An Algorithm for the Coalitional Manipulation Problem under Maximin

Michael Zuckerman, Omer Lev and Jeffrey S. Rosenschein

(Simulations by Amitai Levy)

BISFAI, June 2011

Introduction

- Elections
  - Voters submit linear orders of the candidates
  - A *voting rule* determines the winner based on the votes
- Manipulation
  - A voter casts a vote that is not his true preference, to make himself better off
- Gibbard – Satterthwaite theorem
  - Every reasonable voting rule is manipulable

Unweighted Coalitional Optimization (UCO) problem

- Given
  - A voting rule r
  - The Profile of Non-Manipulators PNM
  - Candidate p preferred by the manipulators
- We are asked to find the minimum k such that there exists a set of manipulators M with $|M| = k$, and a Profile of Manipulators PM such that p is the winner of $PNM \cup PM$ under r.

Our setting, Maximin

- $C = \{c_1,\dots,c_m\}$ – the set of candidates
- S, $|S| = N$ – the set of N non-manipulators
- T, $|T| = n$ – the set of n manipulators, on which we fix an order
- $N_i(c, c') = |\{ k \mid c >_k c', >_k \in S \cup \{1,\dots,i\}\}|$ – the number of voters from S and from the i first manipulators, which prefer c over c'
- $S_i(c) = \min_{c' \neq c} N_i(c, c')$ – the Maximin score of c from S and the first *i* manipulators
- Maximin winner = $\text{argmax}_c\{S_n(c)\}$
- Denote $MIN_i(c) = \{c' \in C \mid S_i(c) = N_i(c, c')\}$

CCUM Complexity

- CCUM under Maximin is NP-complete for any fixed number of manipulators (≥ 2)

  (Xia et al. IJCAI 2009)
- It follows that the UCO is not approximable by constant better than 3/2, unless P = NP
  - Otherwise, if opt = 2, then the output of the algorithm would be < 3, i.e., 2
  - Hence, it would solve the CCUM for n = 2, a contradiction

The heuristic / approximation algorithm

- The current manipulator i
  - Ranks p first
  - Builds a digraph $G^{i-1} = (V, E^{i-1})$, where
    - $V = C \setminus \{p\}$;
    - $(x, y) \in E^{i-1}$ iff $(y \in MIN_{i-1}(x)$ and $p \notin MIN_{i-1}(x))$
  - Iterates over the candidates who have not yet been ranked
  - If there is a candidate with out-degree 0, then it adds such a candidate with the lowest score
  - Otherwise, adds a vertex with the lowest score
  - Removes all the outgoing edges of vertices who had outgoing edge to newly added vertex

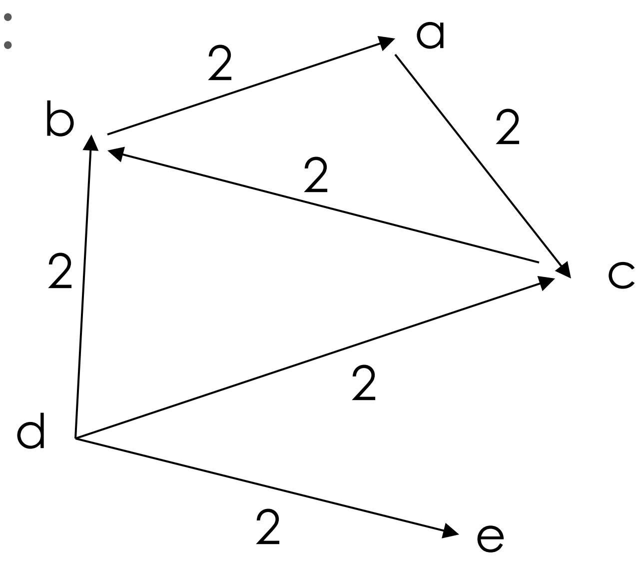
Additions to the algorithm

- The candidates with out-degree 0 are kept in stacks in order to guarantee a DFS-like order among the candidates with the same scores
- If there is no candidate (vertex) with out-degree 0, then it first searches for a cycle, with 2 adjacent vertices having the lowest scores
  - If it finds such a pair of vertices, it adds the front vertex

Example

- C = {a, b, c, d, e, p}
- |S| = 6
- |T| = 2
- The non-manipulators' votes:
  - a > b > c > d > p > e
  - a > b > c > d > p > e
  - b > c > a > p > e > d
  - b > c > p > e > d > a
  - e > d > p > c > a > b
  - e > d > p > c > a > b

G^0:



$S_0(p) = N_0(p, b) = 2$

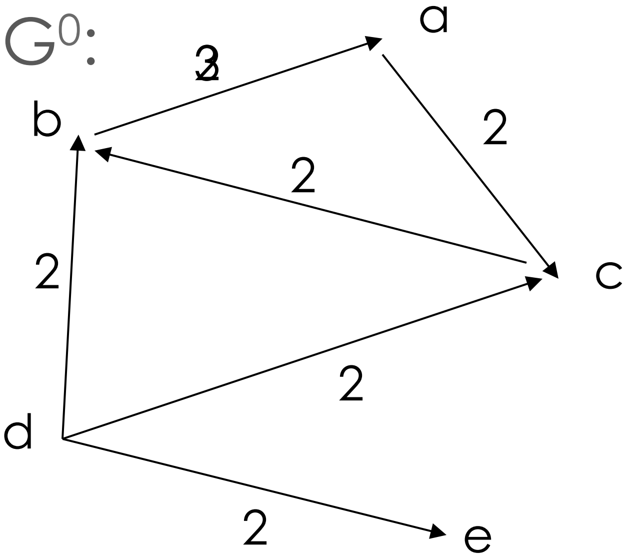
$S_0(e) = N_0(e, p) = 2$

Example (2)

- The non-manipulators' votes:
  - a > b > c > d > p > e
  - a > b > c > d > p > e
  - b > c > a > p > e > d
  - b > c > p > e > d > a
  - e > d > p > c > a > b
  - e > d > p > c > a > b
- The manipulators' votes:

p > **e** > **d** > **b** > **c** > **a**

G^0:

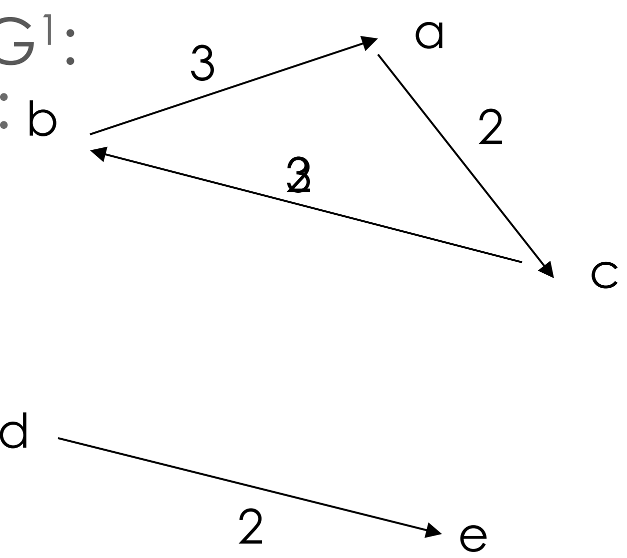


$S_0(p) = N_0(p, b) = 2$
$S_0(e) = N_0(e, p) = 2$

Example (3)

- The non-manipulators' votes:
  - a > b > c > d > p > e
  - a > b > c > d > p > e
  - b > c > a > p > e > d
  - b > c > p > e > d > a
  - e > d > p > c > a > b
  - e > d > p > c > a > b

G^1:



- The manipulators' votes:

p > **e** > **d** > **b** > **c** > **a**

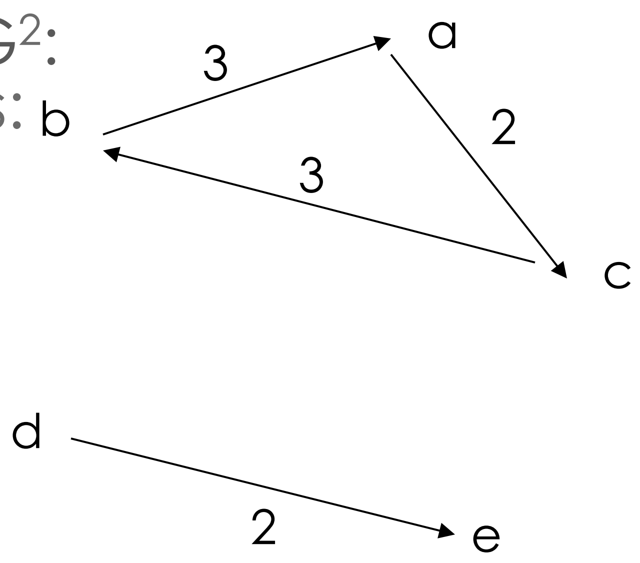
p > **e** > **d** > **c** > **a** > **b**

$S_1(p) = N_1(p, b) = 3$

$S_1(e) = N_1(e, p) = 2$

Example (4)

G^2:



- The non-manipulators' votes:
  - a > b > c > d > p > e
  - a > b > c > d > p > e
  - b > c > a > p > e > d
  - b > c > p > e > d > a
  - e > d > p > c > a > b
  - e > d > p > c > a > b

- The manipulators' votes:

  p > **e** > **d** > **b** > **c** > **a**
  p > **e** > **d** > **c** > **a** > **b**

$S_2(p) = N_2(p, b) = 4$
$\max_{x \neq p} S_2(x) = 3$

p is the winner!

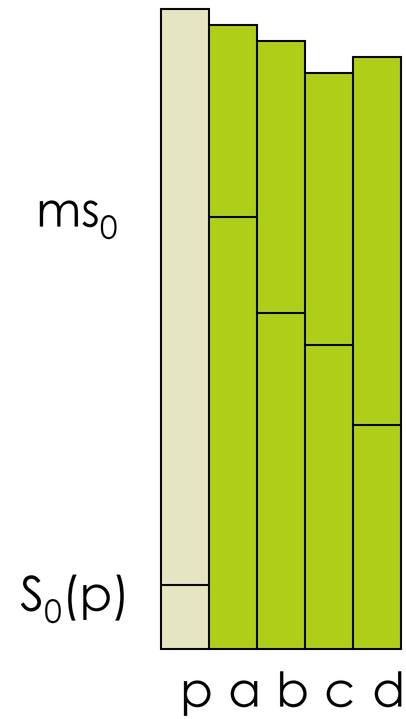
Instances without 2-cycles

- Denote $ms_i = \max_{c \neq p} S_i(c)$
  - The maximum score of *p*'s opponents after *i* stages
- Lemma: If there are no 2-cycles in the graphs built by the algorithm, then for all i, $0 \leq i \leq n-3$ it holds that $ms_{i+3} \leq ms_i + 1$
- Theorem: If there are no 2-cycles, then the algorithm gives a 5/3-approximation of the optimum

Proof of Theorem

$n = \lceil (3\, ms_0 - 3S_0(p) + 3)/2 \rceil$

$opt \geq ms_0 - S_0(p) + 1$



The ratio n/opt is the biggest when opt = 3, $n = \lceil 3/2 * 3 \rceil = 5$

Eliminating the 2-cycles

- Lemma: If at a certain stage i there are no 2-cycles, then for all $j > i$, there will be no 2-cycles at stage j
- We prove that the algorithm performs optimally while there are 2-cycles
  - Intuitively, if there is a 2-cycle, then one of its vertices has the highest score, and it will always be placed in the end – until the cycle is eliminated
- Once the 2-cycles have been eliminated, our algorithm performs a 5/3-approximation on the number of stages left
- Generally we have 5/3-approximation of the optimal solution

Conclusions

- A new heuristic / approximation algorithm for CCUM / UCO under Maximin
- Gives a 5/3-approximation to the optimum
- The lower bound on the approximation ratio of the algorithm (and any algorithm) is 1½
- Simulation results – comparison between this algorithm and the simple greedy algorithm in Zuckerman et al. 2009
- Future work
  - Prove the approx. ratio for a similar algorithm without our technical additions

Thank You

Questions?