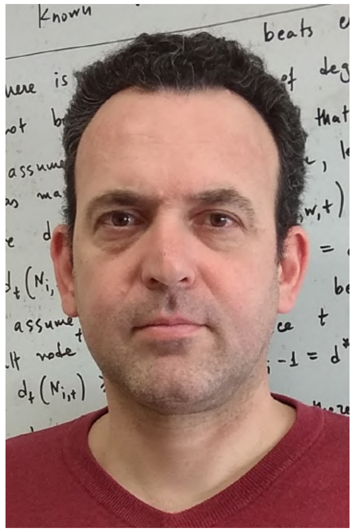
Optimizing over Serial Dictatorships



Nidhi Rathi
Max-Planck-Insitut für Informatik

Ioannis Caragiannis
Aarhus University, Denmark

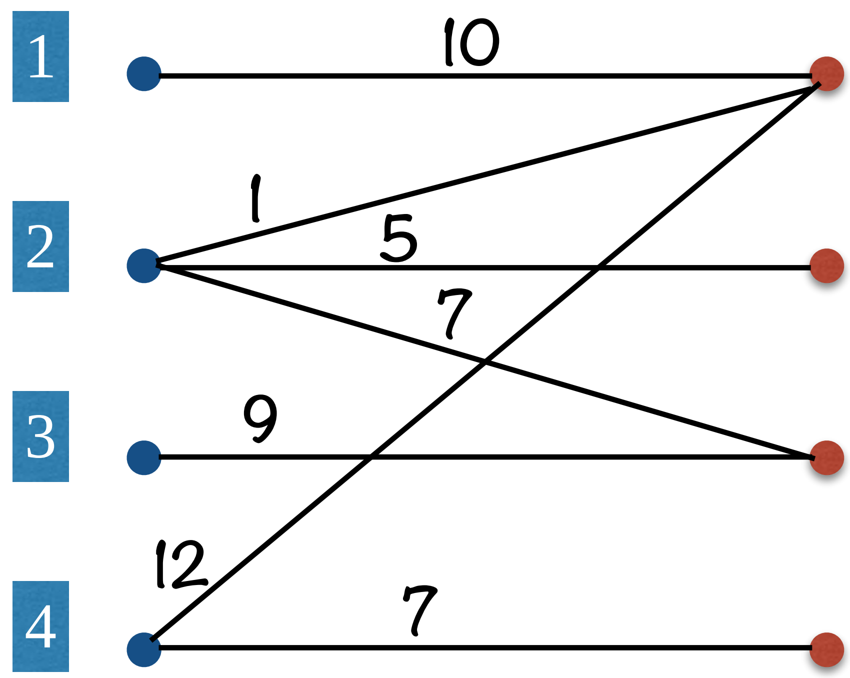
ADFOCS'23 @

max planck institut
informatik

Motivation

<u>SERIAL DICTATORSHIP:</u> an *action sequence* $(\sigma_1, \sigma_2, \ldots, \sigma_n)$ of *n* agents, where each agent picks the best available option at her turn

Motivation



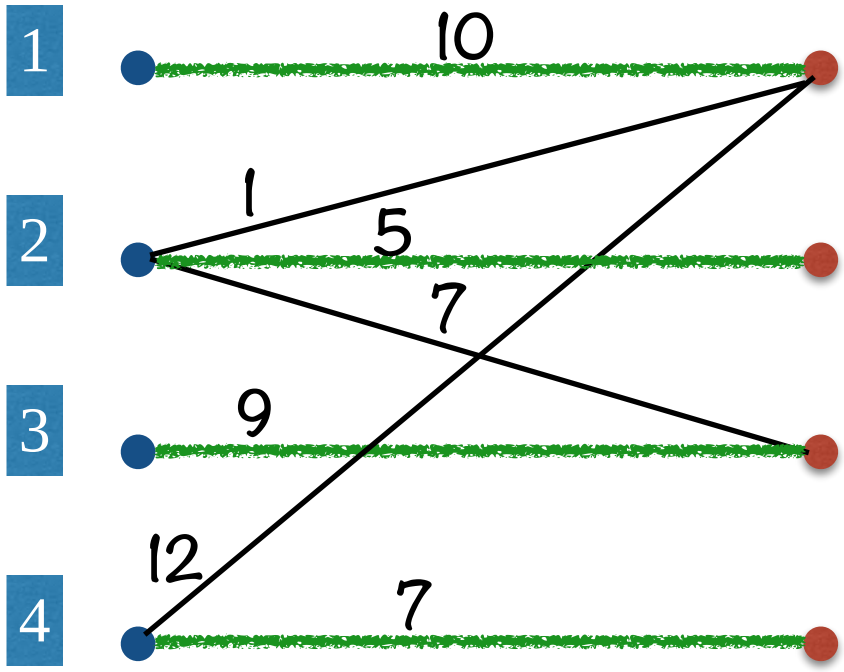
remaining edge weights = 0

Complete weighted bipartite graph

Goal: ***Maximum-weight matching***

SERIAL DICTATORSHIP: an *action sequence* $(\sigma_1, \sigma_2, \ldots, \sigma_n)$ of *n* agents,
where each agent picks the best available option at her turn

Motivation



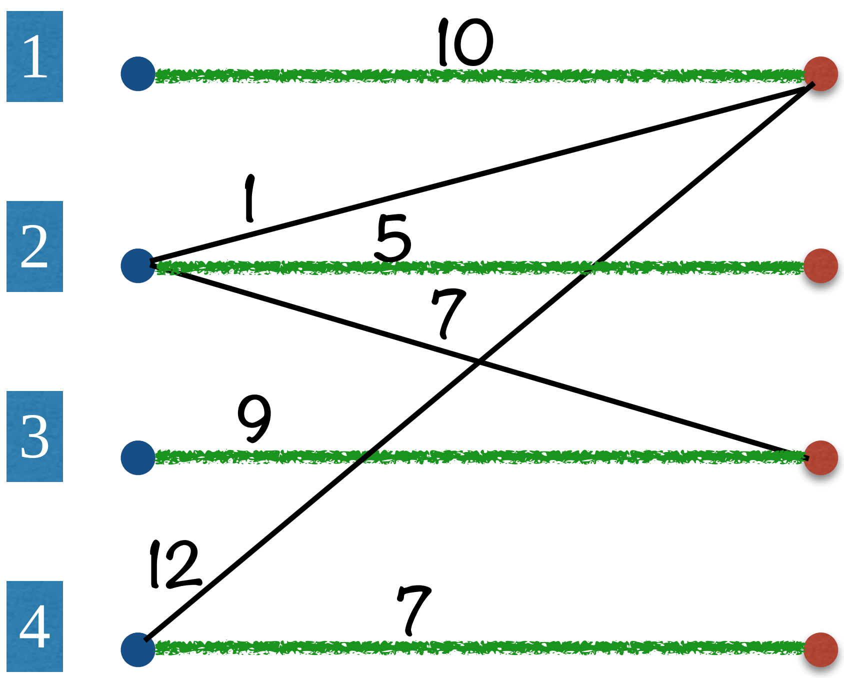
remaining edge weights = 0

<u>Complete weighted bipartite graph</u>

Goal: ***Maximum-weight matching***

<u>SERIAL DICTATORSHIP:</u> an *action sequence* $(\sigma_1, \sigma_2, \ldots, \sigma_n)$ of *n* agents,
where each agent picks the best available option at her turn

Motivation

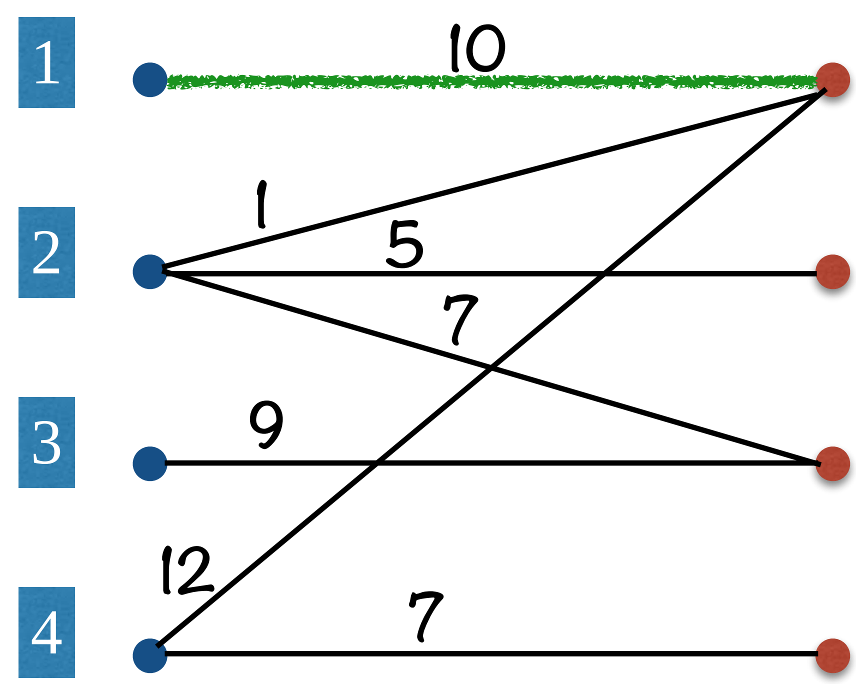


remaining edge weights = 0

Action sequence: 1 4 3 2 produces the maximum-weight matching

Serial Dictatorship: an *action sequence* $(\sigma_1, \sigma_2, \ldots, \sigma_n)$ of *n* agents, where each agent picks the best available option at her turn

Motivation

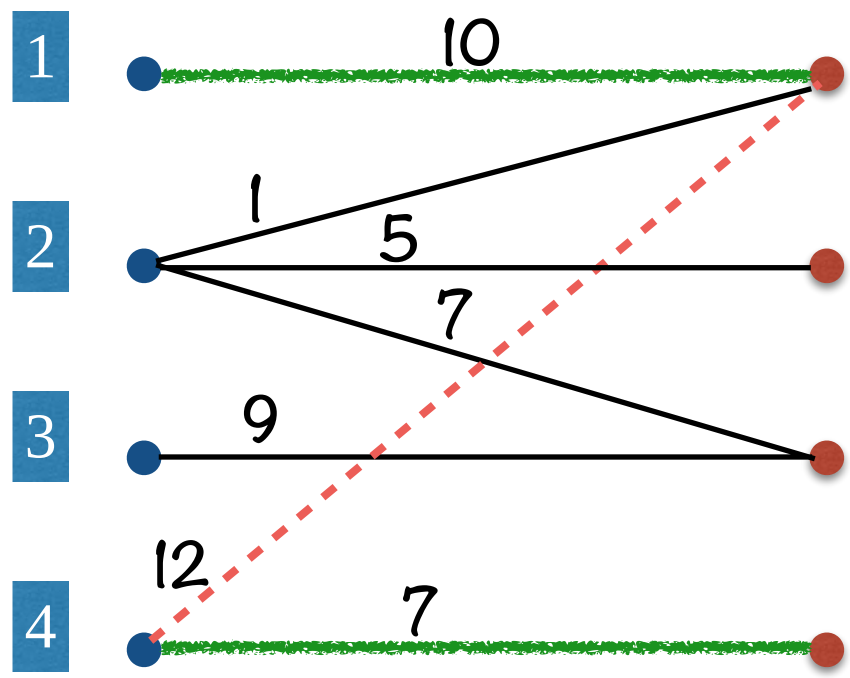


remaining edge weights = 0

Action sequence: 1 4 3 2 produces the maximum-weight matching

<u>Serial Dictatorship:</u> an *action sequence* $(\sigma_1, \sigma_2, \ldots, \sigma_n)$ of *n* agents, where each agent picks the best available option at her turn

Motivation

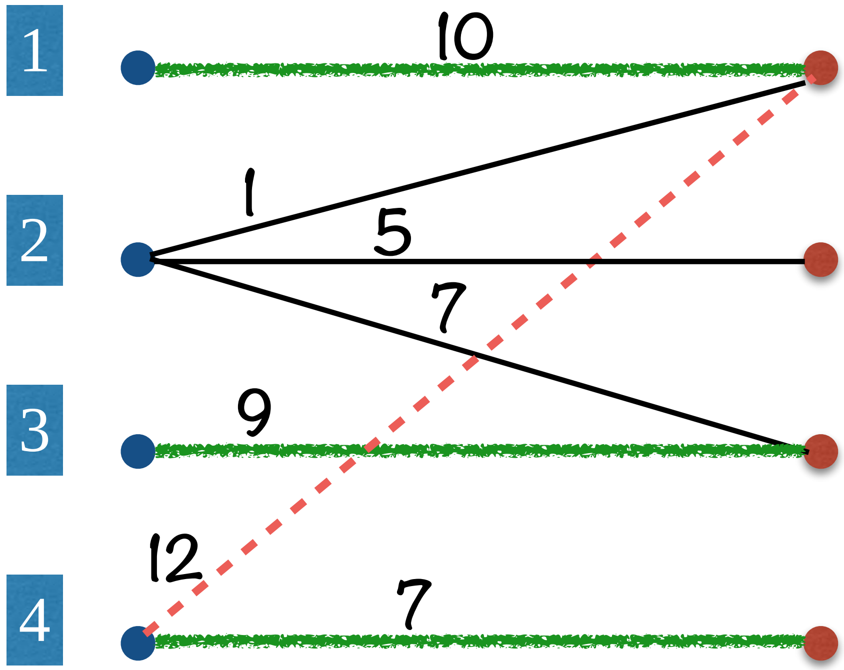


remaining edge weights = 0

Action sequence: 1 4 3 2 produces the maximum-weight matching

SERIAL DICTATORSHIP: an *action sequence* $(\sigma_1, \sigma_2, \ldots, \sigma_n)$ of *n* agents, where each agent picks the best available option at her turn

Motivation

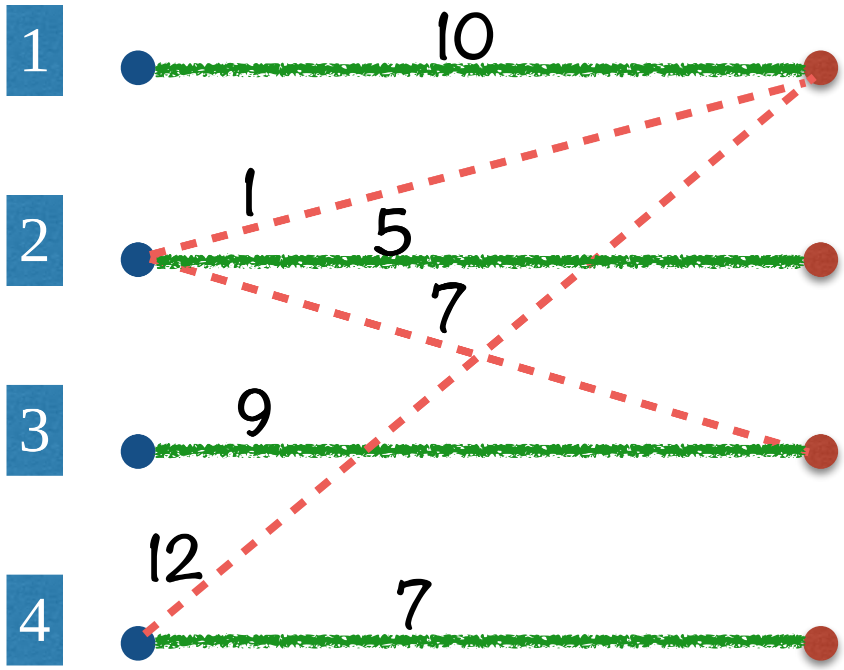


remaining edge weights = 0

Action sequence: 1 4 3 2 produces the maximum-weight matching

SERIAL DICTATORSHIP: an *action sequence* $(\sigma_1, \sigma_2, \ldots, \sigma_n)$ of *n* agents, where each agent picks the best available option at her turn

Motivation

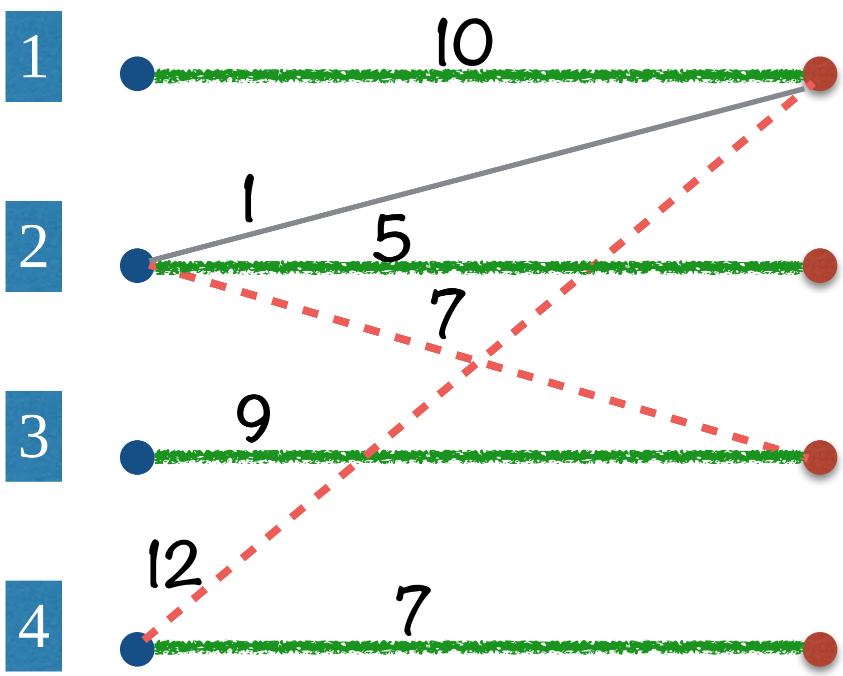


remaining edge weights = 0

Action sequence: 1 4 3 2 produces the maximum-weight matching

<u>SERIAL DICTATORSHIP:</u> an *action sequence* $(\sigma_1, \sigma_2, \ldots, \sigma_n)$ of *n* agents,
where each agent picks the best available option at her turn

Motivation



remaining edge weights = 0

Theorem: Any *max-weight matching* in a complete weighted bipartite graph, can *always* be induced by an *action sequence* of *n* agents.

SERIAL DICTATORSHIP: an *action sequence* $(\sigma_1, \sigma_2, \ldots, \sigma_n)$ of *n* agents,
where each agent picks the best available option at her turn

General Query model

- A set $\{1, 2, \ldots, n\}$ of *n* entities
- *Monotone* valuation functions, $v_i : \mathcal{S} \to \mathbb{R}_+$ for all $i \in [n]$
  ($\mathcal{S}$ is the set of all *ordered* subsets of $[n] \setminus \{i\}$)

General Query model

- A set $\{1, 2, \ldots, n\}$ of *n* entities
- *Monotone* valuation functions, $v_i : \mathcal{S} \to \mathbb{R}_+$ for all $i \in [n]$
  ($\mathcal{S}$ is the set of all *ordered* subsets of $[n] \setminus \{i\}$)

Value Queries: $v_i(S)$ = value of agent *i* when she gets to pick after agents in the ordered set $S \in \mathcal{S}$ have come

General Query model

- A set $\{1, 2, \ldots, n\}$ of *n* entities
- *Monotone* valuation functions, $v_i : \mathcal{S} \rightarrow \mathbb{R}_+$ for all $i \in [n]$
  ($\mathcal{S}$ is the set of all *ordered* subsets of $[n] \setminus \{i\}$)

Value Queries: $v_i(S)$ = value of agent *i* when she gets to pick after agents in the ordered set $S \in \mathcal{S}$ have come

Monotonicity: $v_i(S') \geq v_i(S)$ for all ordered subsets $S' of S$

Eg: $v_2(\phi) \geq v_2(61) \geq v_2(641)$

General Query model

- A set $\{1, 2, \ldots, n\}$ of *n* entities
- *Monotone* valuation functions, $v_i : \mathcal{S} \rightarrow \mathbb{R}_+$ for all $i \in [n]$
  ($\mathcal{S}$ is the set of all *ordered* subsets of $[n] \setminus \{i\}$)

Value Queries: $v_i(S)$ = value of agent *i* when she gets to pick after agents in the ordered set $S \in \mathcal{S}$ have come

Monotonicity: $v_i(S') \geq v_i(S)$ for all ordered subsets S' of S

Eg: $v_2(\phi) \geq v_2(61) \geq v_2(641)$

Goal: Understand the query complexity (# value queries required) of finding an action sequence σ that optimizes $\sum_{i \in [n]} v_i(\sigma^i)$, where σ^i : prefix of i in σ

For $\sigma = (1432)$, the sum is $v_1(\phi) + v_4(1) + v_3(14) + v_2(143)$

General Query model

- *Monotone* valuation functions, $v_i : \mathcal{S} \to \mathbb{R}_+$ for all $i \in [n]$
- Access via *value queries* of the form $v_i(S)$

Theorem:

For instances with *binary* valuations and a given parameter $\varepsilon > 0$

Goal: Understand the query complexity (# value queries required) of finding an action sequence σ that optimizes $\sum_{i \in [n]} v_i(\sigma^i)$, where σ^i : prefix of i in σ

General Query model

- *Monotone* valuation functions, $v_i : \mathcal{S} \to \mathbb{R}_+$ for all $i \in [n]$
- Access via *value queries* of the form $v_i(S)$

Theorem:

For instances with *binary* valuations and a given parameter $\varepsilon > 0$

- any *deterministic* algorithm that makes at most $n^{1/\varepsilon}$ value queries has an *approximation ratio* of at least $n\varepsilon$.

Goal: Understand the query complexity (# value queries required) of finding an action sequence σ that optimizes $\sum_{i \in [n]} v_i(\sigma^i)$, where σ^i : prefix of i in σ

General Query model

- *Monotone* valuation functions, $v_i : \mathcal{S} \rightarrow \mathbb{R}_+$ for all $i \in [n]$
- Access via *value queries* of the form $v_i(S)$

Theorem:

For instances with *binary* valuations and a given parameter $\varepsilon > 0$

- any *deterministic* algorithm that makes at most $n^{1/\varepsilon}$ value queries has an *approximation ratio* of at least $n\varepsilon$.

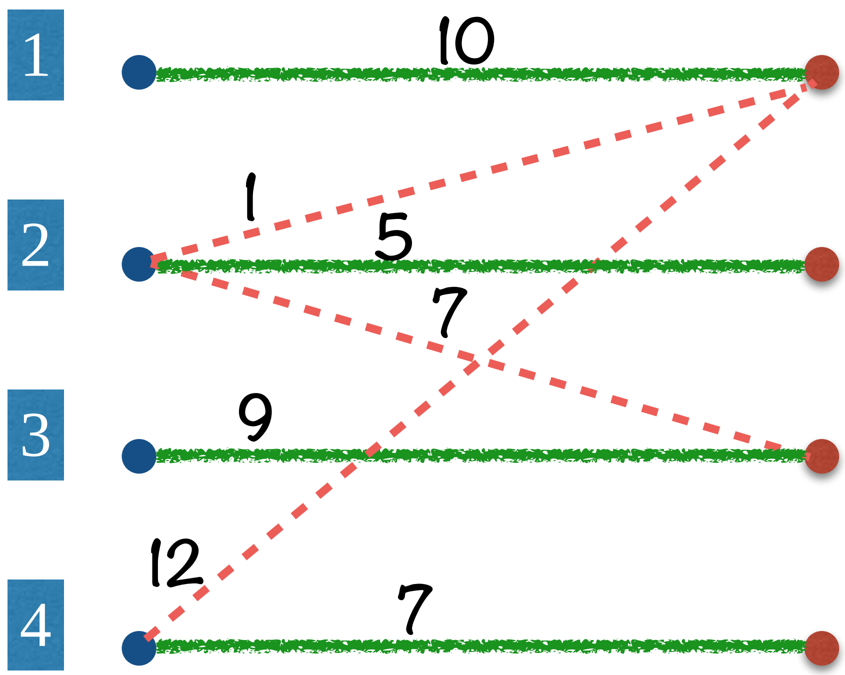
- any *randomized* algorithm that makes at most $\mathcal{O}(poly(n))$ value queries has an *approximation ratio* of at least $n^{\left(\frac{\log\log n}{\log n}\right)}$.

Goal: Understand the query complexity (# value queries required) of finding an action sequence σ that optimizes $\sum_{i\in[n]} v_i(\sigma^i)$, where σ^i : prefix of i in σ

Specific Problems

Maximum weight matching:

$v_i(S) =$ value of maximum-valued item available for i, after agents in S have picked their items.



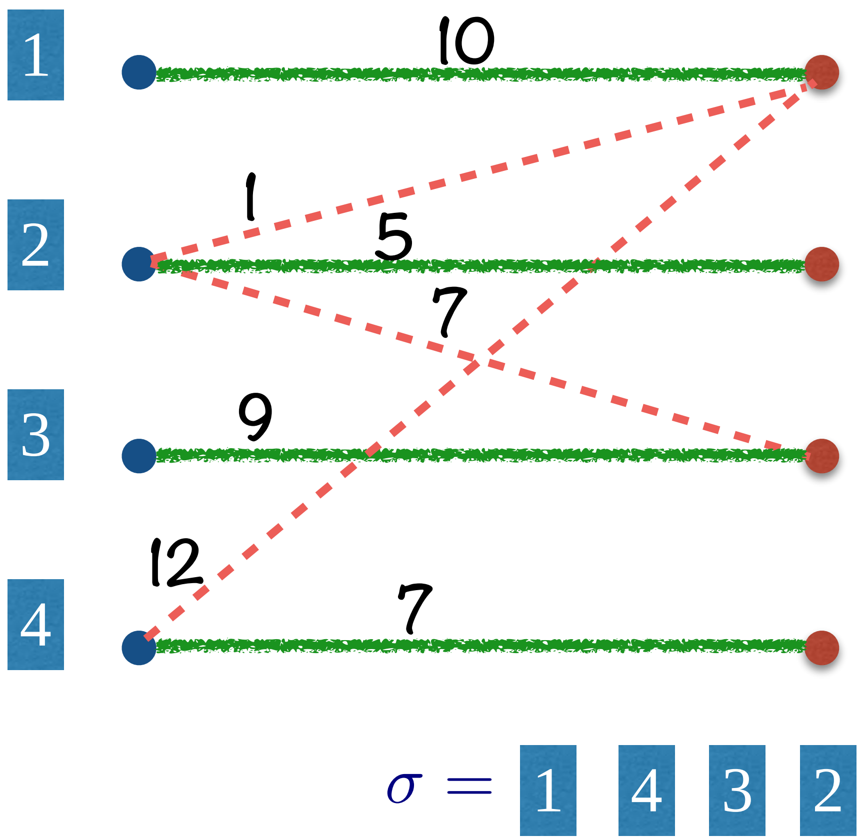
$\sigma =$ 1 4 3 2

Goal: Find an action sequence σ that maximizes the social welfare, $SW(\sigma) = \sum_{i \in [n]} v_i(\sigma^i)$ and understand its relation with the overall maximum social welfare.

Specific Problems

Maximum weight matching:

$v_i(S) =$ value of maximum-valued item available for *i*, after agents in S have picked their items.



Our results:

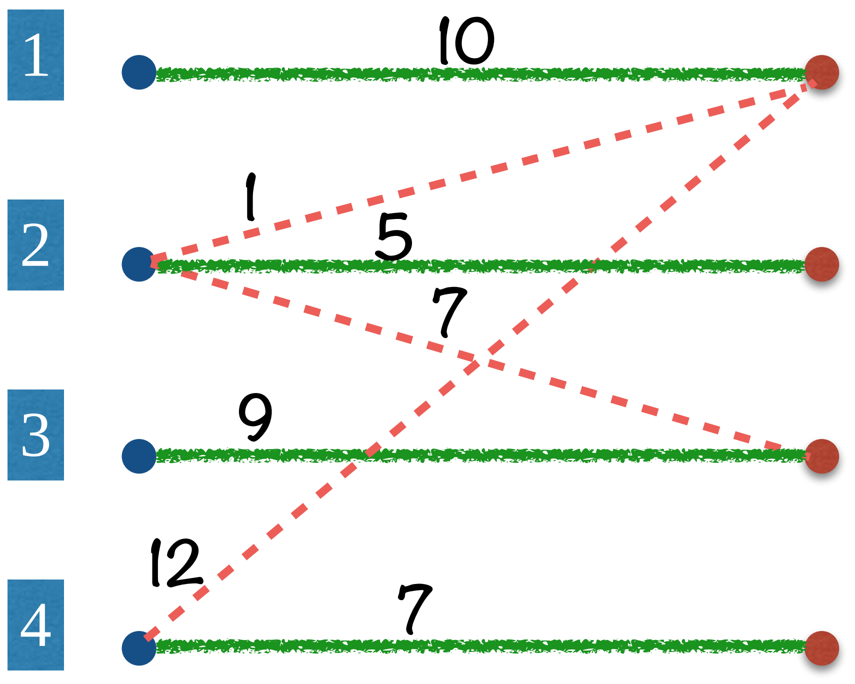
- Any max-weight matching **has** a corresponding *action sequence* of *n* agents that induces it.

Goal: Find an action sequence σ that maximizes the social welfare, $SW(\sigma) = \sum_{i \in [n]} v_i(\sigma^i)$ and understand its relation with the overall maximum social welfare.

Specific Problems

Maximum weight matching:

$v_i(S) =$ value of maximum-valued item available for *i*, after agents in S have picked their items.



$\sigma =$ 1 4 3 2

Our results:

- Any max-weight matching **has** a corresponding *action sequence* of *n* agents that induces it.
- 2-approximation polynomial-time algorithm.
  Can we do better?

Goal: Find an action sequence σ that maximizes the social welfare, $SW(\sigma) = \sum_{i \in [n]} v_i(\sigma^i)$ and understand its relation with the overall maximum social welfare.

Specific Problems

Maximum Satisfiability (weighted version):

$v_i(S) =$ Maximum weight of ***new*** clauses satisfied by variable x_i
after the variables in ordered set S have been set as T or F.

Goal: Find an action sequence σ that maximizes the social welfare, $SW(\sigma) = \sum_{i \in [n]} v_i(\sigma^i)$
and understand its relation with the overall maximum social welfare.

Specific Problems

Maximum Satisfiability (weighted version):

$v_i(S) =$ Maximum weight of ***new*** clauses satisfied by variable x_i
after the variables in ordered set S have been set as T or F.

Our results:

- An optimal assignment for MAX-SAT may ***not*** be produced from *any* action sequence of *n* variables!

Specific Problems

Maximum Satisfiability (weighted version):

$v_i(S) =$ Maximum weight of ***new*** clauses satisfied by variable x_i
after the variables in ordered set S have been set as T or F.

Our results:

- An optimal assignment for MAX-SAT may ***not*** be produced from *any* action sequence of *n* variables!

Conjecture: For any instance of MAX-SAT, there exists an action sequence that achieves **2/3** of the optimal value.
(**2**-approximation is doable)

Specific Problems

Maximum Satisfiability (weighted version):

$v_i(S) =$ Maximum weight of ***new*** clauses satisfied by variable x_i
after the variables in ordered set S have been set as T or F.

Our results:

- An optimal assignment for MAX-SAT may ***not*** be produced from *any* action sequence of *n* variables!
- *Given an instance of MAX-SAT, does there exist an action sequence for all 1's assignment?* NP-complete

Conjecture: For any instance of MAX-SAT, there exists an action sequence that achieves **2/3** of the optimal value.
(**2**-approximation is doable)

The Big Picture

- Introduce a *query model for understanding serial dictatorship* in the abstract setting.
- *Upper and Lower bounds* for the query complexity of optimizing serial dictatorship (the action sequence that maximizes the social welfare)

The Big Picture

- Introduce a *query model for understanding serial dictatorship* in the abstract setting.
- *Upper and Lower bounds* for the query complexity of optimizing serial dictatorship (the action sequence that maximizes the social welfare)
- *Revisit* some of the celebrated problems in theoretical computer science and inspect the connection between their optimal solutions and *serial dictatorships*.

✓ Maximum-weight Matching in bipartite graph
✗ Maximum-weight Matching in non-bipartite graph
✗ Maximum Satisfiability (weighted version)
✗ Longest path with maximum-weight
✓ Maximum-weight Arborescence
Maximum-weight Cut 

The Big Picture

- Introduce a *query model for understanding serial dictatorship* in the abstract setting.
- *Upper and Lower bounds* for the query complexity of optimizing serial dictatorship (the action sequence that maximizes the social welfare)
- *Revisit* some of the celebrated problems in theoretical computer science and inspect the connection between their optimal solutions and *serial dictatorships*.

✓ Maximum-weight Matching in bipartite graph
✗ Maximum-weight Matching in non-bipartite graph
✗ Maximum Satisfiability (weighted version)
✗ Longest path with maximum-weight
✓ Maximum-weight Arborescence
Maximum-weight Cut 

Thank you!

Specific Problems

Longest path with maximum weight:

$v_i(S) =$ Maximum weight that node i can achieve such that the underlying structure is a union of paths

Our results:

- An optimal assignment for Longest-Path may not be produced from any action sequence of n nodes.
- For any instance of Longest-Path, there always exists an action sequence that recovers **1/2** of the optimal value.

Conjecture: The above factor is 2/3.

Goal: Find an action sequence σ that maximizes the social welfare, $SW(\sigma) = \sum_{i \in [n]} v_i(\sigma^i)$ and understand its relation with the overall maximum social welfare.