A General Framework For Computing the Nucleolus Via Dynamic Programming

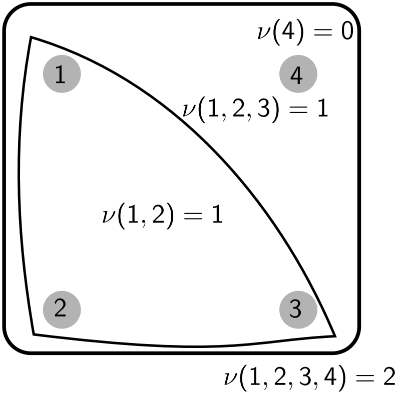
Justin Toth

Joint work with Jochen Könemann

UNIVERSITY OF WATERLOO | FACULTY OF MATHEMATICS
Department of Combinatorics and Optimization

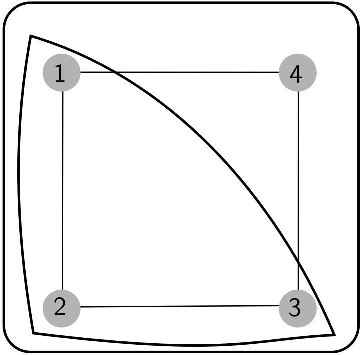
Cooperative Game Theory

- n players with value function $\nu : 2^{[n]} \to \mathbb{R}$



Cooperative Game Theory

- n players with value function $\nu : 2^{[n]} \to \mathbb{R}$
- Comb Opt Games: ν optimizes a combinatorial structure



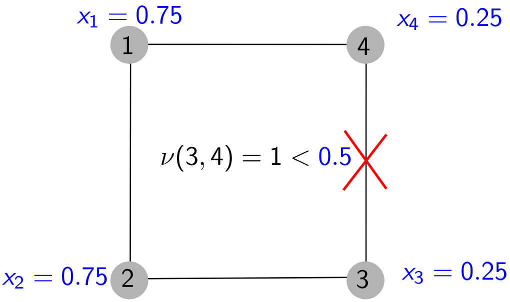
$\nu(1, 2) = 1$

$\nu(1, 2, 3, 4) = 2$

$\nu(4) = 0$

$\nu(1, 2, 3) = 1$

- n players with value function $\nu : 2^{[n]} \to \mathbb{R}$
- Comb Opt Games: ν optimizes a combinatorial structure
- Solution concepts assign payoffs to players



$\nu(1,2) = 1$

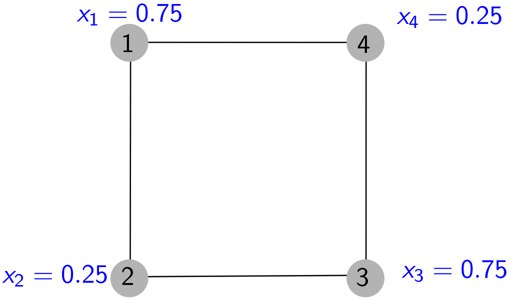
$\nu(1,2,3,4) = 2$

$\nu(4) = 0$

$\nu(1,2,3) = 1$

Cooperative Game Theory

- n players with value function $\nu : 2^{[n]} \to \mathbb{R}$
- Comb Opt Games: ν optimizes a combinatorial structure
- Solution concepts assign payoffs to players
- Core solutions disincentize deviation from grand coaliton $[n]$.

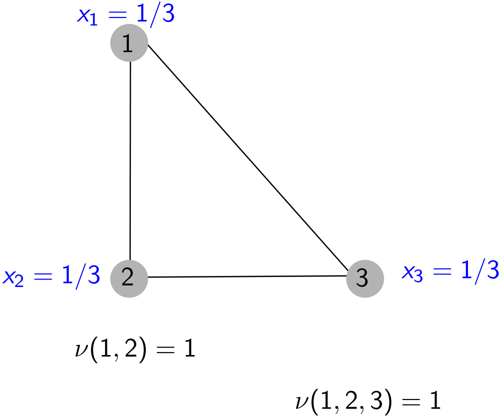


$\nu(1,2) = 1$ $\nu(1,2,3,4) = 2$

$\nu(4) = 0$ $\nu(1,2,3) = 1$

Cooperative Game Theory

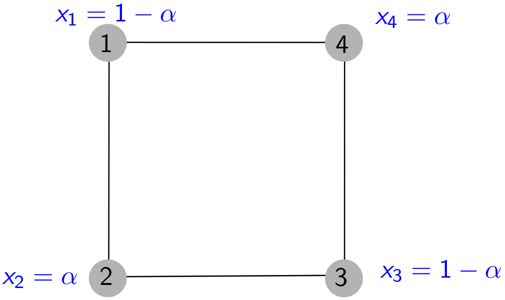
- Core can be empty
- Leastcore: max min excess: $x(S) - \nu(S)$.



Cooperative Game Theory

- Core can be empty
- Leastcore: max min excess: $x(S) - \nu(S)$.
- Leastcore can be non-unique

$\alpha \in [0, 1]$



$\nu(1, 2) = 1$

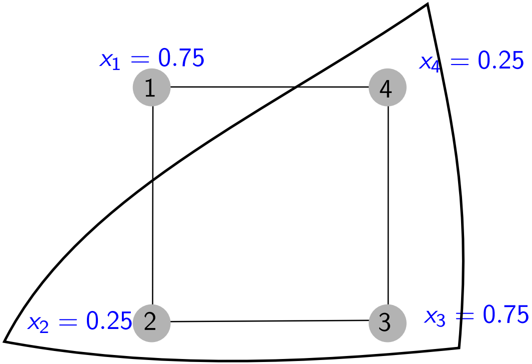
$\nu(1, 2, 3, 4) = 2$

$\nu(4) = 0$

$\nu(1, 2, 3) = 1$

Cooperative Game Theory

- Core can be empty
- Leastcore: max min excess: $x(S) - \nu(S)$.
- Leastcore can be non-unique
- $2nd$ min excess can be improved



$\nu(1,2) = 1$

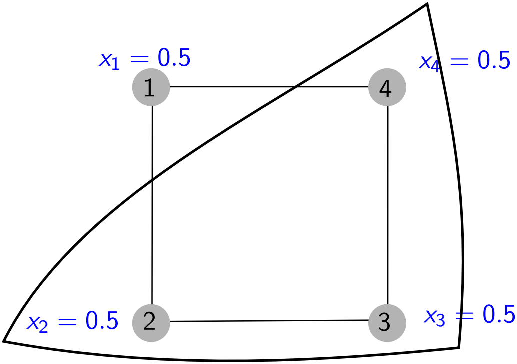
$\nu(1,2,3,4) = 2$

$\nu(4) = 0$

$\nu(1,2,3) = 1$

Cooperative Game Theory

- Core can be empty
- Leastcore: max min excess: $x(S) - \nu(S)$.
- Leastcore can be non-unique
- $2nd$ min excess can be improved
- Nucleolus: max min excess, 2nd min excess, . . . , lexicographically



$\nu(1,2) = 1$ $\nu(1,2,3,4) = 2$

$\nu(4) = 0$ $\nu(1,2,3) = 1$

Refining the Leastcore

- For allocation $x \in \mathbb{R}^n$, let

$$S_1, \ldots, S_{2^n-2}$$

be all proper coalitions in order of non-decreasing excess $\text{ex}(x, S) = x(S) - \nu(S)$:

Refining the Leastcore

- For allocation $x \in \mathbb{R}^n$, let

$$S_1, \ldots, S_{2^n-2}$$

be all proper coalitions in order of non-decreasing excess $\mathrm{ex}(x, S) = x(S) - \nu(S)$:

$$\mathrm{ex}(x, S_1) \leq \mathrm{ex}(x, S_2) \leq \ldots \leq \mathrm{ex}(x, S_{2^n-2})$$

Refining the Leastcore

- For allocation $x \in \mathbb{R}^n$, let

$$S_1, \ldots, S_{2^n-2}$$

  be all proper coalitions in order of non-decreasing excess $\text{ex}(x, S) = x(S) - \nu(S)$:

$$\text{ex}(x, S_1) \leq \text{ex}(x, S_2) \leq \ldots \leq \text{ex}(x, S_{2^n-2})$$

- We define

$$\Theta(x) = (\text{ex}(x, S_1), \ldots, \text{ex}(x, S_{2^n-2})).$$

Refining the Leastcore

- For allocation $x \in \mathbb{R}^n$, let

$$S_1, \ldots, S_{2^n-2}$$

be all proper coalitions in order of non-decreasing excess $\mathrm{ex}(x, S) = x(S) - \nu(S)$:

$$\mathrm{ex}(x, S_1) \leq \mathrm{ex}(x, S_2) \leq \ldots \leq \mathrm{ex}(x, S_{2^n-2})$$

- We define

$$\Theta(x) = (\mathrm{ex}(x, S_1), \ldots, \mathrm{ex}(x, S_{2^n-2})).$$

- Leastcore: allocations x, maximizing first coordinate of $\Theta(x)$

The Nucleolus

Definition: (Schmeidler 1969) The nucleolus of cooperative game (n, ν) is defined as

$$\eta(n, \nu) := \arg \operatorname{lex} \max_{x \in \mathbb{R}^n} \Theta(x)$$

The Nucleolus

Definition: (Schmeidler 1969) The nucleolus of cooperative game (n, ν) is defined as

$$\eta(n, \nu) := \text{arg lex max}_{x \in \mathbb{R}^n} \Theta(x)$$

- The nucleolus is unique and in a sense "the most stable" allocation

The Nucleolus

Definition: (Schmeidler 1969) The nucleolus of cooperative game (n, ν) is defined as

$$\eta(n, \nu) := \text{arg lex max}_{x \in \mathbb{R}^n} \Theta(x)$$

- The nucleolus is unique and in a sense "the most stable" allocation
- (Aumann and Maschler 1985) Nucleolus explains certain wealth division applications in the Babylonian talmud.

Computing The Nucleolus – (Non-Exhaustive) History

- (Solymosi and Raghavan 1994) Efficient algorithm for finding nucleolus in (weighted) bipartite matching games

Computing The Nucleolus – (Non-Exhaustive) History

- (Solymosi and Raghavan 1994) Efficient algorithm for finding nucleolus in (weighted) bipartite matching games
- (Kuipers 1996) Efficient algorithm for finding nucleolus in convex games

Computing The Nucleolus – (Non-Exhaustive) History

- (Solymosi and Raghavan 1994) Efficient algorithm for finding nucleolus in (weighted) bipartite matching games
- (Kuipers 1996) Efficient algorithm for finding nucleolus in convex games
- Finding the nucleolus of MCST games is NP-hard (Faigle et al. 1998).

Computing The Nucleolus – (Non-Exhaustive) History

- (Solymosi and Raghavan 1994) Efficient algorithm for finding nucleolus in (weighted) bipartite matching games
- (Kuipers 1996) Efficient algorithm for finding nucleolus in convex games
- Finding the nucleolus of MCST games is NP-hard (Faigle et al. 1998).
- (Kern, Paulusma 2003) Can efficiently compute nucleolus in unweighted matching games

Computing The Nucleolus – (Non-Exhaustive) History

- (Solymosi and Raghavan 1994) Efficient algorithm for finding nucleolus in (weighted) bipartite matching games
- (Kuipers 1996) Efficient algorithm for finding nucleolus in convex games
- Finding the nucleolus of MCST games is NP-hard (Faigle et al. 1998).
- (Kern, Paulusma 2003) Can efficiently compute nucleolus in unweighted matching games
- Computing the nucleolus in network flow games is NP-hard (Deng, Fang, Sun 2009)

Computing The Nucleolus – (Non-Exhaustive) History

- (Solymosi and Raghavan 1994) Efficient algorithm for finding nucleolus in (weighted) bipartite matching games
- (Kuipers 1996) Efficient algorithm for finding nucleolus in convex games
- Finding the nucleolus of MCST games is NP-hard (Faigle et al. 1998).
- (Kern, Paulusma 2003) Can efficiently compute nucleolus in unweighted matching games
- Computing the nucleolus in network flow games is NP-hard (Deng, Fang, Sun 2009)
- Computing the nucleolus in weighted voting games is NP-hard (Elkind et al. 2007), and pseudo-polynomial time algorithms exist (Elkind and Pasechnik 2008), (Pashkovich 2018)

Computing The Nucleolus – (Non-Exhaustive) History

- (Solymosi and Raghavan 1994) Efficient algorithm for finding nucleolus in (weighted) bipartite matching games
- (Kuipers 1996) Efficient algorithm for finding nucleolus in convex games
- Finding the nucleolus of MCST games is NP-hard (Faigle et al. 1998).
- (Kern, Paulusma 2003) Can efficiently compute nucleolus in unweighted matching games
- Computing the nucleolus in network flow games is NP-hard (Deng, Fang, Sun 2009)
- Computing the nucleolus in weighted voting games is NP-hard (Elkind et al. 2007), and pseudo-polynomial time algorithms exist (Elkind and Pasechnik 2008), (Pashkovich 2018)
- (Baiou and Barahona 2019) Efficient algorithm for shortest path games.

Computing The Nucleolus – (Non-Exhaustive) History

- (Solymosi and Raghavan 1994) Efficient algorithm for finding nucleolus in (weighted) bipartite matching games
- (Kuipers 1996) Efficient algorithm for finding nucleolus in convex games
- Finding the nucleolus of MCST games is NP-hard (Faigle et al. 1998).
- (Kern, Paulusma 2003) Can efficiently compute nucleolus in unweighted matching games
- Computing the nucleolus in network flow games is NP-hard (Deng, Fang, Sun 2009)
- Computing the nucleolus in weighted voting games is NP-hard (Elkind et al. 2007), and pseudo-polynomial time algorithms exist (Elkind and Pasechnik 2008), (Pashkovich 2018)
- (Baiou and Barahona 2019) Efficient algorithm for shortest path games.
- (Könemann, Pashkovich, T. 2019) Can compute the nucleolus of weighted matching games in polynomial time.

Our Contributions

Theorem 1: For a given cooperative game if the min excess problem: for any input $x \in \mathbb{R}^n$, find $S \subseteq [n]$ minimizing $x(S) - \nu(S)$, can be modelled with a dynamic program then the nucleolus of that game can be computed in time $O(n^6 T)$ where T is the time it takes to solve the dynamic program.

Theorem 2: The min excess problem for b-matching games can be modelled with a dynamic program which can be solved in polynomial time if the underlying graph has bounded treewidth.

Corollary: On graphs of bounded treewidth the nucleolus of b-matching games can be computed in polynomial time.

Congruency-Constrained Min Excess Problem

Implicit in (Pashkovich 2018) is the following:

Lemma: If for any prime $p = O(n^2)$, $v \in \mathbb{F}_p^n$, and $q \in \mathbb{F}_p$ one can solve

$$\min\{x(S) - \nu(S) : v(S) \equiv p \mod p\}$$

in time T then one can compute the nucleolus in time $O(\text{poly}(n)T)$.

Congruency-Constrained Min Excess Problem

Implicit in (Pashkovich 2018) is the following:

Lemma: If for any prime $p = O(n^2)$, $v \in \mathbb{F}_p^n$, and $q \in \mathbb{F}_p$ one can solve

$$\min\{x(S) - \nu(S) : v(S) \equiv p \mod p\}$$

in time T then one can compute the nucleolus in time $O(\text{poly}(n)T)$.

- Min excess problem: $\epsilon_1 = \min\{x(S) - \nu(S) : S \subseteq [n]\} \equiv$ separation oracle for leastcore points.

Congruency-Constrained Min Excess Problem

Implicit in (Pashkovich 2018) is the following:

Lemma: If for any prime $p = O(n^2)$, $v \in \mathbb{F}_p^n$, and $q \in \mathbb{F}_p$ one can solve

$$\min\{x(S) - \nu(S) : v(S) \equiv p \mod p\}$$

in time T then one can compute the nucleolus in time $O(\text{poly}(n)T)$.

- Min excess problem: $\epsilon_1 = \min\{x(S) - \nu(S) : S \subseteq [n]\} \equiv$ separation oracle for leastcore points.
- To compute nucleolus need to separate 2nd min excess, . . . , and so on.

$$\min\{x(S) - \nu(S) : x(S) - \nu(S) \neq \epsilon_1, S \subseteq [n]\}$$

Congruency-Constrained Min Excess Problem

Implicit in (Pashkovich 2018) is the following:

Lemma: If for any prime $p = O(n^2)$, $v \in \mathbb{F}_p^n$, and $q \in \mathbb{F}_p$ one can solve

$$\min\{x(S) - \nu(S) : v(S) \equiv p \mod p\}$$

in time T then one can compute the nucleolus in time $O(\text{poly}(n)T)$.

- Min excess problem: $\epsilon_1 = \min\{x(S) - \nu(S) : S \subseteq [n]\} \equiv$ separation oracle for leastcore points.
- To compute nucleolus need to separate 2nd min excess, . . . , and so on.

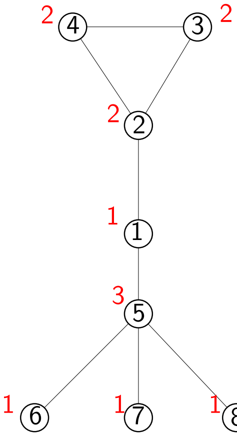
$$\min\{x(S) - \nu(S) : x(S) - \nu(S) \neq \epsilon_1, S \subseteq [n]\}$$

- Insight: $x(S) - \nu(S) \neq \epsilon_1$ can be replaced with poly(n) congruency tests of the form

$$\min\{x(S) - \nu(S) : v(S) \equiv q \mod p, S \subseteq [n]\}$$

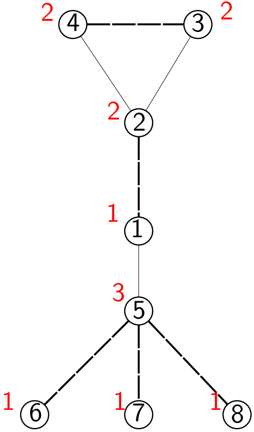
b-Matching Games on Graphs of Bounded Treewidth

- *b-matching games:* each player i can form b_i partnerships.



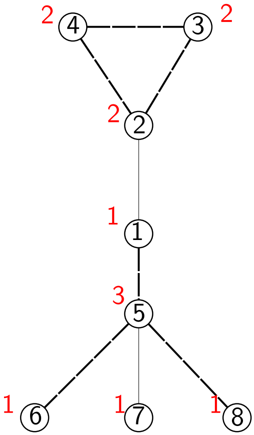
b-Matching Games on Graphs of Bounded Treewidth

- b-matching games: each player i can form b_i partnerships.



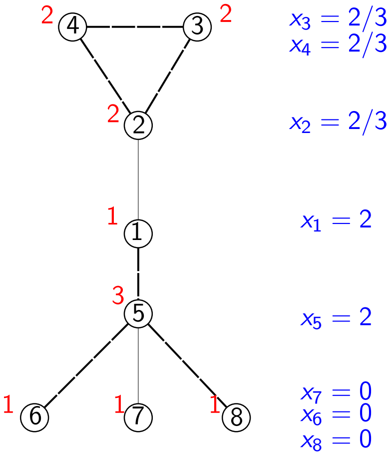
b-Matching Games on Graphs of Bounded Treewidth

- b-matching games: each player i can form b_i partnerships.



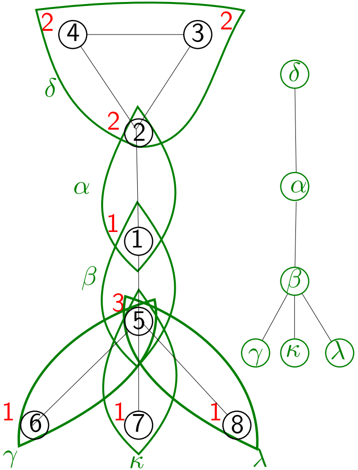
b-Matching Games on Graphs of Bounded Treewidth

- b-matching games: each player i can form b_i partnerships.
- min $x(S) - \nu(S)$ where $\nu :=$ max w-weight b-Matching on $G[S]$.



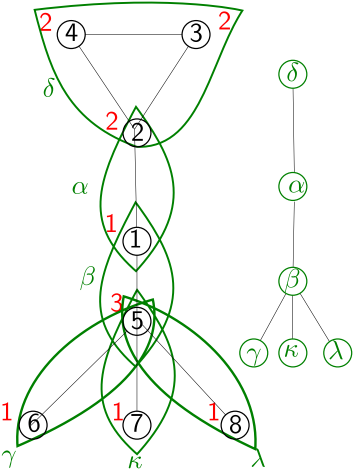
b-Matching Games on Graphs of Bounded Treewidth

- b-matching games: each player i can form b_i partnerships.
- min $x(S) - \nu(S)$ where $\nu :=$ max w-weight b-Matching on $G[S]$.
- tree decomposition: Cover edges with bags of vertices.
- bag intersections form a tree structure.
- Size of largest bag - 1 is treewidth



- $C[\zeta, F] :=$ optimal solution on bags rooted at ζ using edges F in bag ζ.
- $C[\alpha, \{12\}] \mathrel{\hat{=}}$

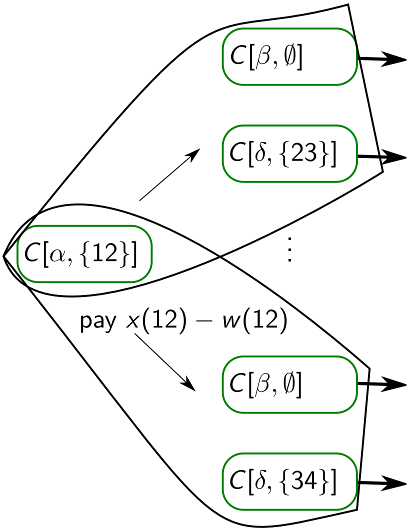
$$\min\ x(12) - w(12) + C[\beta, F^\beta] + C[\delta, F^\delta]$$
$$\text{s.t.}\ F^\beta \subseteq \{15\}$$
$$F^\delta \subseteq \{23, 34, 24\}$$
$$|\{e \in F^\beta : 1 \in e\}| \leq 0$$
$$|\{e \in F^\delta : 2 \in e\}| \leq 1$$



b-Matching Games on Graphs of Bounded Treewidth

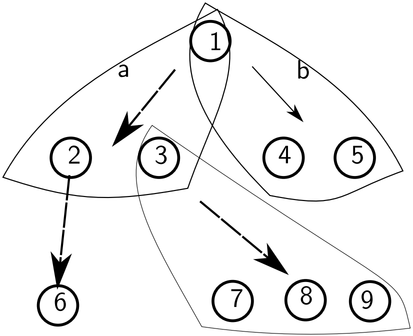
- $C[\zeta, F] :=$ optimal solution on bags rooted at ζ using edges F in bag ζ.
- $C[\alpha, \{12\}] \hat{=}$

$$\min\ x(12) - w(12) + C[\beta, F^\beta] + C[\delta, F^\delta]$$
$$\text{s.t.}\ F^\beta \subseteq \{15\}$$
$$F^\delta \subseteq \{23, 34, 24\}$$
$$|\{e \in F^\beta : 1 \in e\}| \leq 0$$
$$|\{e \in F^\delta : 2 \in e\}| \leq 1$$



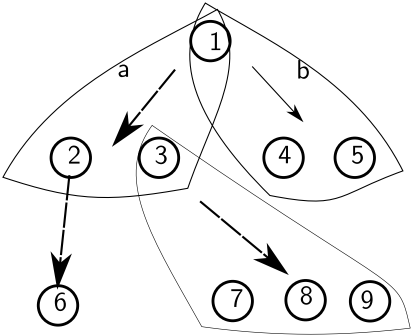
Hypergraph Model of Dynamic Programming

- Dynamic program solutions: hyperpaths from root of Directed Acylic Hypergraph to leaves.



Hypergraph Model of Dynamic Programming

- Dynamic program solutions: hyperpaths from root of Directed Acylic Hypergraph to leaves.
- (Campbell, Martin, Rardin 1990) Convex Hull of Hyperpaths has compact extended formulation.

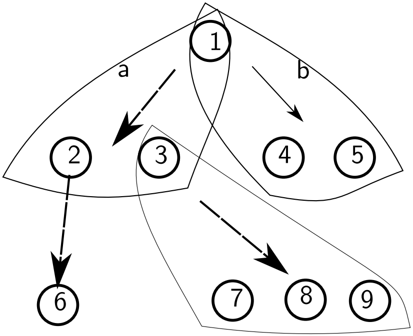


Hypergraph Model of Dynamic Programming

- Dynamic program solutions: hyperpaths from root of Directed Acylic Hypergraph to leaves.
- (Campbell, Martin, Rardin 1990) Convex Hull of Hyperpaths has compact extended formulation.
- How can we identify all hyperpaths P such that

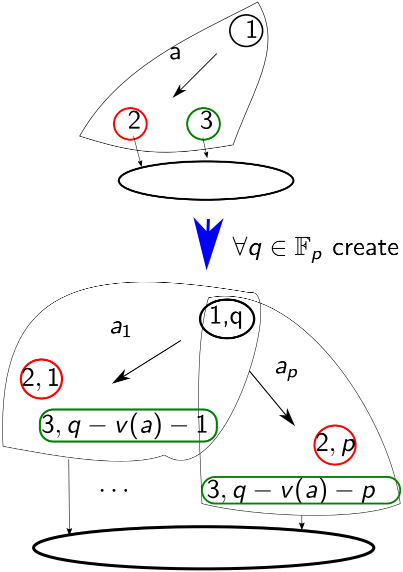
$$\sum_{a \in P} v(a) \equiv q \mod p$$

for some $v \in \mathbb{F}_p^n$, $q \in \mathbb{F}_p$, prime p?



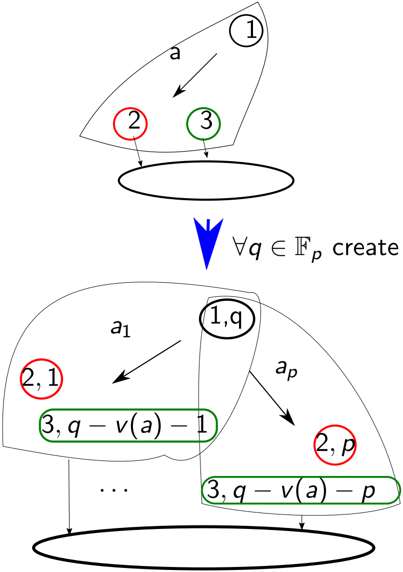
Congruency Constrained Dynamic Programming

- Let $v \in \mathbb{F}_p^n$.
- Construct a new congruency hypergraph.



Congruency Constrained Dynamic Programming

- Let $v \in \mathbb{F}_p^n$.
- Construct a new congruency hypergraph.
- For each $q \in \mathbb{F}_p$ and for each node u in hypergraph do:
- Create node (u, q) to track hyperpaths P rooted at u of congruency $v(P) \equiv q \mod p$.
- Heads of arc a_i should sum to $q - v(a) \mod p$.



Conclusion

Results

- When we can solve the min excess problem efficiently with a dynamic program we can also compute the nucleolus efficiently
- The min excess problem of b-matching games can be modelled efficiently with a dynamic program on graphs of bounded treewidth
- The nucleolus of b-matching games can be computed efficiently on graphs of bounded treewidth

Conclusion

Results

- When we can solve the min excess problem efficiently with a dynamic program we can also compute the nucleolus efficiently
- The min excess problem of b-matching games can be modelled efficiently with a dynamic program on graphs of bounded treewidth
- The nucleolus of b-matching games can be computed efficiently on graphs of bounded treewidth

Open Problems

- Can we apply this framework to compute the nucleolus of other interesting games?
- Branched Polyhedral Systems (Kaibel and Loos 2010) are a common generalization of dynamic programming and disjunctive programming. Can our techniques extend to that setting?
- What is the complexity of computing the nucleolus of b-matching games in general?