Schulze and Ranked-Pairs Voting Are Fixed-Parameter Tractable to Bribe, Manipulate, and Control

Lane A. Hemaspaandra, Rahman Lavaee, and Curtis Menton

UNIVERSITY of ROCHESTER

Problem

Schulze and ranked-pairs elections have been proven resistant to bribery and many cases of control (the attack problems are NP-complete).

Is it possible to carry out these attacks (or to find out that it is impossible to attack) using algorithms that for each fixed number of candidates run in time $O(n^k)$, where k is independent of the number of candidates?

In other words, are the attack problems *fixed parameter tractable* (FPT) when parameterizing on the number of candidates?

Results

Bribery, all the control cases, and manipulation are in FPT for Schulze and ranked pairs, parameterized on the number of candidates.

Approach

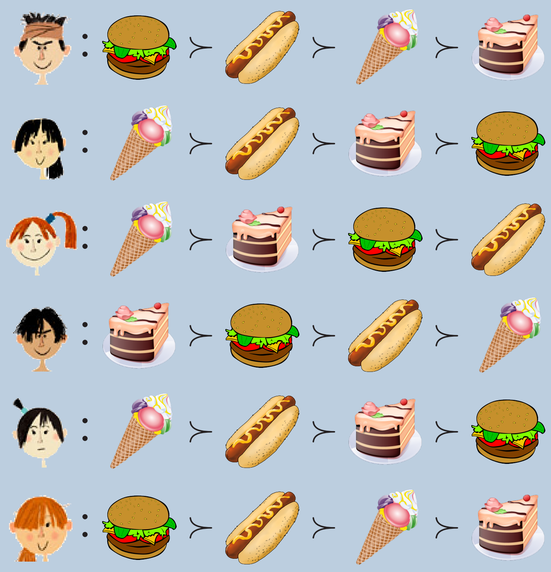
The main idea is to find a structure that on one hand is rich enough to specify the exact winner set, and yet on the other hand is so restrictive that the number of such structures is bounded purely as a function of the number of candidates.

This allows us to loop over all the structures that satisfy the goal of the attack and for each of those directly wedge it into an integer linear programming feasibility problem (ILPFP) to see if that structure can be achieved by setting up the attack appropriately. The attack itself has to be formulated within the ILPFP.

The number of constraints and variables in the integer linear program is guaranteed to be bounded as a function of the number of candidates. Thus we can solve it using Lenstra's algorithm and achieve FPT performance overall. For both Schulze and ranked pairs we find this type of structure.

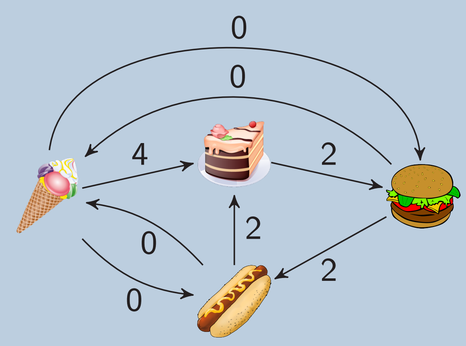
Schulze Voting

Voters cast their votes:



We build the Weighted Majority Graph (WMG) based on these preferences (the weight of the WMG edge from a to b is the number of voters preferring a to b minus the ones preferring b to a).

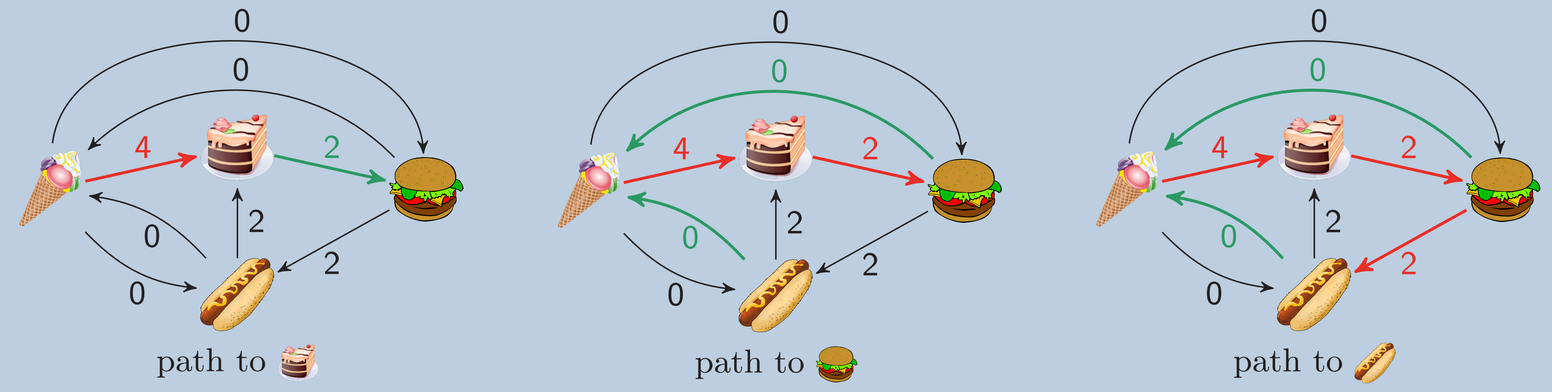
Let the strength of a path be the weakest link in that path. Each candidate a whose strongest path to every other candidate b is at least as strong as every path from b back to a is a winner. Here is the unique winner.



Schulze Winner Set Certification Framework

How can it be certified in a succinct way that the Schulze winner set is { }?

We do this using Schulze winner set certification framework (SWCF). Such a structure specifies for each winner a a strong path γ_{ab} to every other candidate b that is at least as strong as every simple path from b back to a (in each such path it specifies a weak link that is dominated by every edge in γ_{ab}).



Also, for each nonwinner b, it specifies a path from some candidate to b that is strictly stronger than any simple path from b to that candidate (here the same paths above satisfy this criterion).

Integer Linear Programming Feasibility Problems

Every SWCF merely consists of inequalities between the weights of the WMG edges. The challenge is then how to represent these weights in our linear programs. If j is the number of candidates then there are $j!$ different types of votes. The weight of the edge from a to b equals

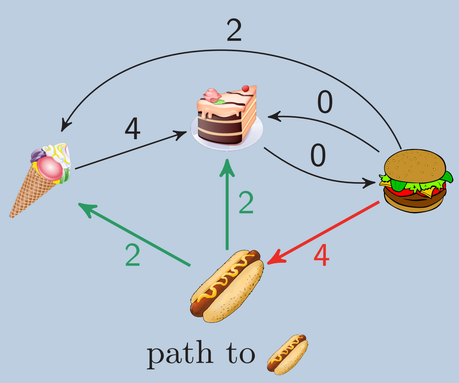
$$D(a,b) = \Big(\sum_{\substack{\{i \mid 1 \le i \le j! \text{ and the} \\ \text{vote type } i \text{ prefers } a \text{ to } b\}}} v_i\Big) - \Big(\sum_{\substack{\{i \mid 1 \le i \le j! \text{ and the} \\ \text{vote type } i \text{ prefers } b \text{ to } a\}}} v_i\Big),$$

where v_i is the number of votes of type i. In order to handle various types of attacks we will introduce variables and constants to allow v_i to be replaced with an appropriate expression.

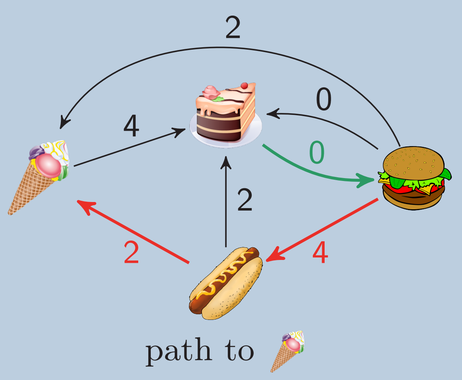
Example: Bribery

Suppose that we want to bribe one of the voters to make a Schulze winner. We need to find out which voter to bribe and which vote we want her to cast instead. While looping over SWCFs, we reach the one shown below.

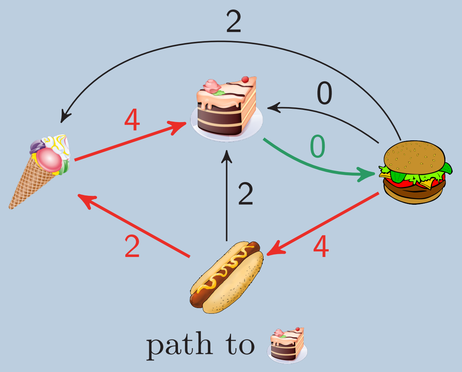
To formulate the constraints into an integer linear programming feasibility problem, we replace each v_i with $n_i - \sum_{1\le\ell\le j!} m_{i,\ell} + \sum_{1\le\ell\le j!} m_{\ell,i}$ where n_i is the number of votes of type i in the original election and $m_{i,\ell}$ is a variable representing how many voters will be bribed from casting a vote of type i to cast a vote of type ℓ.
We solve the resulting integer linear program and get $m_{\langle \succ \succ \succ \rangle,\langle \succ \succ \succ \rangle} = 1$. So we can accomplish our goal by bribing to vote $\succ \succ \succ$.



$D(,) \ge D(,)$
$D(,) \ge D(,)$



$D(,) \ge D(,)$
$D(,) \ge D(,)$



$D(,) \ge D(,)$
$D(,) \ge D(,)$
$D(,) \ge D(,)$