text{otherwise.} \end{cases}$$

OBSERVATION. For any arborescence A':

$$\mathrm{wt}_A(A') = \sum_{e \in A'} \mathrm{wt}_A(e)$$

$$= \#\text{ of vertices that prefer } A' - \#\text{ of vertices that prefer } A.$$

A is popular $\iff$ $\mathrm{wt}_A(A') \leq 0$ for all arborescences A' in G.

Since $\mathrm{wt}_A(A) = 0$:

- A is popular $\iff$ A is a max-weight arborescence under wt_A.

The LP-method

LP for max-weight arborescence:

$$\max \sum_{e \in E} \mathrm{wt}_A(e) \cdot x_e$$

$$\begin{aligned}
\sum_{e \in \delta^-(v)} x_e &= 1 \quad \forall\, v \in V \\
\sum_{e \in S} x_e &\leq \mathrm{rank}(S) \quad \forall\, S \subseteq E \\
x_e &\geq 0 \quad \forall e \in E.
\end{aligned}$$

For any $S \subseteq E$:

- $\mathrm{rank}(S)$ = size of a largest acyclic subset of S in the underlying undirected graph.

The LP-method

LP for max-weight arborescence:

$$\max \sum_{e \in E} \mathrm{wt}_A(e) \cdot x_e$$

$$\begin{aligned}
\sum_{e \in \delta^-(v)} x_e &= 1 \quad \forall\, v \in V \\
\sum_{e \in S} x_e &\leq \mathrm{rank}(S) \quad \forall\, S \subseteq E \\
x_e &\geq 0 \quad \forall e \in E.
\end{aligned}$$

For any $S \subseteq E$:

- $\mathrm{rank}(S)$ = size of a largest acyclic subset of S in the underlying undirected graph.

The feasible region of this LP is the arborescence polytope of G.

- Arborescence A is popular $\iff$ optimal value of this LP is 0.

The LP-method

The dual LP:

$$\min \sum_{v \in V} y_v \; + \sum_{S \subseteq E} \text{rank}(S) \cdot z_S$$

$$\begin{aligned} y_v + \sum_{S : e \in S} z_S &\geq \text{wt}_A(e) \quad \forall\, e \in \delta^-(v), \forall v \in V \\ z_S &\geq 0 \quad \forall\, S \subseteq E. \end{aligned}$$

The dual variables are y_v where $v \in V$ and z_S where $S \subseteq E$.

- Arborescence A is popular $\iff$ optimal value of the dual LP is 0.

The LP-method

The dual LP:

$$\min \sum_{v \in V} y_v \; + \sum_{S \subseteq E} \text{rank}(S) \cdot z_S$$

$$\begin{aligned} y_v + \sum_{S : e \in S} z_S &\geq \text{wt}_A(e) \quad \forall\, e \in \delta^-(v), \forall v \in V \\ z_S &\geq 0 \quad \forall\, S \subseteq E. \end{aligned}$$

The dual variables are y_v where $v \in V$ and z_S where $S \subseteq E$.

- Arborescence A is popular $\iff$ optimal value of the dual LP is 0.

The above LP has a special optimal solution: the support of $\vec{z}$ is a *nice* chain $\mathcal{S}$.

- $\mathcal{S} = \{S_0, S_1, \ldots, S_t\}$ where $S_0 \subset S_1 \subset \cdots \subset S_t = E$.
- $|A \cap S_i| = \text{rank}(S_i)$ for all $i \in \{0, \ldots, t\}$.

Correctness of our algorithm

The popular arborescence algorithm is searching for a nice chain.

If a branching B is returned:

- there is a nice chain $\mathcal{L} = \{L_0, L_1, \ldots, L_k\}$ where $L_i = \{e : \ell(e) \leq i\}$;
  - this implies B is a popular arborescence in G.

Correctness of our algorithm

The popular arborescence algorithm is searching for a nice chain.

If a branching B is returned:

- there is a nice chain $\mathcal{L} = \{L_0, L_1, \ldots, L_k\}$ where $L_i = \{e : \ell(e) \leq i\}$;
  - this implies B is a popular arborescence in G.

When our algorithm says "no":

- we show there is no nice chain;
- hence G has no popular arborescence.

Correctness of our algorithm

The popular arborescence algorithm is searching for a nice chain.

If a branching B is returned:

- there is a nice chain $\mathcal{L} = \{L_0, L_1, \ldots, L_k\}$ where $L_i = \{e : \ell(e) \leq i\}$;
  - this implies B is a popular arborescence in G.

When our algorithm says "no":

- we show there is no nice chain;
- hence G has no popular arborescence.

Thank you!