Voting in Two-Crossing Elections

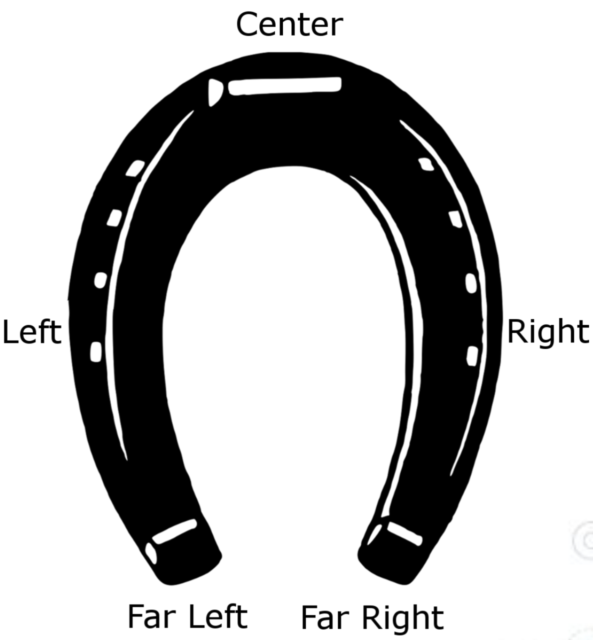
Andrei Constantinescu
Roger Wattenhofer

Distributed Computing Group
ETH zürich

1.

Motivation

The Horseshoe Theory

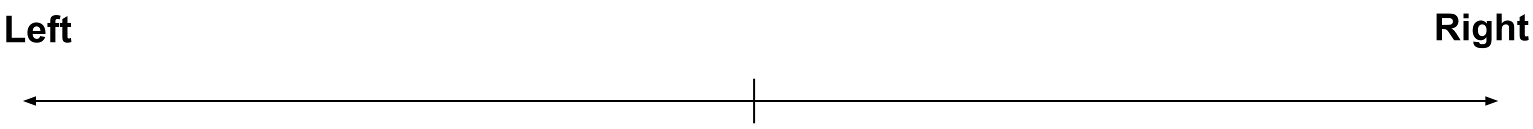


Left-Right Spectrum

Candidate c located at x(c); voter v has ideal point x(v).

Preference by Euclidean distance

Left-Right Spectrum

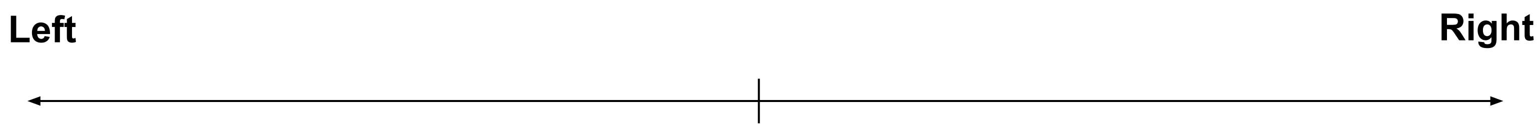


Candidate c located at x(c); voter v has ideal point x(v).

Preference by Euclidean distance

Left-Right Spectrum

Candidates: $c_1, \ldots, c_M$;

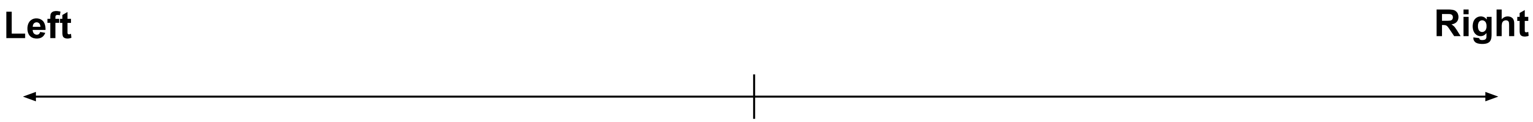


Candidate c located at x(c); voter v has ideal point x(v).

Preference by Euclidean distance

Left-Right Spectrum

Candidates: $c_1, \ldots, c_M$; Voters: $v_1, \ldots, v_N$.

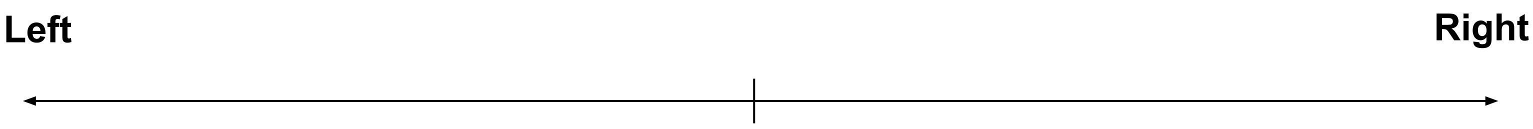


Candidate c located at x(c); voter v has ideal point x(v).

Preference by Euclidean distance

Left-Right Spectrum

Candidates: $c_1, \ldots, c_M$; Voters: $v_1, \ldots, v_N$.

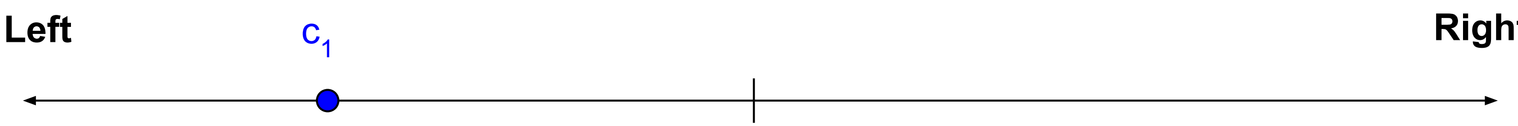


Candidate c located at x(c); voter v has ideal point x(v).

Preference by Euclidean distance

Left-Right Spectrum

Candidates: $c_1, \ldots, c_M$; Voters: $v_1, \ldots, v_N$.

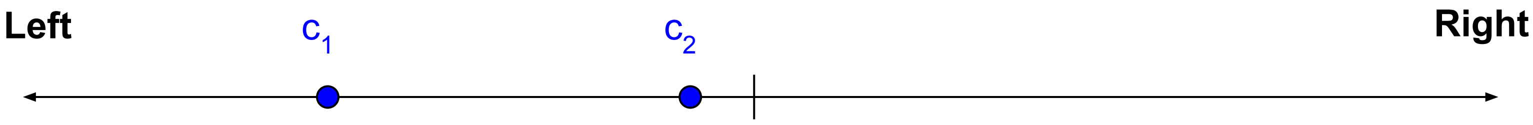


Candidate c located at $x(c)$; voter v has ideal point x(v).

Preference by Euclidean distance

Left-Right Spectrum

Candidates: $c_1, \ldots, c_M$; Voters: $v_1, \ldots, v_N$.

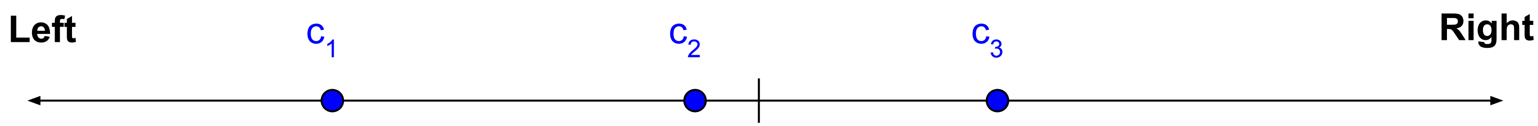


Candidate c located at x(c); voter v has ideal point x(v).

Preference by Euclidean distance

Left-Right Spectrum

Candidates: $c_1, \ldots, c_M$; Voters: $v_1, \ldots, v_N$.

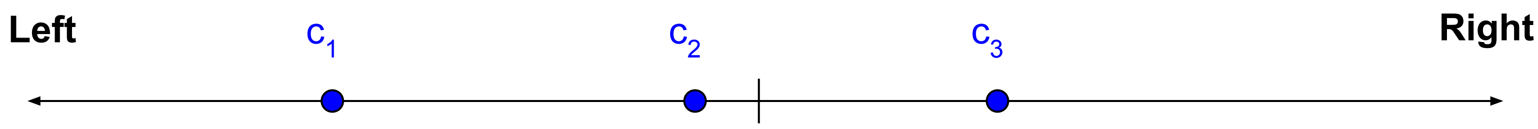


Candidate c located at $x(c)$, voter v has ideal point $x(v)$.

Preference by Euclidean distance

Left-Right Spectrum

Candidates: $c_1, \ldots, c_M$; Voters: $v_1, \ldots, v_N$.

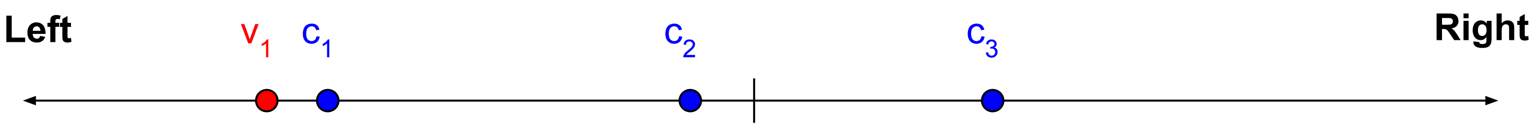


Candidate c located at $x(c)$; voter v has ideal point $x(v)$.

Preference by Euclidean distance

Left-Right Spectrum

Candidates: $c_1, \ldots, c_M$; Voters: $v_1, \ldots, v_N$.

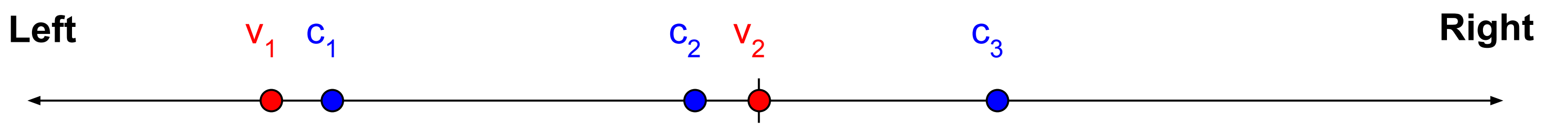


Candidate c located at $x(c)$; voter v has ideal point $x(v)$.

Preference by Euclidean distance

Left-Right Spectrum

Candidates: $c_1, \ldots, c_M$; Voters: $v_1, \ldots, v_N$.

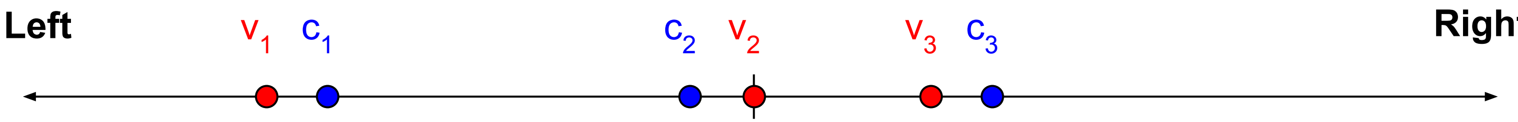


Candidate c located at $x(c)$; voter v has ideal point $x(v)$.

Preference by Euclidean distance

Left-Right Spectrum

Candidates: $c_1, \ldots, c_M$; Voters: $v_1, \ldots, v_N$.

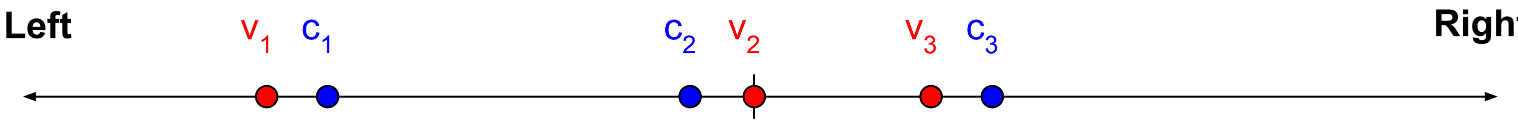


Candidate c located at $x(c)$; voter v has ideal point $x(v)$.

Preference by Euclidean distance

Left-Right Spectrum

Candidates: $c_1, \ldots, c_M$; Voters: $v_1, \ldots, v_N$.

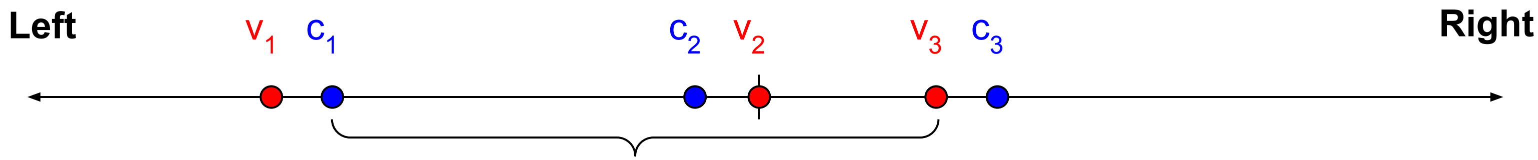


Candidate c located at $x(c)$; voter v has ideal point $x(v)$.

Preference by Euclidean distance

Left-Right Spectrum

Candidates: $c_1, \ldots, c_M$; Voters: $v_1, \ldots, v_N$.

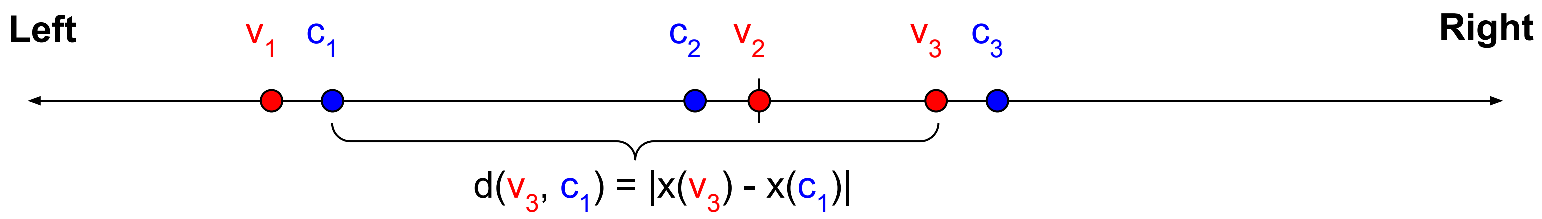


Candidate c located at x(c); voter v has ideal point x(v).

Preference by Euclidean distance

Left-Right Spectrum

Candidates: $c_1, \ldots, c_M$; Voters: $v_1, \ldots, v_N$.

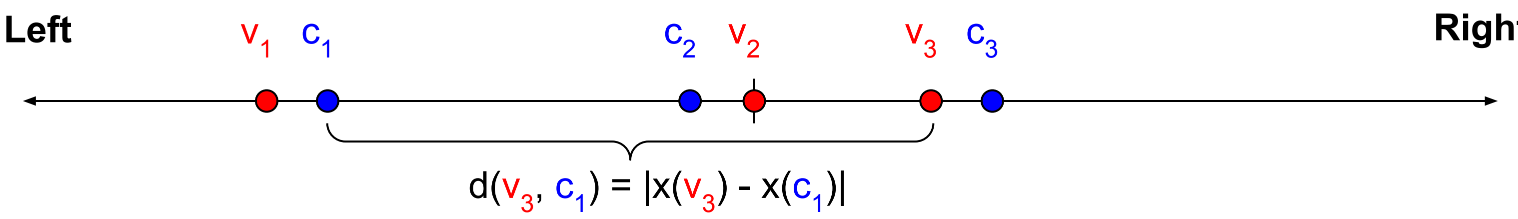


Candidate c located at $x(c)$; voter v has ideal point $x(v)$.

Preference by Euclidean distance

Left-Right Spectrum

Candidates: $c_1, \ldots, c_M$; Voters: $v_1, \ldots, v_N$.

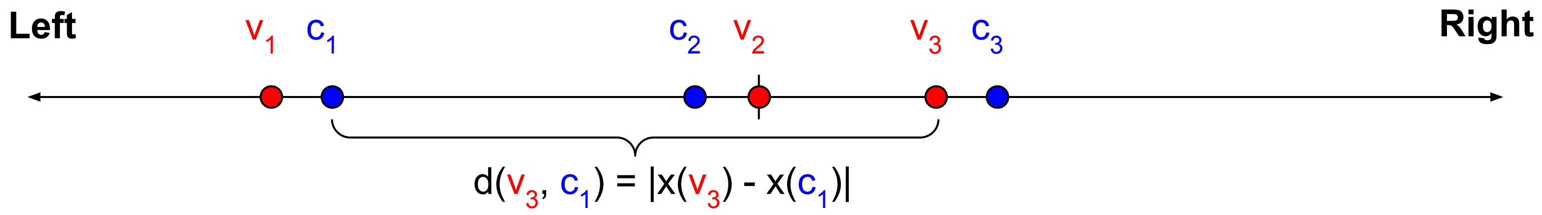


Candidate c located at $x(c)$; voter v has ideal point $x(v)$.

Preference by Euclidean distance → $v_i : c_j \succ c_k$

Left-Right Spectrum

Candidates: $c_1, \ldots, c_M$; Voters: $v_1, \ldots, v_N$.

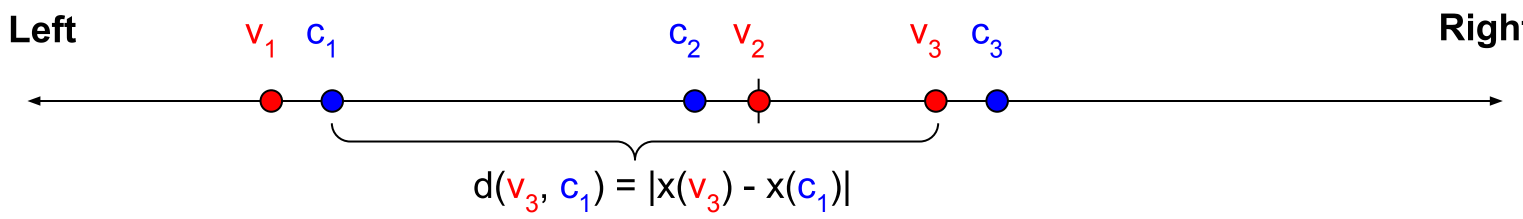


Candidate c located at $x(c)$; voter v has ideal point $x(v)$.

Preference by Euclidean distance → $v_i : c_j \succ c_k$ iff $d(v_i, c_j) < d(v_i, c_k)$.

Left-Right Spectrum

Candidates: $c_1, \ldots, c_M$; Voters: $v_1, \ldots, v_N$.



Candidate c located at $x(c)$; voter v has ideal point $x(v)$.

Preference by Euclidean distance → $v_i : c_j \succ c_k$ iff $d(v_i, c_j) < d(v_i, c_k)$.

i.e.

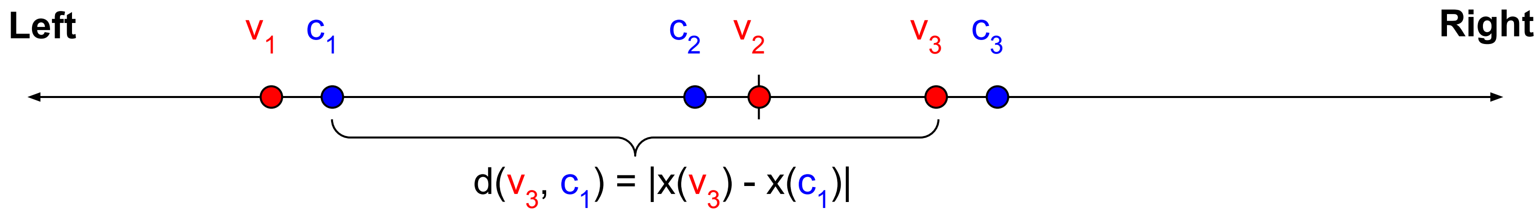
$v_1 : c_1 \succ c_2 \succ c_3$

$v_2 : c_2 \succ c_3 \succ c_1$

$v_3 : c_3 \succ c_2 \succ c_1$

Left-Right Spectrum

Candidates: $c_1, \ldots, c_M$; Voters: $v_1, \ldots, v_N$.



Candidate c located at $x(c)$; voter v has ideal point $x(v)$.

Preference by Euclidean distance → $v_i : c_j \succ c_k$ iff $d(v_i, c_j) < d(v_i, c_k)$.

i.e.

Majority Tournament

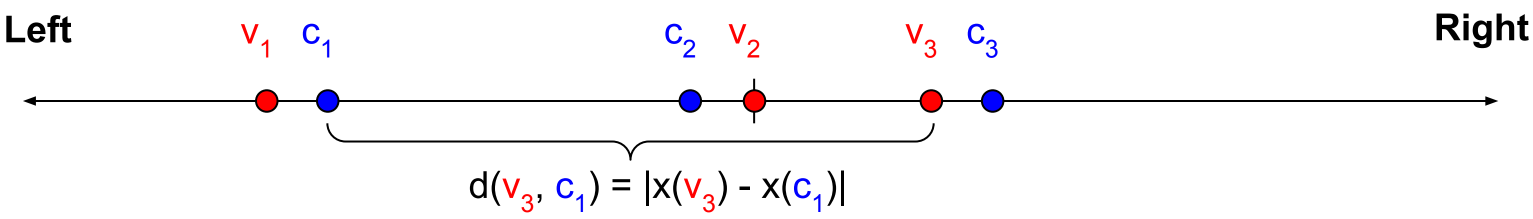
$v_1 : c_1 \succ c_2 \succ c_3$

$v_2 : c_2 \succ c_3 \succ c_1$

$v_3 : c_3 \succ c_2 \succ c_1$

Left-Right Spectrum

Candidates: $c_1, \ldots, c_M$; Voters: $v_1, \ldots, v_N$.



Candidate c located at $x(c)$; voter v has ideal point $x(v)$.

Preference by Euclidean distance $\rightarrow v_i : c_j \succ c_k$ iff $d(v_i, c_j) < d(v_i, c_k)$.

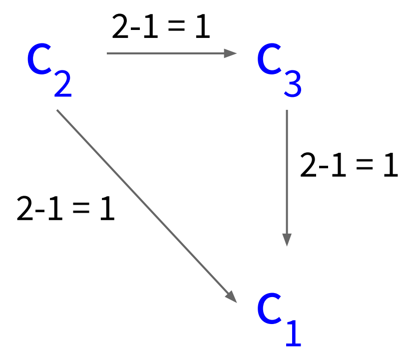
i.e.

$v_1 : c_1 \succ c_2 \succ c_3$

$v_2 : c_2 \succ c_3 \succ c_1$

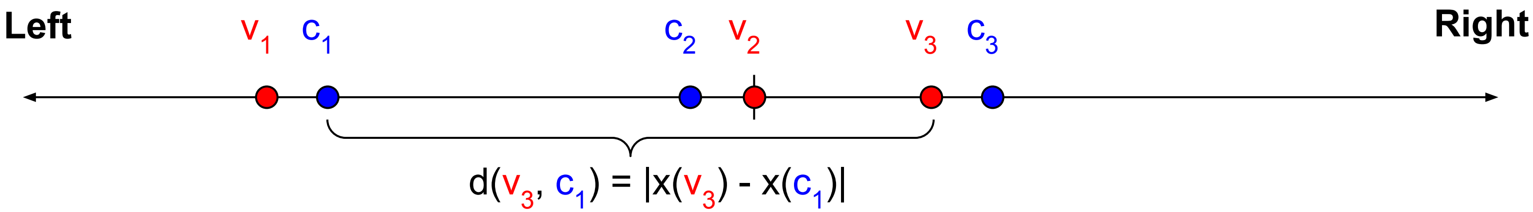
$v_3 : c_3 \succ c_2 \succ c_1$

Majority Tournament



Left-Right Spectrum

Candidates: $c_1, \ldots, c_M$; Voters: $v_1, \ldots, v_N$.



Candidate c located at $x(c)$; voter v has ideal point $x(v)$.

Preference by Euclidean distance → $v_i : c_j \succ c_k$ iff $d(v_i, c_j) < d(v_i, c_k)$.

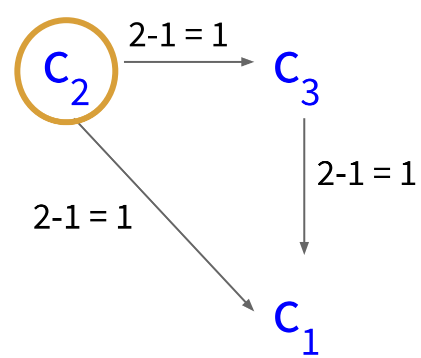
i.e.

$v_1 : c_1 \succ c_2 \succ c_3$

$v_2 : c_2 \succ c_3 \succ c_1$

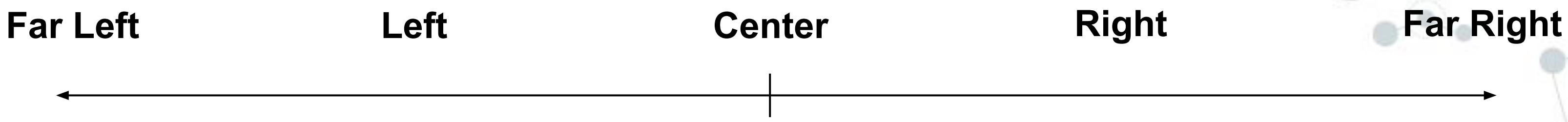
$v_3 : c_3 \succ c_2 \succ c_1$

Majority Tournament

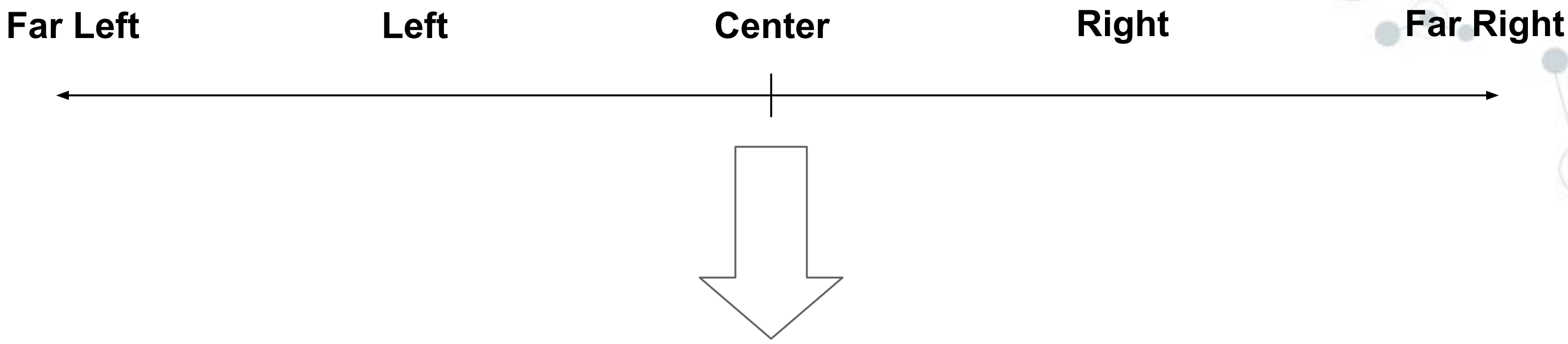


Horseshoe Spectrum

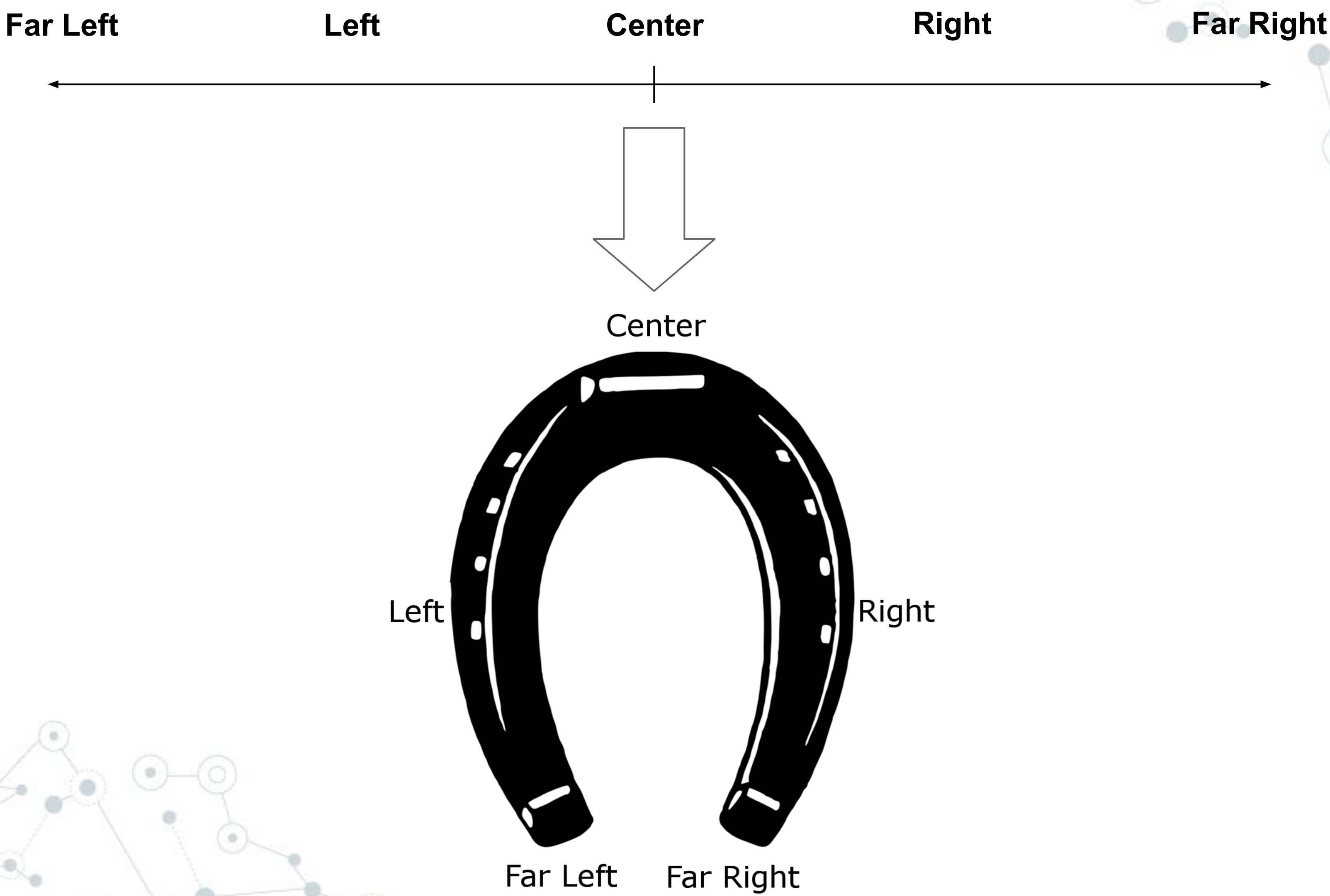
Horseshoe Spectrum



Horseshoe Spectrum

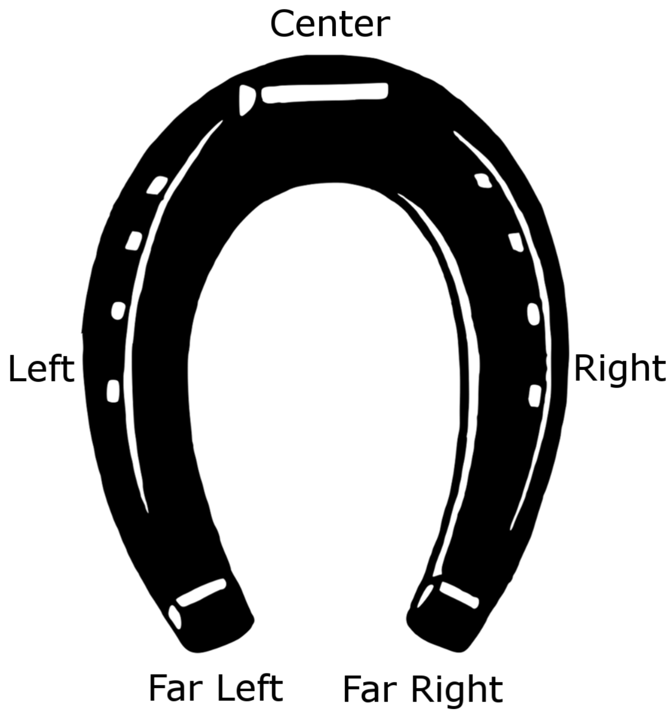


Horseshoe Spectrum

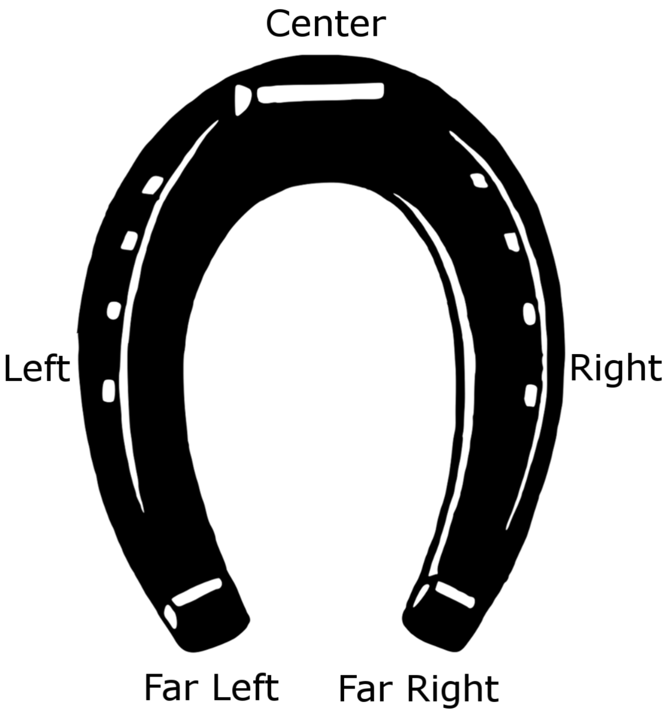


Horseshoe Spectrum

Horseshoe Spectrum

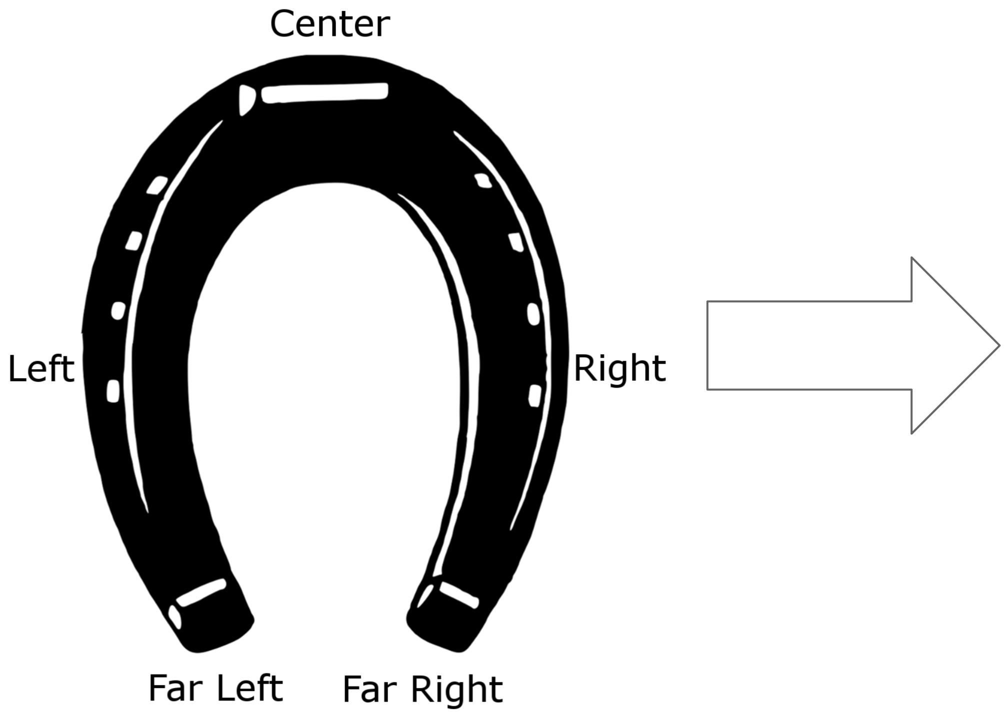


Horseshoe Spectrum



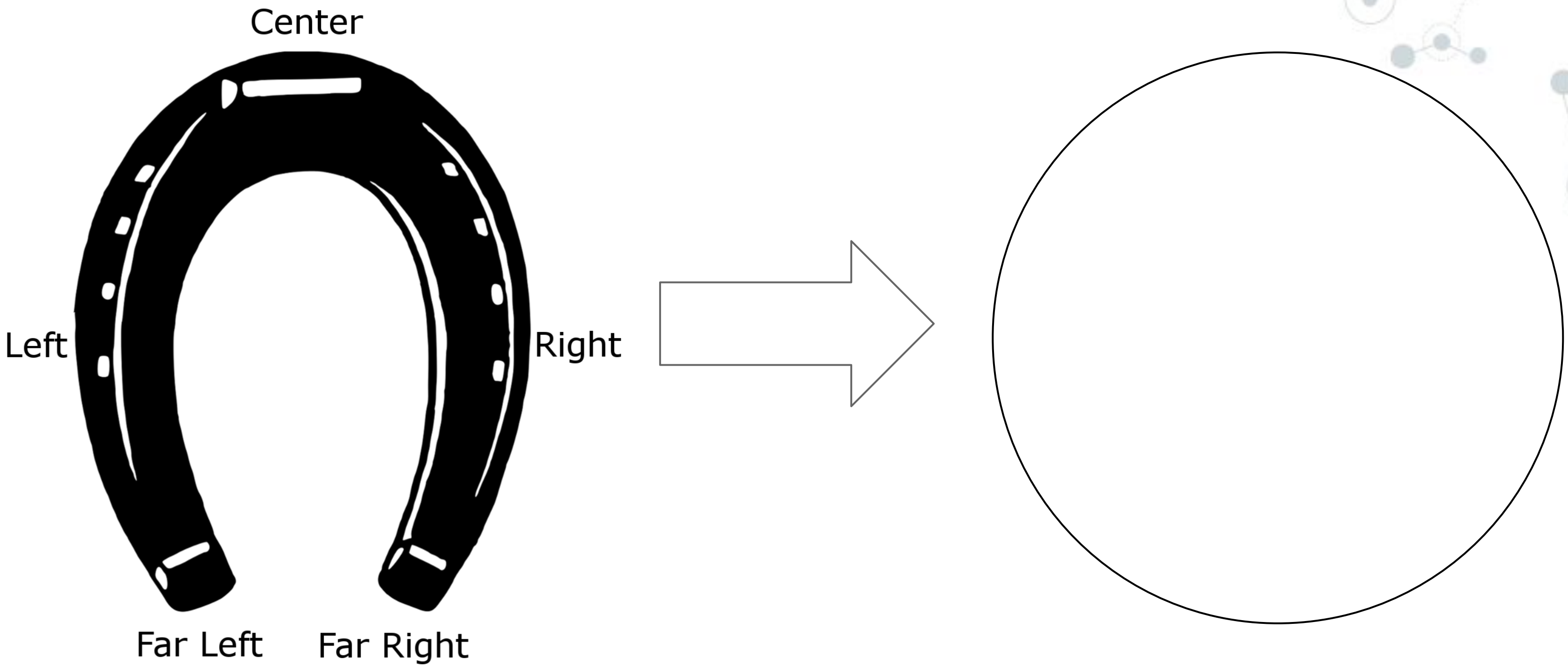
"Unholy Alliance"

Horseshoe Spectrum



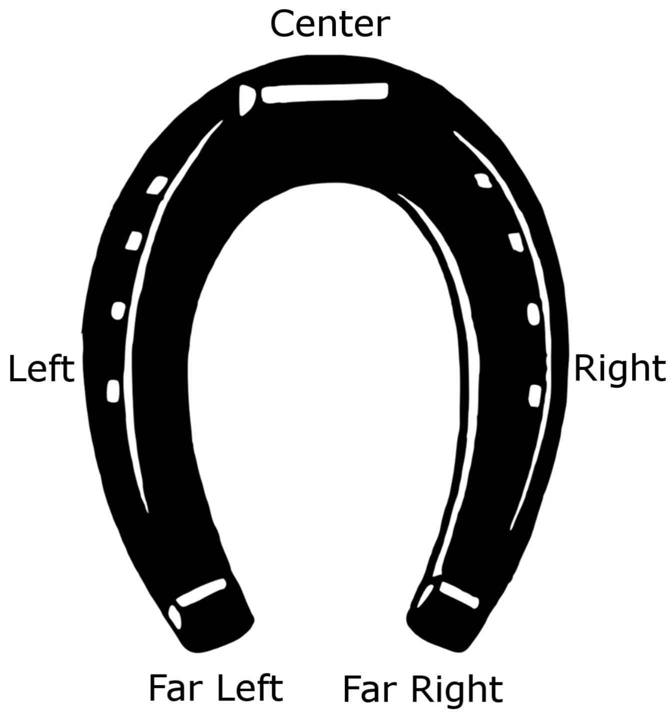
"Unholy Alliance"

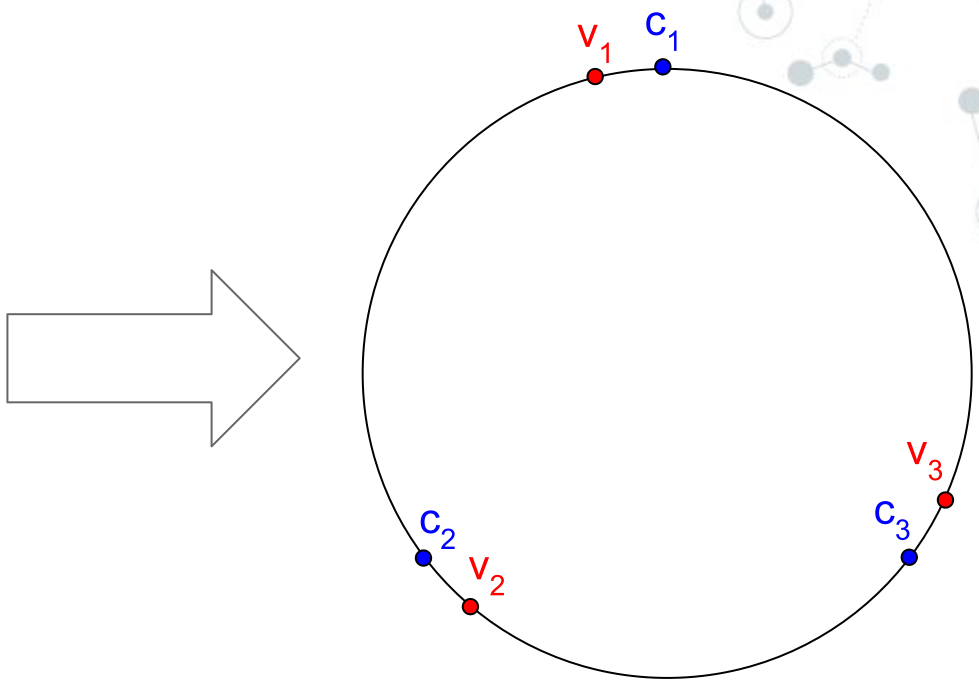
Horseshoe Spectrum



"Unholy Alliance"

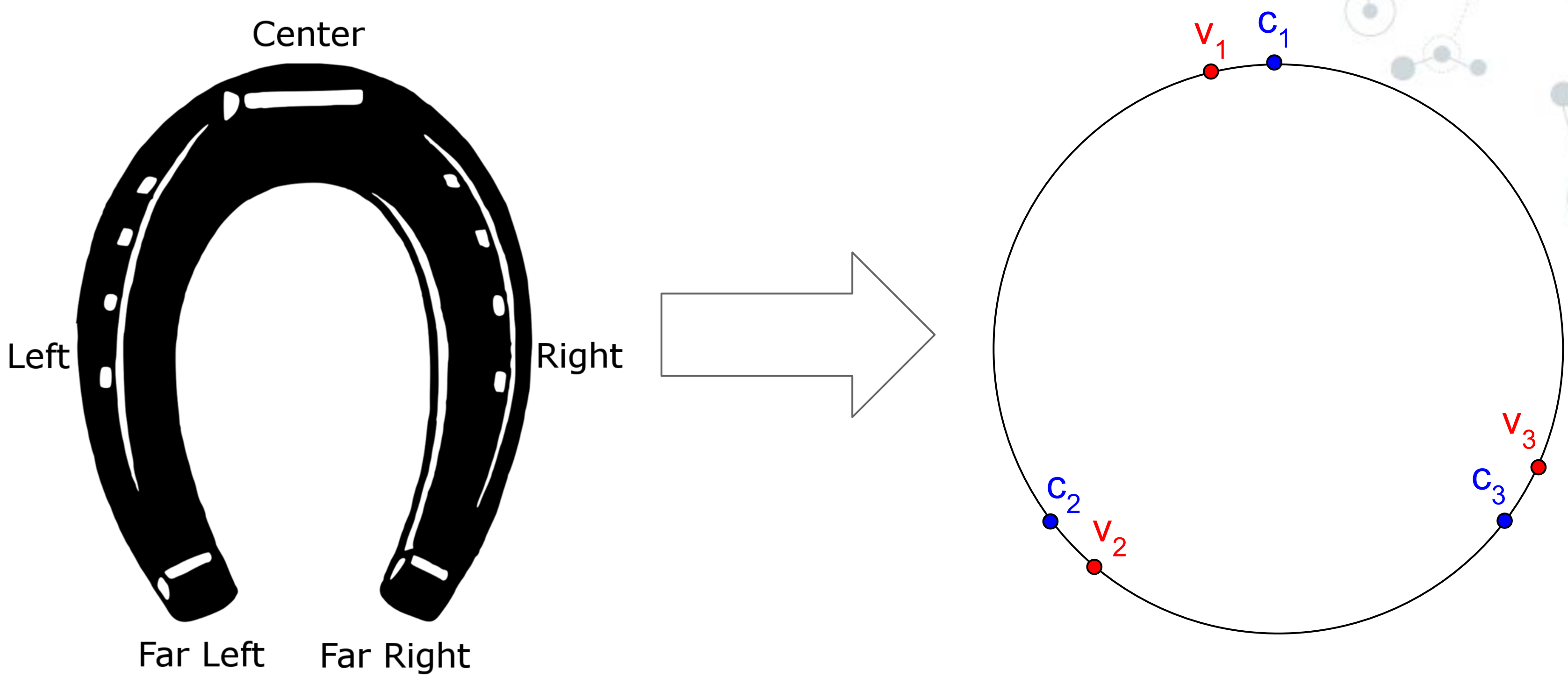
Horseshoe Spectrum





"Unholy Alliance"

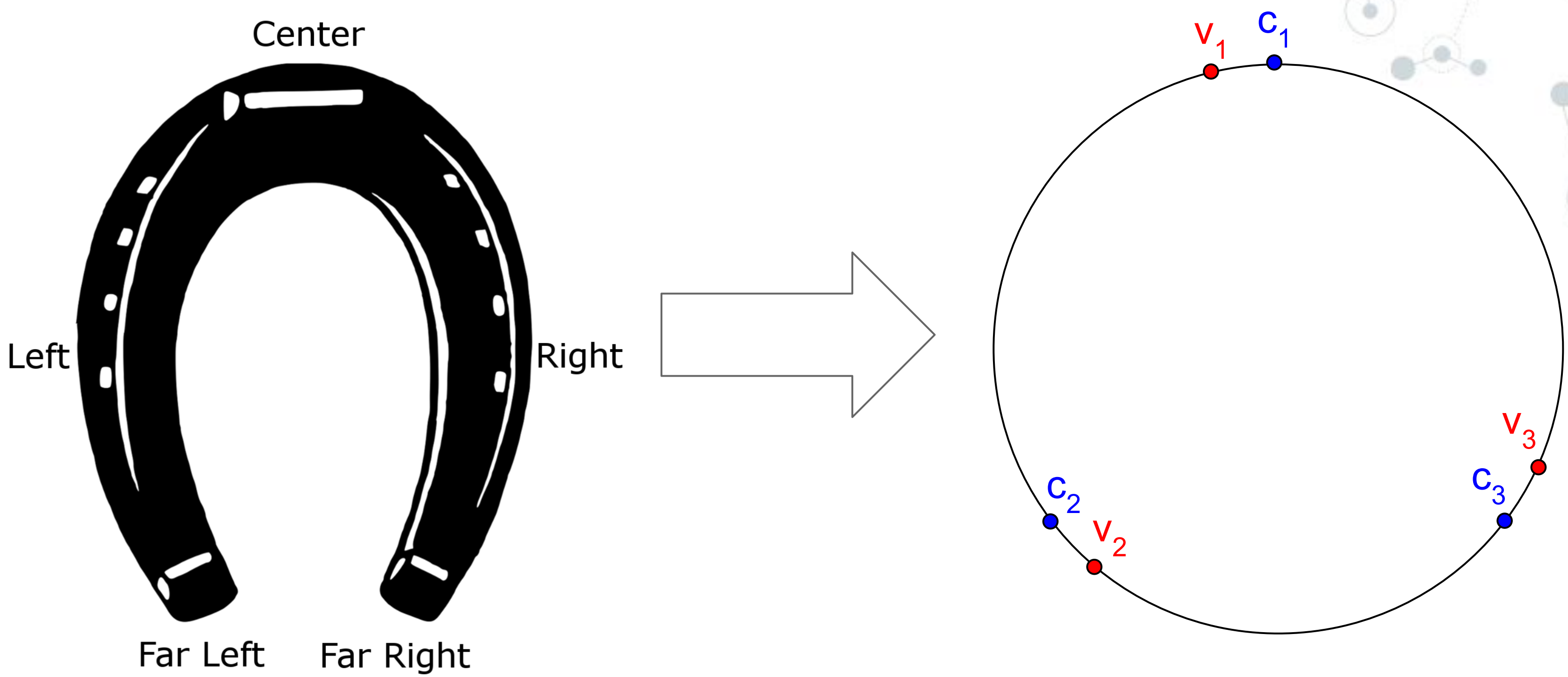
Horseshoe Spectrum



"Unholy Alliance"

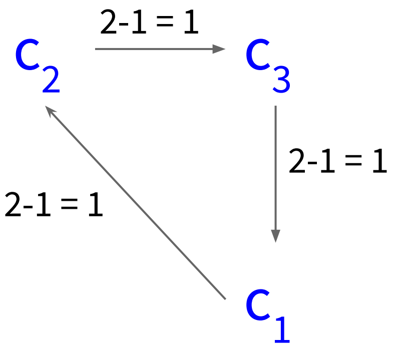
$$v_1 : c_1 \succ c_2 \succ c_3$$
$$v_2 : c_2 \succ c_3 \succ c_1$$
$$v_3 : c_3 \succ c_1 \succ c_2$$

Horseshoe Spectrum

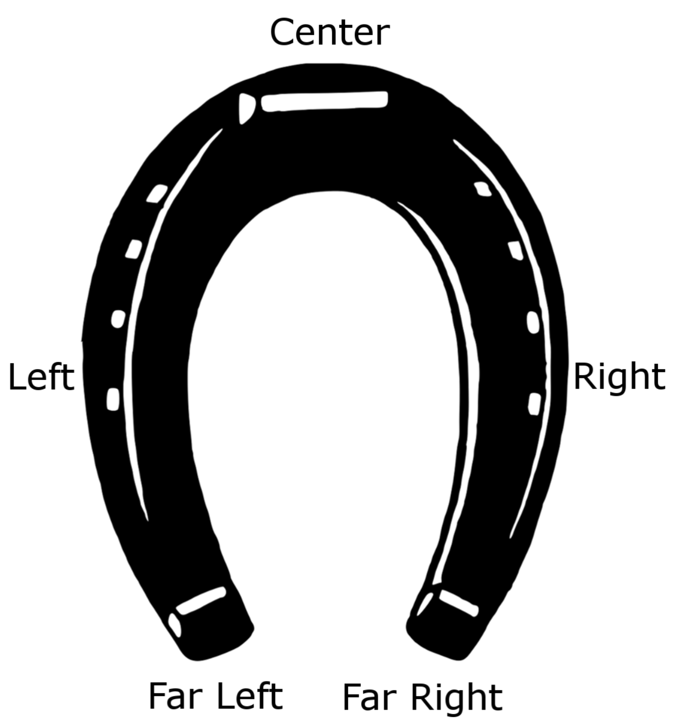


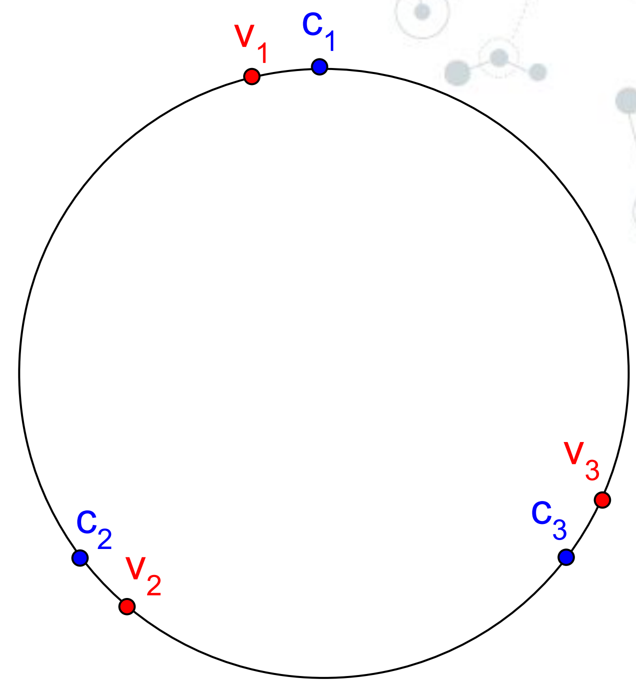
"Unholy Alliance"

$$v_1 : c_1 \succ c_2 \succ c_3$$
$$v_2 : c_2 \succ c_3 \succ c_1$$
$$v_3 : c_3 \succ c_1 \succ c_2$$



Horseshoe Spectrum



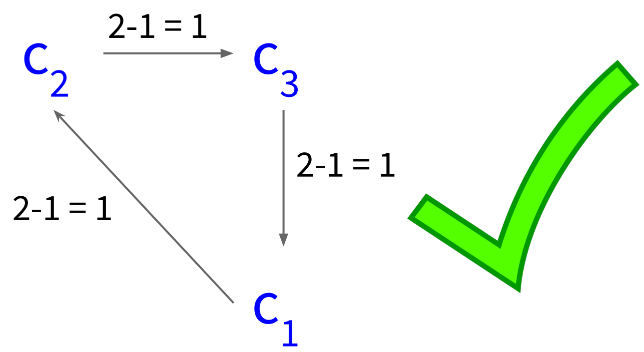


"Unholy Alliance"

$v_1 : c_1 \succ c_2 \succ c_3$

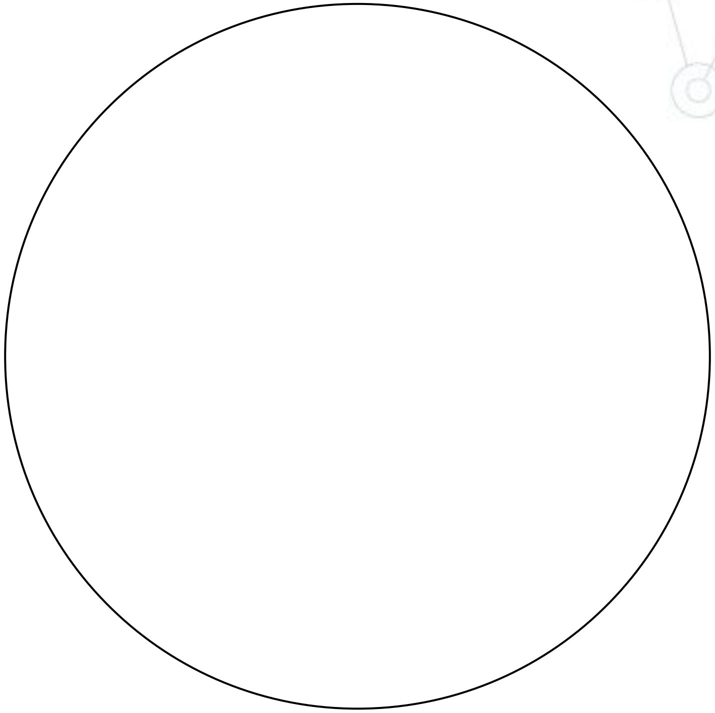
$v_2 : c_2 \succ c_3 \succ c_1$

$v_3 : c_3 \succ c_1 \succ c_2$



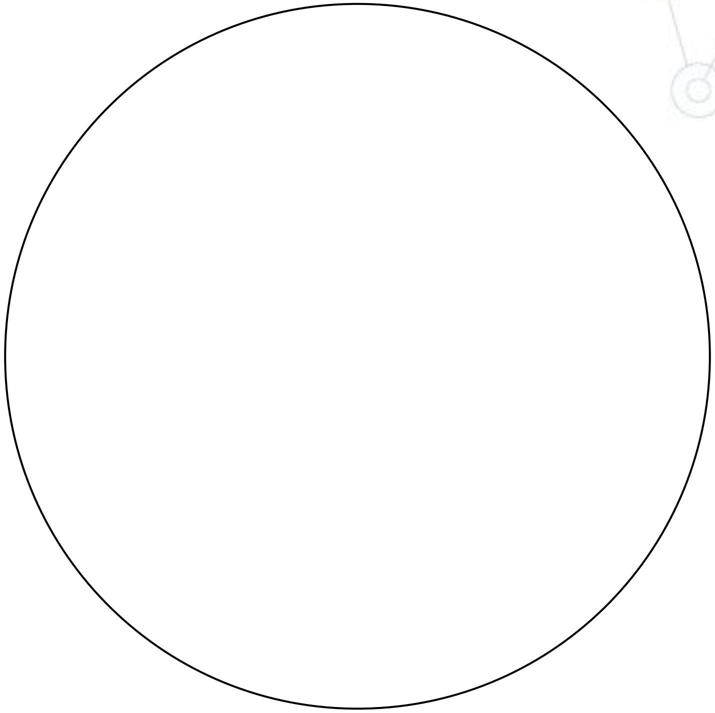
A More General Property

A More General Property



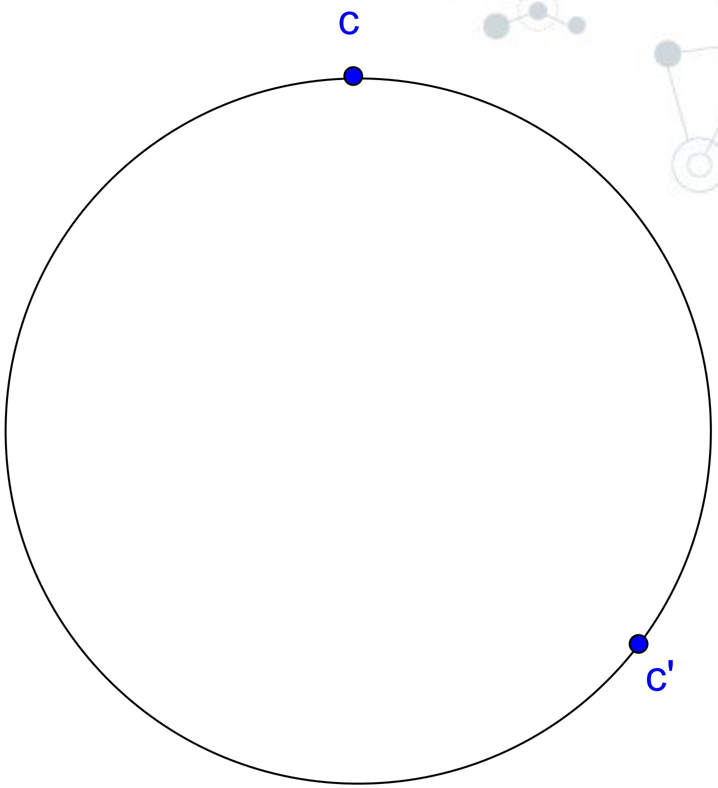
A More General Property

Consider candidates c and c'.



A More General Property

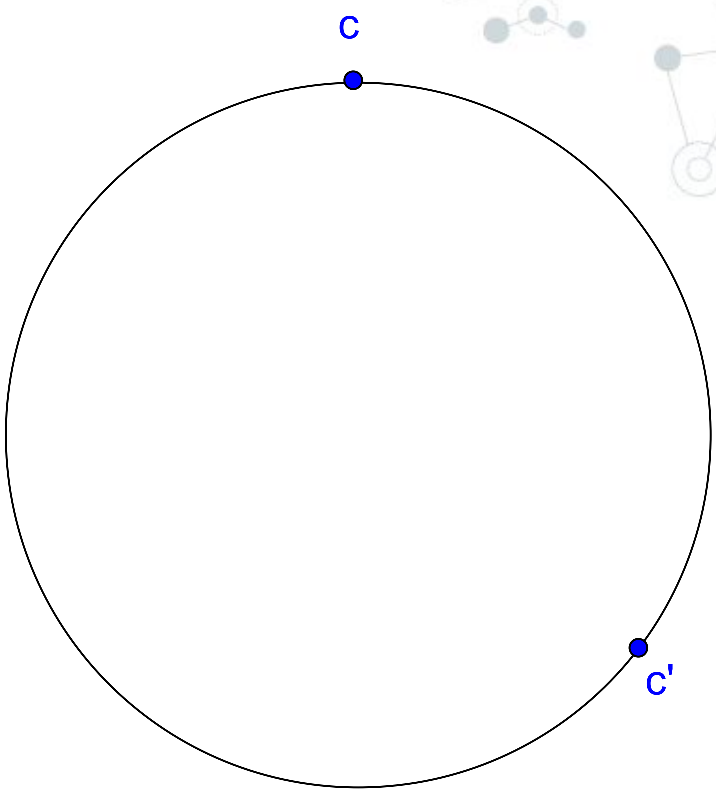
Consider candidates c and c'.



A More General Property

Consider candidates c and c'.

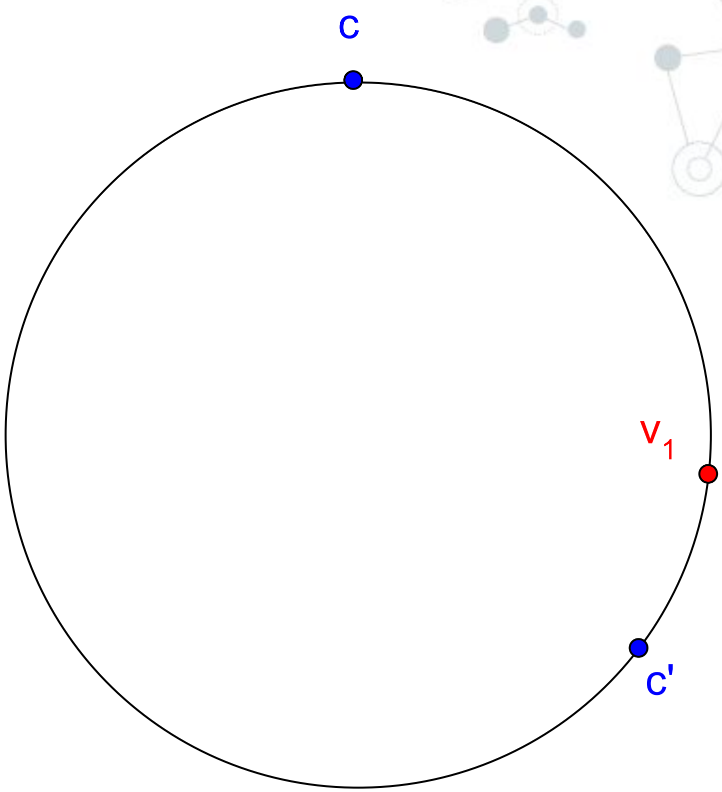
And voters sorted by angle.



A More General Property

Consider candidates c and c'.

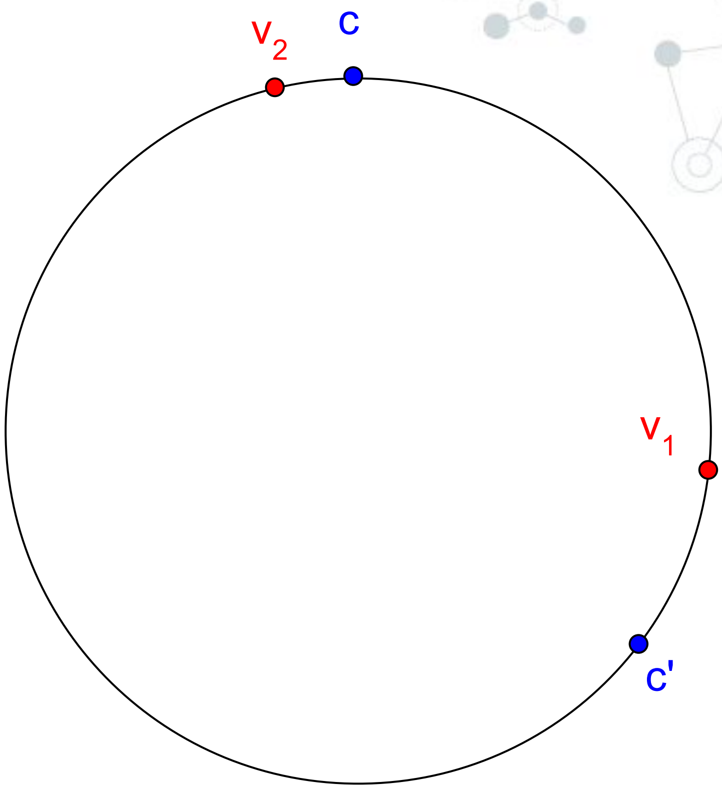
And voters sorted by angle.



A More General Property

Consider candidates c and c'.

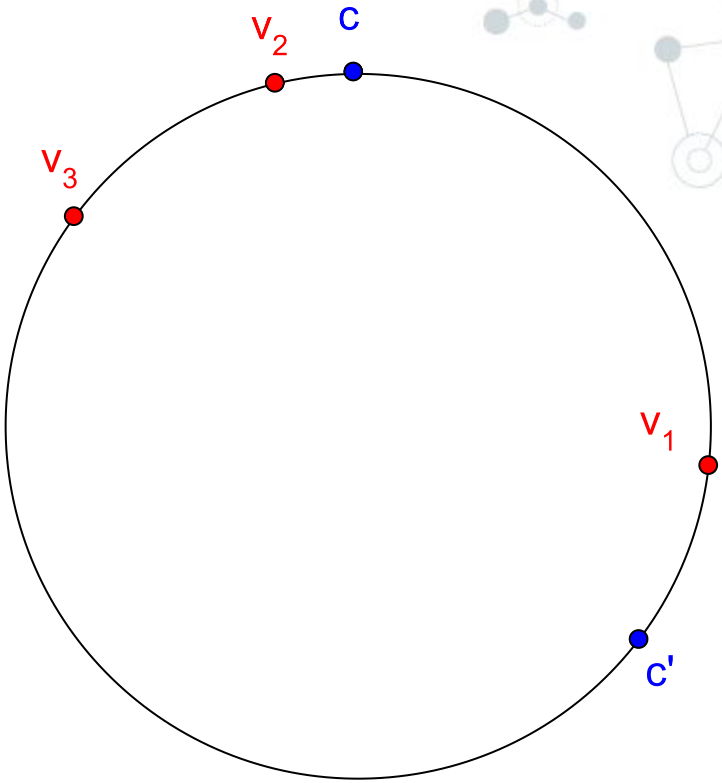
And voters sorted by angle.



A More General Property

Consider candidates c and c'.

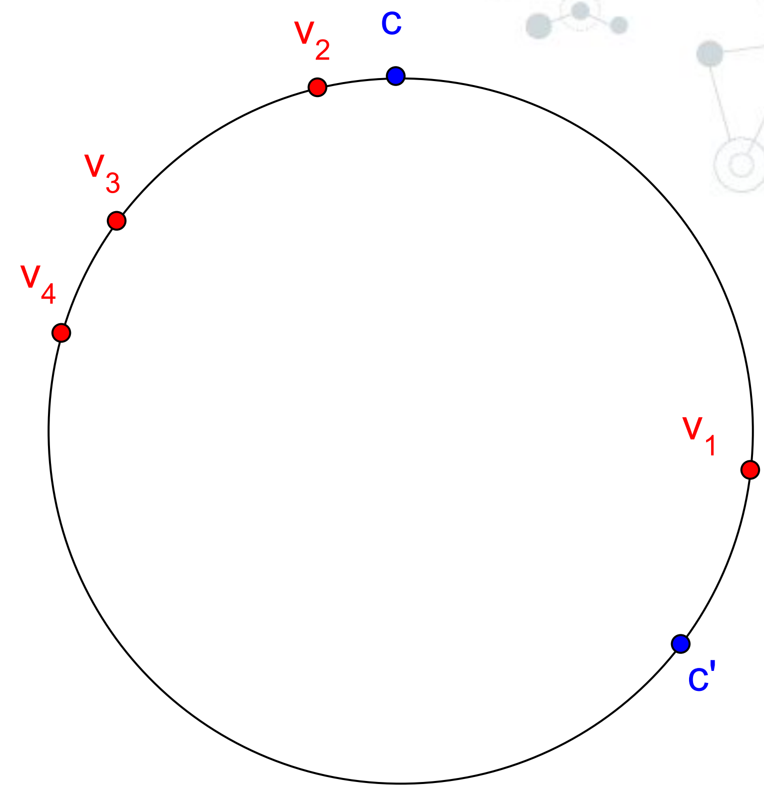
And voters sorted by angle.



A More General Property

Consider candidates c and c'.

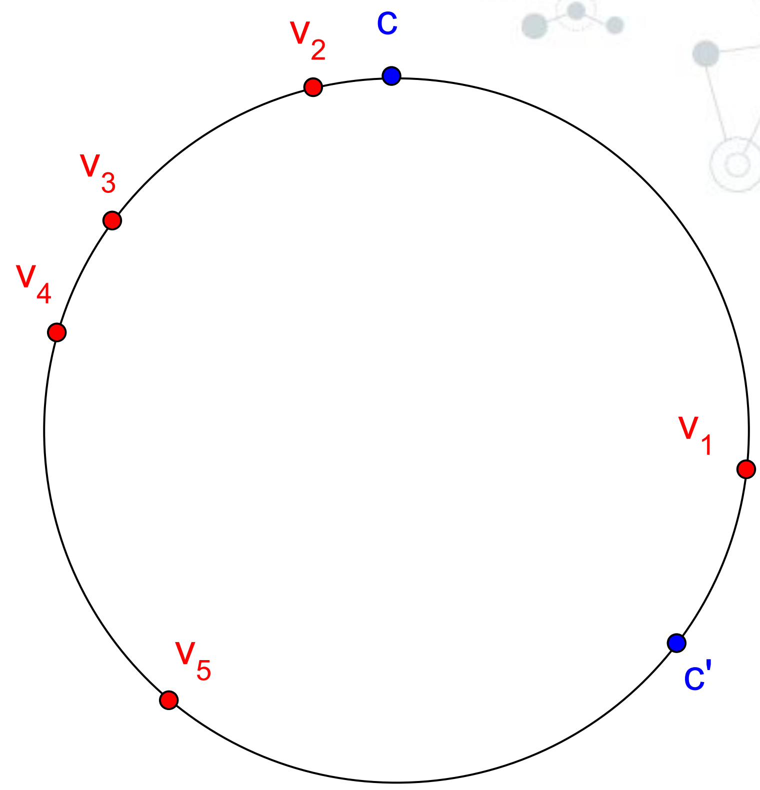
And voters sorted by angle.



A More General Property

Consider candidates c and c'.

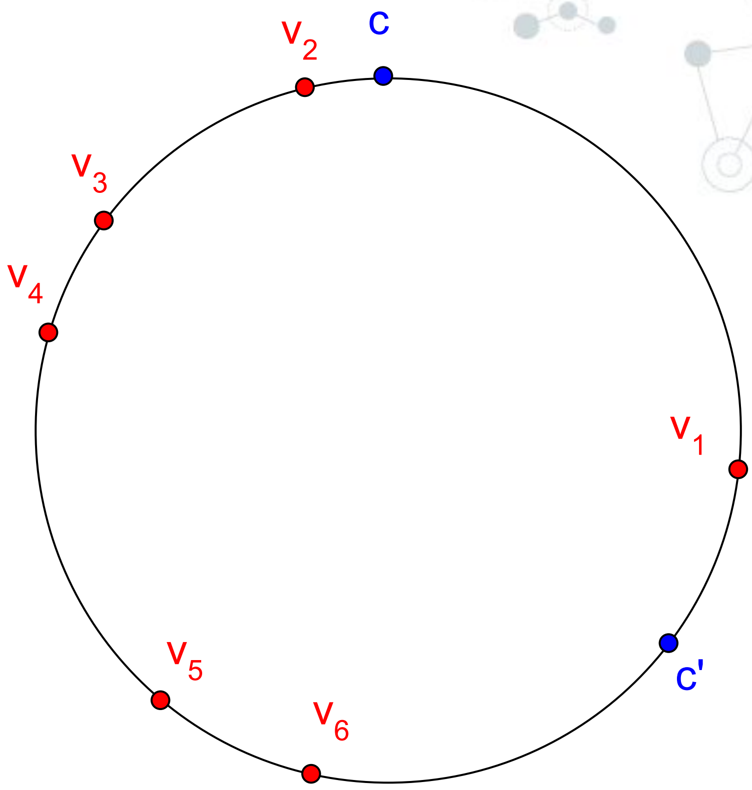
And voters sorted by angle.



A More General Property

Consider candidates c and c'.

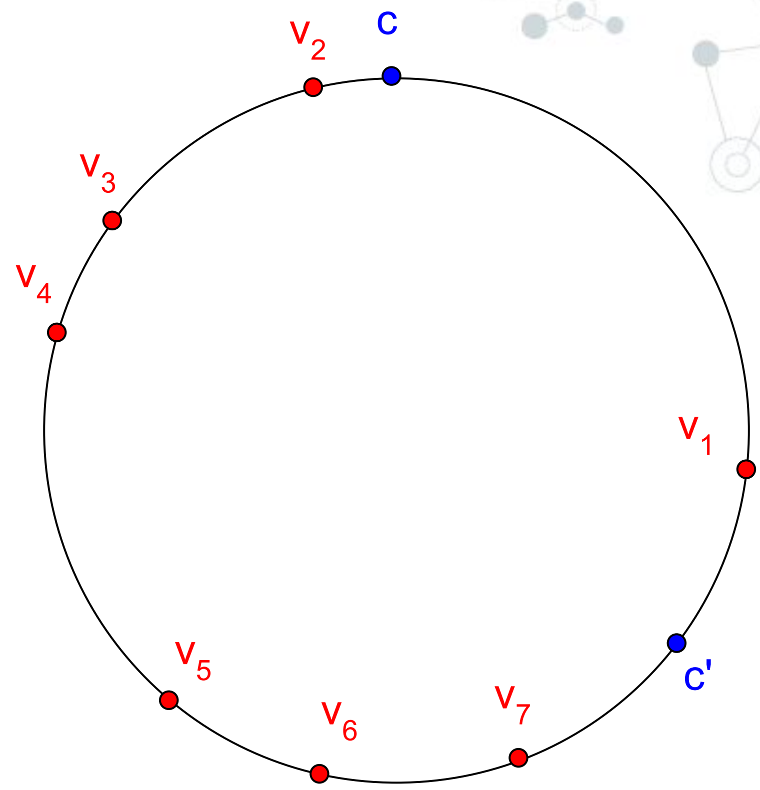
And voters sorted by angle.



A More General Property

Consider candidates c and c'.

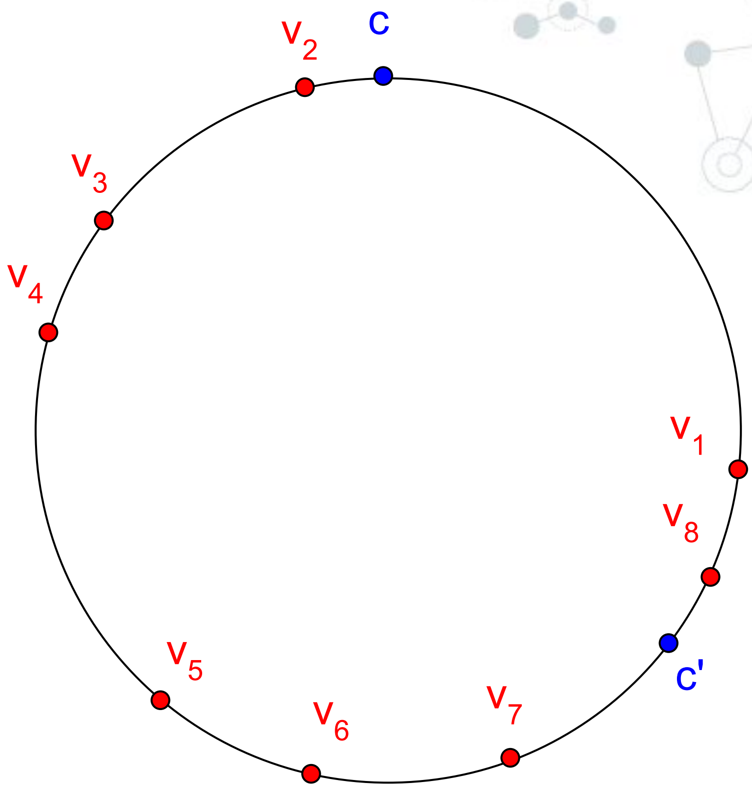
And voters sorted by angle.



A More General Property

Consider candidates c and c'.

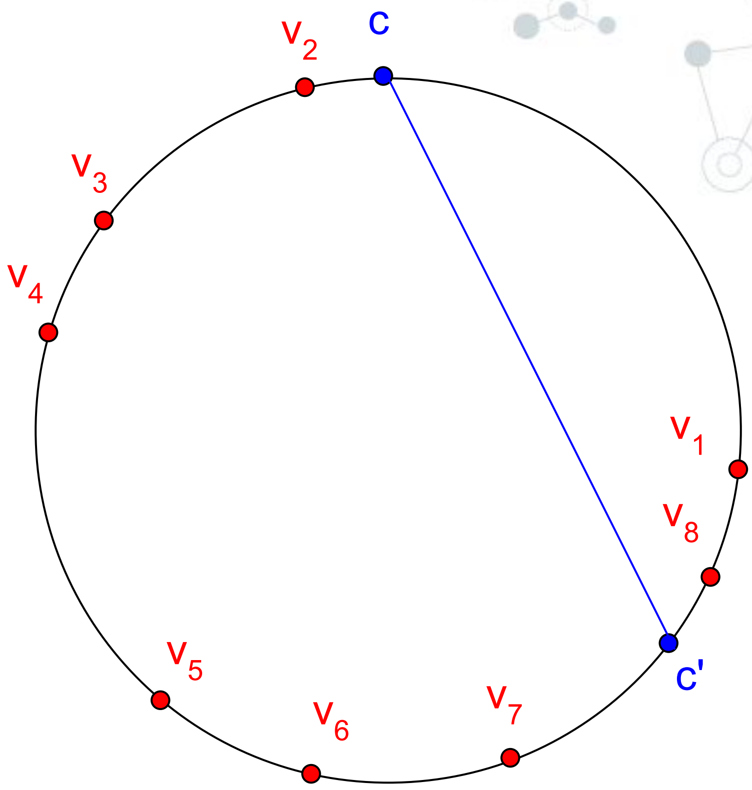
And voters sorted by angle.



A More General Property

Consider candidates c and c'.

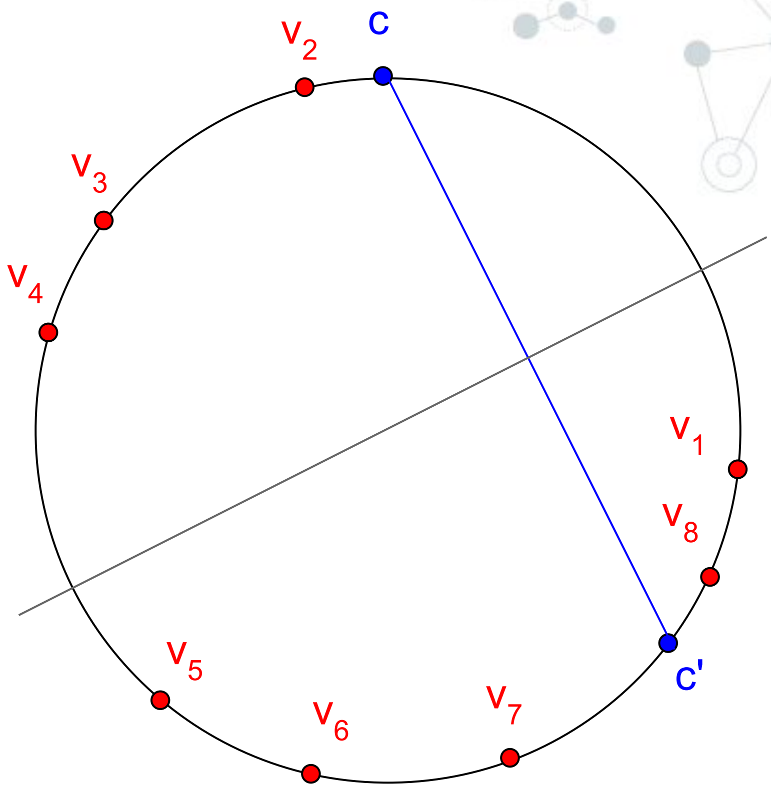
And voters sorted by angle.



A More General Property

Consider candidates c and c'.

And voters sorted by angle.

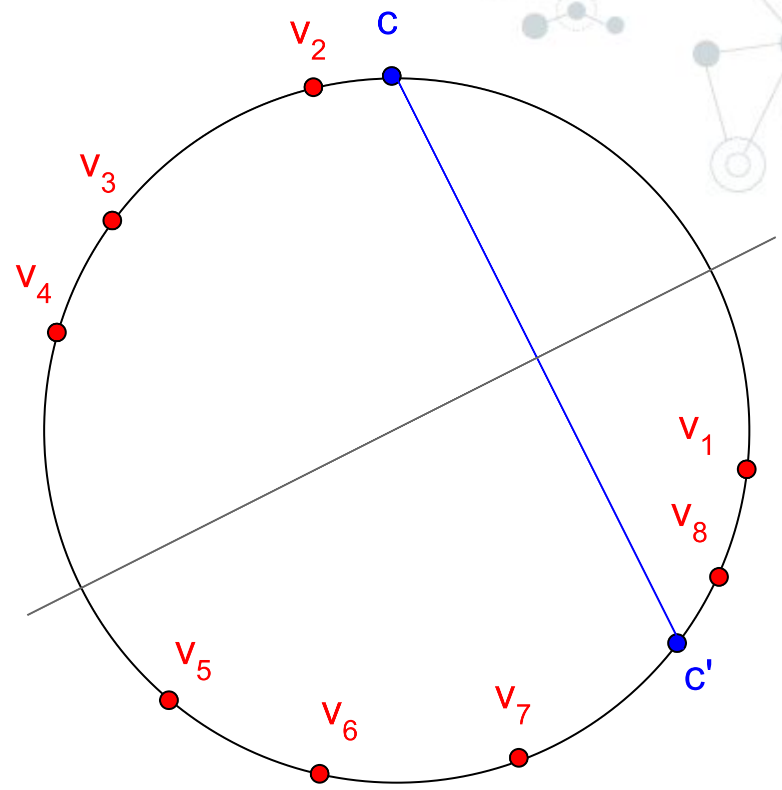


A More General Property

Consider candidates c and c'.

And voters sorted by angle.

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
$c \succ c'$	0	1	1	1	0	0	0	0

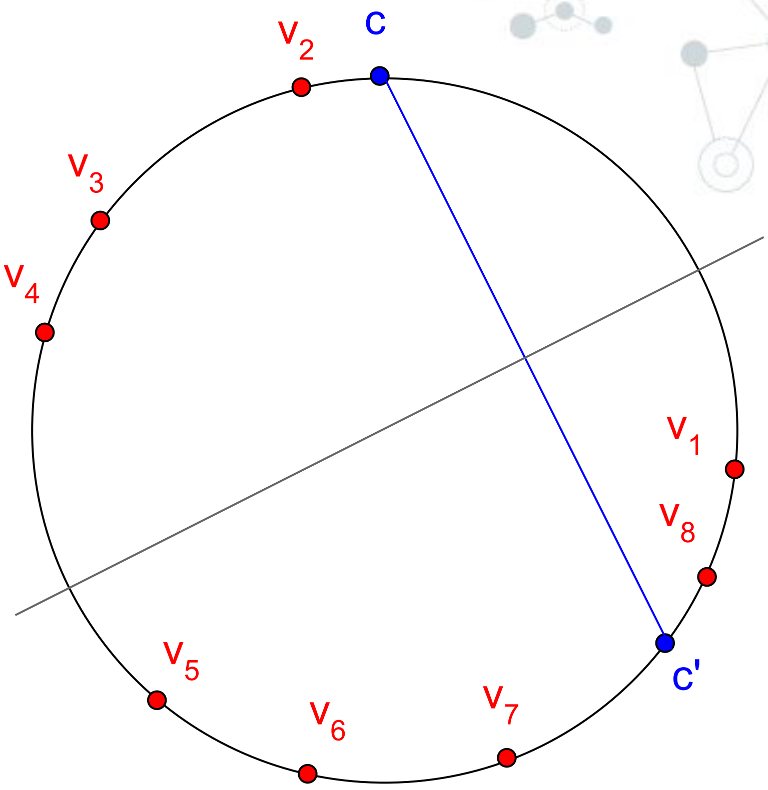


A More General Property

Consider candidates c and c'.

And voters sorted by angle.

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
c ≻ c'	0	1	1	1	0	0	0	0



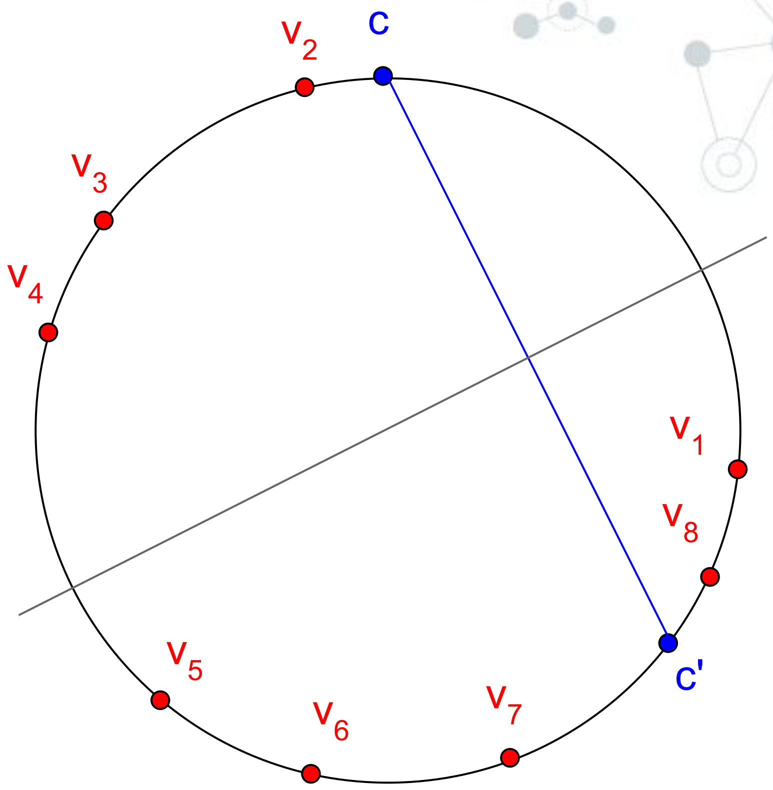
Voters preferring c to c'
form a "circular" interval!

A More General Property

Consider candidates c and c'.

And voters sorted by angle.

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
c ≻ c'	0	1	1	1	0	0	0	0
c' ≻ c	1	0	0	0	1	1	1	1



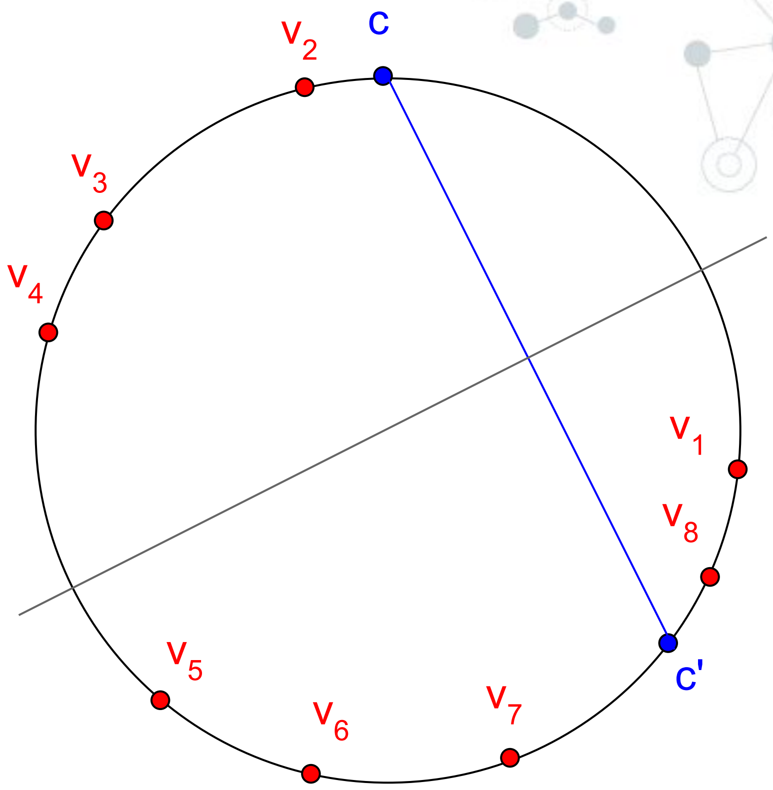
Voters preferring c to c' form a "circular" interval!

A More General Property

Consider candidates c and c'.

And voters sorted by angle.

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
c ≻ **c'**	0	1	1	1	0	0	0	0
c' ≻ **c**	1	0	0	0	1	1	1	1



Voters preferring c to c' form a "circular" interval!

⇔ At most 2 switches per row.

k-Crossing Elections

k-Crossing Elections

An election is *k-crossing* if voters can be reordered such that preference between any two candidates c, c' changes at most k times as we sweep through the voters in order:

k-Crossing Elections

An election is *k-crossing* if voters can be reordered such that preference between any two candidates c, c' changes at most k times as we sweep through the voters in order:

$v_1 : c_1 \succ c_2 \succ c_3$

$v_2 : c_3 \succ c_2 \succ c_1$

$v_3 : c_2 \succ c_3 \succ c_1$

$v_4 : c_3 \succ c_1 \succ c_2$

k-Crossing Elections

An election is *k-crossing* if voters can be reordered such that preference between any two candidates c, c' changes <u>at most k times</u> as we sweep through the voters in order:

$v_1 : c_1 \succ c_2 \succ c_3$

$v_2 : c_3 \succ c_2 \succ c_1$

$v_3 : c_2 \succ c_3 \succ c_1$

$v_4 : c_3 \succ c_1 \succ c_2$

	v_1	v_2	v_3	v_4
$c_1 \succ c_2$	1	0	0	1
$c_2 \succ c_3$	1	0	1	0
$c_1 \succ c_3$	1	0	0	0

k-Crossing Elections

An election is *k-crossing* if voters can be reordered such that preference between any two candidates c, c' changes <u>at most k times</u> as we sweep through the voters in order:

$v_1 : c_1 \succ c_2 \succ c_3$

$v_2 : c_3 \succ c_2 \succ c_1$

$v_3 : c_2 \succ c_3 \succ c_1$

$v_4 : c_3 \succ c_1 \succ c_2$

	v_1	v_2	v_3	v_4
$c_1 \succ c_2$	1	0	0	1
$c_2 \succ c_3$	1	0	1	0
$c_1 \succ c_3$	1	0	0	0

	v_1	v_3	v_2	v_4
$c_1 \succ c_2$	1	0	0	1
$c_2 \succ c_3$	1	1	0	0
$c_1 \succ c_3$	1	0	0	0

2.

Recognition

Using the Consecutive Ones Problem

Recognition

Recognition

Deciding whether an election is k-crossing.

Recognition

Deciding whether an election is k-crossing.

- Single-crossing: *poly-time* [Elkind et al., 2012; Bredereck et al., 2013].

Recognition

Deciding whether an election is k-crossing.

- Single-crossing: *poly-time*
  [Elkind et al., 2012; Bredereck et al., 2013].
- Two-crossing: *poly-time*
  Reduction to consecutive ones **(this paper)**.

Recognition

Deciding whether an election is k-crossing.

- Single-crossing: *poly-time*
  [Elkind et al., 2012; Bredereck et al., 2013].
- Two-crossing: *poly-time*
  Reduction to consecutive ones **(this paper)**.
- k-crossing: *open*
  We conjecture NP-complete for $k \geq 4$.

Recognition for Two-Crossing

Recognition for Two-Crossing

Given candidates $c_1, \ldots, c_M$ and voters $v_1, \ldots, v_N$

Recognition for Two-Crossing

Given candidates $c_1, \ldots, c_M$ and voters $v_1, \ldots, v_N$

$v_1 : c_1 \succ c_2 \succ c_3$

$v_2 : c_3 \succ c_2 \succ c_1$

$v_3 : c_2 \succ c_3 \succ c_1$

$v_4 : c_3 \succ c_1 \succ c_2$

Recognition for Two-Crossing

Given candidates $c_1, \ldots, c_M$ and voters $v_1, \ldots, v_N$, build matrix with rows indexed by pairs (c_i, c_j) with $i < j$

$v_1 : c_1 \succ c_2 \succ c_3$

$v_2 : c_3 \succ c_2 \succ c_1$

$v_3 : c_2 \succ c_3 \succ c_1$

$v_4 : c_3 \succ c_1 \succ c_2$

Recognition for Two-Crossing

Given candidates $c_1, \ldots, c_M$ and voters $v_1, \ldots, v_N$, build matrix with rows indexed by pairs (c_i, c_j) with $i < j$

$v_1 : c_1 \succ c_2 \succ c_3$

$v_2 : c_3 \succ c_2 \succ c_1$

$v_3 : c_2 \succ c_3 \succ c_1$

$v_4 : c_3 \succ c_1 \succ c_2$

Recognition for Two-Crossing

Given candidates $c_1, \ldots, c_M$ and voters $v_1, \ldots, v_N$, build matrix with rows indexed by pairs (c_i, c_j) with i < j

$c_1 \succ c_2$				
$c_2 \succ c_3$				
$c_1 \succ c_3$				

$v_1 : c_1 \succ c_2 \succ c_3$

$v_2 : c_3 \succ c_2 \succ c_1$

$v_3 : c_2 \succ c_3 \succ c_1$

$v_4 : c_3 \succ c_1 \succ c_2$

Recognition for Two-Crossing

Given candidates $c_1, \ldots, c_M$ and voters $v_1, \ldots, v_N$, build matrix with rows indexed by pairs (c_i, c_j) with $i < j$ and columns indexed by voters v_k.

$c_1 \succ c_2$				
$c_2 \succ c_3$				
$c_1 \succ c_3$				

$v_1 : c_1 \succ c_2 \succ c_3$

$v_2 : c_3 \succ c_2 \succ c_1$

$v_3 : c_2 \succ c_3 \succ c_1$

$v_4 : c_3 \succ c_1 \succ c_2$

Recognition for Two-Crossing

Given candidates $c_1, \dots, c_M$ and voters $v_1, \dots, v_N$, build matrix with rows indexed by pairs (c_i, c_j) with $i < j$ and columns indexed by voters v_k.

	v_1	v_2	v_3	v_4
$c_1 \succ c_2$				
$c_2 \succ c_3$				
$c_1 \succ c_3$				

$v_1 : c_1 \succ c_2 \succ c_3$

$v_2 : c_3 \succ c_2 \succ c_1$

$v_3 : c_2 \succ c_3 \succ c_1$

$v_4 : c_3 \succ c_1 \succ c_2$

Recognition for Two-Crossing

Given candidates $c_1, \ldots, c_M$ and voters $v_1, \ldots, v_N$, build matrix with rows indexed by pairs (c_i, c_j) with $i < j$ and columns indexed by voters v_k. Put a 1 at row (c_i, c_j), column v_k, iff v_k prefers c_i to c_j.

	v_1	v_2	v_3	v_4
$c_1 \succ c_2$				
$c_2 \succ c_3$				
$c_1 \succ c_3$				

$v_1 : c_1 \succ c_2 \succ c_3$

$v_2 : c_3 \succ c_2 \succ c_1$

$v_3 : c_2 \succ c_3 \succ c_1$

$v_4 : c_3 \succ c_1 \succ c_2$

Recognition for Two-Crossing

Given candidates $c_1, \ldots, c_M$ and voters $v_1, \ldots, v_N$, build matrix with rows indexed by pairs (c_i, c_j) with $i < j$ and columns indexed by voters v_k. Put a 1 at row (c_i, c_j), column v_k, iff v_k prefers c_i to c_j.

	v_1	v_2	v_3	v_4
$c_1 \succ c_2$	1	0	0	1
$c_2 \succ c_3$	1	0	1	0
$c_1 \succ c_3$	1	0	0	0

$v_1 : c_1 \succ c_2 \succ c_3$

$v_2 : c_3 \succ c_2 \succ c_1$

$v_3 : c_2 \succ c_3 \succ c_1$

$v_4 : c_3 \succ c_1 \succ c_2$

Recognition for Two-Crossing

Given candidates $c_1, \ldots, c_M$ and voters $v_1, \ldots, v_N$, build matrix with rows indexed by pairs (c_i, c_j) with $i < j$ and columns indexed by voters v_k. Put a 1 at row (c_i, c_j), column v_k, iff v_k prefers c_i to c_j.

Then, check whether columns can be permuted s.t. 1s in each row form a continuous **circular** run.

	v_1	v_2	v_3	v_4
$c_1 \succ c_2$	1	0	0	1
$c_2 \succ c_3$	1	0	1	0
$c_1 \succ c_3$	1	0	0	0

$v_1 : c_1 \succ c_2 \succ c_3$

$v_2 : c_3 \succ c_2 \succ c_1$

$v_3 : c_2 \succ c_3 \succ c_1$

$v_4 : c_3 \succ c_1 \succ c_2$

Recognition for Two-Crossing

Given candidates $c_1, \ldots, c_M$ and voters $v_1, \ldots, v_N$, build matrix with rows indexed by pairs (c_i, c_j) with $i < j$ and columns indexed by voters v_k. Put a 1 at row (c_i, c_j), column v_k, iff v_k prefers c_i to c_j.

Then, check whether columns can be permuted s.t. 1s in each row form a continuous **circular** run.

[Booth and Lueker, 1976]

	v_1	v_2	v_3	v_4
$c_1 \succ c_2$	1	0	0	1
$c_2 \succ c_3$	1	0	1	0
$c_1 \succ c_3$	1	0	0	0

$v_1 : c_1 \succ c_2 \succ c_3$

$v_2 : c_3 \succ c_2 \succ c_1$

$v_3 : c_2 \succ c_3 \succ c_1$

$v_4 : c_3 \succ c_1 \succ c_2$

Recognition for Two-Crossing

Given candidates $c_1, \ldots, c_M$ and voters $v_1, \ldots, v_N$, build matrix with rows indexed by pairs (c_i, c_j) with $i < j$ and columns indexed by voters v_k. Put a 1 at row (c_i, c_j), column v_k, iff v_k prefers c_i to c_j.

Then, check whether columns can be permuted s.t. 1s in each row form a continuous **circular** run.

[Booth and Lueker, 1976]

$v_1 : c_1 \succ c_2 \succ c_3$

$v_2 : c_3 \succ c_2 \succ c_1$

$v_3 : c_2 \succ c_3 \succ c_1$

$v_4 : c_3 \succ c_1 \succ c_2$

	v_1	v_2	v_3	v_4
$c_1 \succ c_2$	1	0	0	1
$c_2 \succ c_3$	1	0	1	0
$c_1 \succ c_3$	1	0	0	0

	v_1	v_3	v_2	v_4
$c_1 \succ c_2$	1	**0**	**0**	1
$c_2 \succ c_3$	1	**1**	**0**	0
$c_1 \succ c_3$	1	**0**	**0**	0

Recognition for Two-Crossing

Given candidates $c_1, \ldots, c_M$ and voters $v_1, \ldots, v_N$, build matrix with rows indexed by pairs (c_i, c_j) with $i < j$ and columns indexed by voters v_k. Put a 1 at row (c_i, c_j), column v_k, iff v_k prefers c_i to c_j.

Then, check whether columns can be permuted s.t. 1s in each row form a continuous **circular** run.

[Booth and Lueker, 1976]

$v_1 : c_1 > c_2 > c_3$

$v_2 : c_3 > c_2 > c_1$

$v_3 : c_2 > c_3 > c_1$

$v_4 : c_3 > c_1 > c_2$

$O(NM^2)$

	v_1	v_2	v_3	v_4
$c_1 > c_2$	1	0	0	1
$c_2 > c_3$	1	0	1	0
$c_1 > c_3$	1	0	0	0

	v_1	v_3	v_2	v_4
$c_1 > c_2$	1	**0**	**0**	1
$c_2 > c_3$	1	**1**	**0**	0
$c_1 > c_3$	1	**0**	**0**	0

3.

Majority Tournament Universality

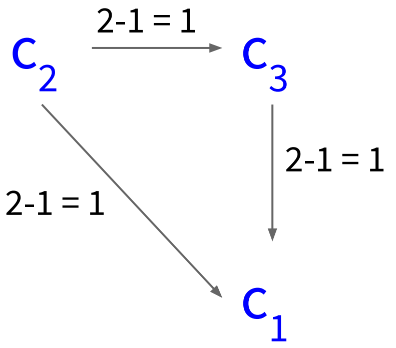
And NP-Hardness of Kemeny

Weighted Majority Tournament

Two-crossing: **also any (weighted) tournament can be obtained!**

Weighted Majority Tournament

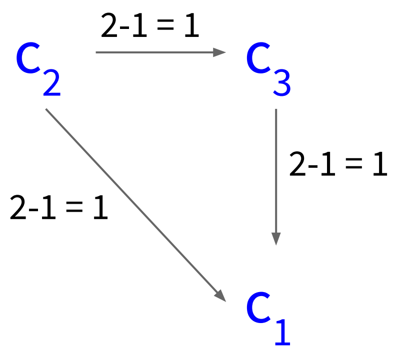
Two-crossing: **also any (weighted) tournament can be obtained!**



Weighted Majority Tournament

Single-crossing: tournament is transitive.

Two-crossing: **also any (weighted) tournament can be obtained!**



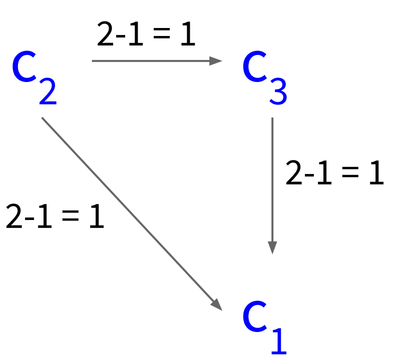
Weighted Majority Tournament

Single-crossing: tournament is transitive.

Two-crossing: **also any (weighted) tournament can be obtained!**

General elections: any (weighted) tournament can be obtained.

[McGarvey, 1953; Debord, 1987]



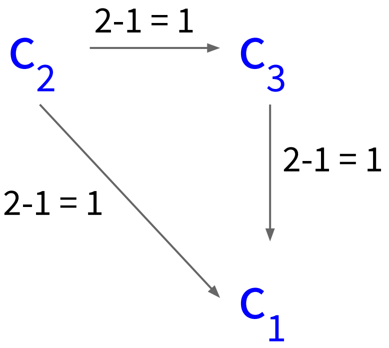
Weighted Majority Tournament

Single-crossing: tournament is transitive.

Two-crossing: **also any (weighted) tournament can be obtained!**

General elections: any (weighted) tournament can be obtained.

[McGarvey, 1953; Debord, 1987]



Proof

Proof

Construct the **“Double-BubbleSort”** profile.

Proof

Construct the **“Double-BubbleSort”** profile.

e.g. M = 4 candidates.

Proof

Construct the **"Double-BubbleSort"** profile.

e.g. M = 4 candidates.

v_1
1
2
3
4

Proof

Construct the **“Double-BubbleSort”** profile.

e.g. M = 4 candidates.

v_1	v_2
1	2
2	1
3	3
4	4

Proof

Construct the **“Double-BubbleSort”** profile.

e.g. M = 4 candidates.

v_1	v_2	v_3
1	2	2
2	1	3
3	3	1
4	4	4

Proof

Construct the **“Double-BubbleSort”** profile.

e.g. M = 4 candidates.

v_1	v_2	v_3	v_4
1	2	2	2
2	1	3	3
3	3	1	4
4	4	4	1

Proof

Construct the **“Double-BubbleSort”** profile.

e.g. M = 4 candidates.

v_1	v_2	v_3	v_4	v_5
1	2	2	2	3
2	1	3	3	2
3	3	1	4	4
4	4	4	1	1

Proof

Construct the **“Double-BubbleSort”** profile.

e.g. M = 4 candidates.

v_1	v_2	v_3	v_4	v_5	v_6
1	2	2	2	3	3
2	1	3	3	2	4
3	3	1	4	4	2
4	4	4	1	1	1

Proof

Construct the **“Double-BubbleSort”** profile.

e.g. M = 4 candidates.

v_1	v_2	v_3	v_4	v_5	v_6	v_7
1	2	2	2	3	3	4
2	1	3	3	2	4	3
3	3	1	4	4	2	2
4	4	4	1	1	1	1

Proof

Construct the **“Double-BubbleSort”** profile.

e.g. M = 4 candidates.

v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
1	2	2	2	3	3	4	4
2	1	3	3	2	4	3	3
3	3	1	4	4	2	2	1
4	4	4	1	1	1	1	2

Proof

Construct the **“Double-BubbleSort”** profile.

e.g. M = 4 candidates.

v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9
1	2	2	2	3	3	4	4	4
2	1	3	3	2	4	3	3	1
3	3	1	4	4	2	2	1	3
4	4	4	1	1	1	1	2	2

Proof

Construct the **“Double-BubbleSort”** profile.

e.g. M = 4 candidates.

v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}
1	2	2	2	3	3	4	4	4	1
2	1	3	3	2	4	3	3	1	4
3	3	1	4	4	2	2	1	3	3
4	4	4	1	1	1	1	2	2	2

Proof

Construct the **“Double-BubbleSort”** profile.

e.g. M = 4 candidates.

v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}
1	2	2	2	3	3	4	4	4	1	1
2	1	3	3	2	4	3	3	1	4	4
3	3	1	4	4	2	2	1	3	3	2
4	4	4	1	1	1	1	2	2	2	3

Proof

Construct the **"Double-BubbleSort"** profile.

e.g. M = 4 candidates.

v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}
1	2	2	2	3	3	4	4	4	1	1	1
2	1	3	3	2	4	3	3	1	4	4	2
3	3	1	4	4	2	2	1	3	3	2	4
4	4	4	1	1	1	1	2	2	2	3	3

Proof

Construct the **"Double-BubbleSort"** profile.

e.g. M = 4 candidates.

v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}
1	2	2	2	3	3	4	4	4	1	1	1	1
2	1	3	3	2	4	3	3	1	4	4	2	2
3	3	1	4	4	2	2	1	3	3	2	4	3
4	4	4	1	1	1	1	2	2	2	3	3	4

Proof

Construct the **“Double-BubbleSort”** profile.

e.g. M = 4 candidates.

v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}
1	2	2	2	3	3	4	4	4	1	1	1	1
2	1	3	3	2	4	3	3	1	4	4	2	2
3	3	1	4	4	2	2	1	3	3	2	4	3
4	4	4	1	1	1	1	2	2	2	3	3	4

This profile is two-crossing!

Proof

Construct the **"Double-BubbleSort"** profile.

e.g. M = 4 candidates.

v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}
1	2	2	2	3	3	4	4	4	1	1	1	1
2	1	3	3	2	4	3	3	1	4	4	2	2
3	3	1	4	4	2	2	1	3	3	2	4	3
4	4	4	1	1	1	1	2	2	2	3	3	4

$1 \xrightarrow{1} 3$

This profile is two-crossing!

Proof

Construct the **“Double-BubbleSort”** profile.

e.g. M = 4 candidates.

v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}
1	2	2	2	3	3	4	4	4	1	1	1	1
2	1	3	3	2	4	3	3	1	4	4	2	2
3	3	1	4	4	2	2	1	3	3	2	4	3
4	4	4	1	1	1	1	2	2	2	3	3	4

$1 \xrightarrow{1} 3$

This profile is two-crossing!

Proof

Construct the **“Double-BubbleSort”** profile.

e.g. M = 4 candidates.

v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}
1	2	2	2	3	3	4	4	4	1	1	1	1
2	1	3	3	2	4	3	3	1	4	4	2	2
3	3	1	4	4	2	2	1	3	3	2	4	3
4	4	4	1	1	1	1	2	2	2	3	3	4

$1 \xrightarrow{1} 3$

This profile is two-crossing!

Proof

Construct the **“Double-BubbleSort”** profile.

e.g. M = 4 candidates.

v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}
1	2	2	2	3	3	4	4	4	1	1	1	1
2	1	3	3	2	4	3	3	1	4	4	2	2
3	3	1	4	4	2	2	1	3	3	2	4	3
4	4	4	1	1	1	1	2	2	2	3	3	4

V

$1 \xrightarrow{1} 3$

This profile is two-crossing!

Proof

Construct the **“Double-BubbleSort”** profile.

e.g. M = 4 candidates.

v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}
1	2	2	2	3	3	4	4	4	1	1	1	1
2	1	3	3	2	4	3	3	1	4	4	2	2
3	3	1	4	4	2	2	1	3	3	2	4	3
4	4	4	1	1	1	1	2	2	2	3	3	4

V

V'

$1 \xrightarrow{1} 3$

This profile is two-crossing!

Proof

Construct the **“Double-BubbleSort”** profile.

e.g. M = 4 candidates.

v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}
1	2	2	2	3	3	4	4	4	1	1	1	1
2	1	3	3	2	4	3	3	1	4	4	2	2
3	3	1	4	4	2	2	1	3	3	2	4	3
4	4	4	1	1	1	1	2	2	2	3	3	4

v

v'

$1 \xrightarrow{1} 3$

This profile is two-crossing!

Proof

Construct the **"Double-BubbleSort"** profile.

e.g. M = 4 candidates.

v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}
1	2	2	2	3	3	4	4	4	1	1	1	1
2	1	3	3	2	4	3	3	1	4	4	2	2
3	3	1	4	4	2	2	1	3	3	2	4	3
4	4	4	1	1	1	1	2	2	2	3	3	4

V

V'

$1 \xrightarrow{3} 3$

This profile is two-crossing!

Consequences: NP-hardness

Consequences: NP-hardness

Thus, NP-hardness results carry over to two-crossing:

Consequences: NP-hardness

Thus, NP-hardness results carry over to two-crossing:

- **Kemeny and Slater are NP-hard.**

Consequences: NP-hardness

Thus, NP-hardness results carry over to two-crossing:

- **Kemeny and Slater are NP-hard.**
- Banks, Minimal Extending Set, Tournament Equilibrium Set and Ranked Pairs also NP-hard.

4.

Young's Rule

Using Total Unimodularity

Young's Rule

Young's Rule

The Young score of candidate c is the least number of voters that need to be removed to make c a Condorcet winner.

Young's Rule

The Young score of candidate c is the least number of voters that need to be removed to make c a Condorcet winner. Winners are candidates with the least score.

Young's Rule

The Young score of candidate c is the least number of voters that need to be removed to make c a Condorcet winner. Winners are candidates with the least score.

- NP-hard in general:
  [Rothe et al., 2003; Brandt et al., 2015; Fitzsimmons and Hemaspaandra, 2020].

Young's Rule

The Young score of candidate c is the least number of voters that need to be removed to make c a Condorcet winner. Winners are candidates with the least score.

- NP-hard in general: [Rothe et al., 2003; Brandt et al., 2015; Fitzsimmons and Hemaspaandra, 2020].
- Two-crossing: **scores in poly-time (this paper)**.

Young's Rule

The natural LP does not have integer vertices.

By fixing the number of voters to keep we arrive at an LP with integer vertices, so we can solve the LP.

Young's Rule

The natural LP does not have integer vertices.

By fixing the number of voters to keep we arrive at an LP with integer vertices, so we can solve the LP.

Young's Rule

The natural LP does not have integer vertices.

By fixing the number of voters to keep we arrive at an LP with integer vertices, so we can solve the LP.

Young's Rule

The natural LP does not have integer vertices.

By fixing the number of voters to keep we arrive at an LP with integer vertices, so we can solve the LP.

By reducing to negative weight cycle detection we further **improve the running time to $O((n + m^2)n^{3/2} \log n)$**.

5.

Chamberlin-Courant Rule

Using Dynamic Programming

Representation

Representation

In an election we need to select a committee of K candidates to best represent the electorate.

Representation

In an election we need to select a committee of K candidates to best represent the electorate.

v_1 : Blue > Yellow > Red > Pink > Green

v_2 : Yellow > Green > Red > Pink > Blue

v_3 : Green > Red > Blue > Pink > Yellow

Representation

In an election we need to select a committee of K candidates to best represent the electorate.

e.g. **K = 2**

v_1 : Blue > Yellow > Red > Pink > Green

v_2 : Yellow > Green > Red > Pink > Blue

v_3 : Green > Red > Blue > Pink > Yellow

Representation

In an election we need to select a committee of K candidates to best represent the electorate.

e.g. **K = 2**

v_1	:		>	Yellow	>		>	Pink	>	
v_2	:	Yellow	>		>		>	Pink	>	
v_3	:		>		>		>	Pink	>	Yellow

Representation

In an election we need to select a committee of K candidates to best **represent** the electorate.

e.g. **K = 2**

v_1 :		>	$\boxed{\texttt{Yellow}}$	>		>	Pink	>	
v_2 :	$\boxed{\texttt{Yellow}}$	>		>		>	Pink	>	
v_3 :		>		>		>	$\boxed{\texttt{Pink}}$	>	Yellow

Representation

In an election we need to select a committee of K candidates to best **represent** the electorate.

e.g. **K = 2**

v_1 : > Yellow > > Pink >

v_2 : Yellow > > > Pink >

v_3 : > > > Pink > Yellow

Q: How to compare K-committees?

The Chamberlin-Courant Rule

The Chamberlin-Courant Rule

Voters specify their *dissatisfaction* with each candidate.

The Chamberlin-Courant Rule

Voters specify their *dissatisfaction* with each candidate.

	0	1	5	8	9
v_1 :	Blue >	Yellow >	Red >	Pink >	Green
	0	3	3	4	8
v_2 :	Yellow >	Green >	Red >	Pink >	Blue
	0	1	1	2	3
v_3 :	Green >	Red >	Blue >	Pink >	Yellow

The Chamberlin-Courant Rule

Voters specify their *dissatisfaction* with each candidate.

Pick the K-committee that **minimizes** the total/maximum dissatisfaction.

		0		1		5		8		9
v_1	:	Blue	>	Yellow	>	Red	>	Pink	>	Green
		0		3		3		4		8
v_2	:	Yellow	>	Green	>	Red	>	Pink	>	Blue
		0		1		1		2		3
v_3	:	Green	>	Red	>	Blue	>	Pink	>	Yellow

The Chamberlin-Courant Rule

Voters specify their *dissatisfaction* with each candidate.

Pick the K-committee that **minimizes** the total/maximum dissatisfaction.

				1				8		3	
v_1	:		>	Yellow	>		>	Pink	>		
		0						4			
v_2	:	Yellow	>		>		>	Pink	>		
								2		3	
v_3	:		>		>		>	Pink	>	Yellow	

The Chamberlin-Courant Rule

Voters specify their *dissatisfaction* with each candidate.

Pick the K-committee that **minimizes** the total/maximum dissatisfaction.

v_1 :		>	$\boxed{\overset{1}{\text{Yellow}}}$	>		>	$\overset{8}{\text{Pink}}$	>	
v_2 :	$\boxed{\overset{0}{\text{Yellow}}}$	>		>		>	$\overset{4}{\text{Pink}}$	>	
v_3 :		>		>		>	$\boxed{\overset{2}{\text{Pink}}}$	>	$\overset{3}{\text{Yellow}}$

Total = **3** (Utilitarian-CC) **- in this talk.**

The Chamberlin-Courant Rule

Voters specify their *dissatisfaction* with each candidate.

Pick the K-committee that **minimizes** the total/maximum dissatisfaction.

				1				8		
v_1	:		>	Yellow	>		>	Pink	>	
		0						4		
v_2	:	Yellow	>		>		>	Pink	>	
								2		3
v_3	:		>		>		>	Pink	>	Yellow

Total = **3** (Utilitarian-CC) **- in this talk.**

Maximum = **2** (Egalitarian-CC) [Betzler et al.; 2013]

Hardness of CC

Hardness of CC

Utilitarian-CC is NP-hard.

[Procaccia et al., 2008], [Lu and Boutilier, 2011]

Hardness of CC

Utilitarian-CC is NP-hard.

[Procaccia et al., 2008], [Lu and Boutilier, 2011]

Egalitarian-CC is NP-hard.

[Betzler et al., 2013]

Hardness of CC

Utilitarian-CC is NP-hard.

[Procaccia et al., 2008], [Lu and Boutilier, 2011]

Egalitarian-CC is NP-hard.

[Betzler et al., 2013]

Egalitarian-CC is NP-hard for three-crossing.

[Misra et al., 2017]

Hardness of CC

Utilitarian-CC is NP-hard.

[Procaccia et al., 2008], [Lu and Boutilier, 2011]

Egalitarian-CC is NP-hard.

[Betzler et al., 2013]

Egalitarian-CC is NP-hard for three-crossing.

[Misra et al., 2017]

Both polynomial for single-crossing.

[Skowron et al., 2015], [Constantinescu and Elkind, 2021]

Hardness of CC

Utilitarian-CC is NP-hard.

[Procaccia et al., 2008], [Lu and Boutilier, 2011]

Egalitarian-CC is NP-hard.

[Betzler et al., 2013]

Egalitarian-CC is NP-hard for three-crossing.

[Misra et al., 2017]

Both polynomial for single-crossing.

[Skowron et al., 2015], [Constantinescu and Elkind, 2021]

Both polynomial for two-crossing (this paper).

Preliminaries

Preliminaries

Say voters $v_1, \ldots, v_N$ are in a two-crossing order.

Preliminaries

Say voters $v_1, \ldots, v_N$ are in a two-crossing order.

Let $r : \{v_1, \ldots, v_N\} \rightarrow \{c_1, \ldots, c_M\}$ be the function mapping voters to representatives in an optimal CC committee.

Preliminaries

Say voters $v_1, \ldots, v_N$ are in a two-crossing order.

Let $r : \{v_1, \ldots, v_N\} \to \{c_1, \ldots, c_M\}$ be the function mapping voters to representatives in an optimal CC committee.

v	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
r(v)	B	R	R	Y	R	P	P	G

Decomposition For Two-Crossing

Decomposition For Two-Crossing

v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}
G	R	B	O	B	R	P	P	R	Y

Decomposition For Two-Crossing

v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}
G	R	B	O	B	R	P	P	R	Y

R splits

Decomposition For Two-Crossing

v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}
G	R	B	O	B	R	P	P	R	Y

R splits

v_1
G

Decomposition For Two-Crossing

v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}
G	R	B	O	B	R	P	P	R	Y

R splits

v_1
G

v_3	v_4	v_5
B	O	B

Decomposition For Two-Crossing

v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}
G	R	B	O	B	R	P	P	R	Y

R splits

v_1
G

v_3	v_4	v_5
B	O	B

v_7	v_8
P	P

Decomposition For Two-Crossing

v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}
G	R	B	O	B	R	P	P	R	Y

R splits

v_1	v_3	v_4	v_5	v_7	v_8	v_{10}
G	B	O	B	P	P	Y

Decomposition For Two-Crossing

v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}
G	R	B	O	B	R	P	P	R	Y

R splits

v_1		v_3	v_4	v_5		v_7	v_8		v_{10}
G		B	O	B		P	P		Y

B splits

Decomposition For Two-Crossing

v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}
G	R	B	O	B	R	P	P	R	Y

R splits

v_1
G

v_3	v_4	v_5
B	O	B

v_7	v_8
P	P

v_{10}
Y

B splits

v_4
O

Decomposition For Two-Crossing

v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}
G	R	B	O	B	R	P	P	R	Y

R splits

v_1
G

v_3	v_4	v_5
B	O	B

v_7	v_8
P	P

v_{10}
Y

B splits

v_4
O

There exists a decomposable optimal committee!

Future Directions

Future Directions

1. Try two-crossing on PrefLib.

Future Directions

1. Try two-crossing on PrefLib.

2. Hardness of recognizing k-crossigness.

Future Directions

1. Try two-crossing on PrefLib.
2. Hardness of recognizing k-crossigness.
3. Hardness of Dodgson's rule for two-crossing.

Future Directions

1. Try two-crossing on PrefLib.
2. Hardness of recognizing k-crossigness.
3. Hardness of Dodgson's rule for two-crossing.
4. Hardness of Young's rule for three-crossing.

Future Directions

1. Try two-crossing on PrefLib.
2. Hardness of recognizing k-crossigness.
3. Hardness of Dodgson's rule for two-crossing.
4. Hardness of Young's rule for three-crossing.
5. Three-crossing and above in general?

Hope you enjoyed!