Proportional Representation under Single-Crossing Preferences Revisited

Andrei Constantinescu
Edith Elkind

University of Oxford

1.

Framework

Multiwinner Voting &
The Chamberlin-Courant Rule

Framework

Framework

In an election N voters vote for M candidates.

Framework

In an election N voters vote for M candidates.

Voters express preference by ordering candidates.

Framework

In an election N voters vote for M candidates.

Voters express preference by ordering candidates.

e.g. **N = 3**, **M = 5**:

Framework

In an election N voters vote for M candidates.

Voters express preference by ordering candidates.

e.g. **N = 3**, **M = 5**:

```
V1 : Blue   > Yellow > Red  > Pink > Green
V2 : Yellow > Green  > Red  > Pink > Blue
V3 : Green  > Red    > Blue > Pink > Yellow
```

Framework

In an election N voters vote for M candidates.

Voters express preference by ordering candidates.

e.g. **N = 3**, **M = 5**:

```
V1 : Blue   > Yellow > Red  > Pink > Green
V2 : Yellow > Green  > Red  > Pink > Blue
V3 : Green  > Red    > Blue > Pink > Yellow
```

(preference profile)

Multiwinner Voting Rules

Multiwinner Voting Rules

Given a preference profile we need to select a committee of K candidates to represent the electorate.

Multiwinner Voting Rules

Given a preference profile we need to select a committee of K candidates to represent the electorate.

e.g. **K = 2**

Multiwinner Voting Rules

Given a preference profile we need to select a committee of K candidates to represent the electorate.

e.g. **K = 2**

```
V1 : Blue   > Yellow > Red   > Pink > Green
V2 : Yellow > Green  > Red   > Pink > Blue
V3 : Green  > Red    > Blue  > Pink > Yellow
```

Multiwinner Voting Rules

Given a preference profile we need to select a committee of K candidates to represent the electorate.

e.g. **K = 2**

V1 : > Yellow > > Pink >

V2 : Yellow > > > Pink >

V3 : > > > Pink > Yellow

Multiwinner Voting Rules

Given a preference profile we need to select a committee of K candidates to represent the electorate.

e.g. **K = 2**

```
V1 :        > Yellow >        > Pink >
V2 : Yellow >        >        > Pink >
V3 :        >        >        > Pink > Yellow
```

Multiwinner Voting Rules

Given a preference profile we need to select a committee of K candidates to represent the electorate.

e.g. **K = 2**

```
V1 :        > Yellow >        > Pink >
V2 : Yellow >        >        > Pink >
V3 :        >        >        > Pink > Yellow
```

Q: How do we pick the K-committee?

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.

Need to pick the K-committee that **minimizes** the total/maximum dissatisfaction.

The Chamberlin-Courant Rule

Voters specify their *dissatisfaction* with each candidate.

Need to pick the K-committee that **minimizes** the total/maximum dissatisfaction.

V1 : Blue > Yellow > Red > Pink > Green

V2 : Yellow > Green > Red > Pink > Blue

V3 : Green > Red > Blue > Pink > Yellow

The Chamberlin-Courant Rule

Voters specify their *dissatisfaction* with each candidate.

Need to pick the K-committee that **minimizes** the total/maximum dissatisfaction.

V1 :	Blue (0)	>	Yellow (1)	>	Red (5)	>	Pink (8)	>	Green (9)
V2 :	Yellow (0)	>	Green (3)	>	Red (3)	>	Pink (4)	>	Blue (8)
V3 :	Green (0)	>	Red (1)	>	Blue (1)	>	Pink (2)	>	Yellow (3)

The Chamberlin-Courant Rule

Voters specify their *dissatisfaction* with each candidate.

Need to pick the K-committee that **minimizes** the total/maximum dissatisfaction.

```
                1                8
V1 :       >  Yellow  >       >  Pink  >
       0                         4
V2 : Yellow  >        >       >  Pink  >
                                 2         3
V3 :       >          >       >  Pink  >  Yellow
```

The Chamberlin-Courant Rule

Voters specify their *dissatisfaction* with each candidate.

Need to pick the K-committee that **minimizes** the total/maximum dissatisfaction.

V1	:		>	1 Yellow	>		>	8 Pink	>	
V2	:	0 Yellow	>		>		>	4 Pink	>	
V3	:		>		>		>	2 Pink	>	3 Yellow

The Chamberlin-Courant Rule

Voters specify their *dissatisfaction* with each candidate.

Need to pick the K-committee that **minimizes** the total/maximum dissatisfaction.

V1 :		>	Yellow (1)	>		>	Pink (8)	>	
V2 :	Yellow (0)	>		>		>	Pink (4)	>	
V3 :		>		>		>	Pink (2)	>	Yellow (3)

Total = **3** (Utilitarian-CC)

The Chamberlin-Courant Rule

Voters specify their *dissatisfaction* with each candidate.

Need to pick the K-committee that **minimizes** the total/maximum dissatisfaction.

V1 :		>	Yellow (1)	>		>	Pink (8)	>	
V2 :	Yellow (0)	>		>		>	Pink (4)	>	
V3 :		>		>		>	Pink (2)	>	Yellow (3)

Total = **3** (Utilitarian-CC)

Maximum = **2** (Egalitarian-CC) [Betzler, Slinko, Uhlmann'13]

The Chamberlin-Courant Rule

Voters specify their *dissatisfaction* with each candidate.

Need to pick the K-committee that **minimizes** the total/maximum dissatisfaction.

```
V1 :        > Yellow(1) >        > Pink(8) >
V2 : Yellow(0) >        >        > Pink(4) >
V3 :        >        >        > Pink(2) > Yellow(3)
```

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

Maximum = **2** (Egalitarian-CC) [Betzler, Slinko, Uhlmann'13]

Hardness of CC

Hardness of CC

Utilitarian-CC is NP-hard

[Procaccia, Rosenschein, Zohar'08]

[Lu, Boutilier'11]

Egalitarian-CC is NP-hard

[Betzler, Slinko, Uhlmann'13]

A way out!

A way out!

Real elections have more structure, making CC easier!

A way out!

Real elections have more structure, making CC easier! We consider ***single-crossing*** preferences.

[Roberts'77, Mirrlees'71]

2.

Structured Preferences

Single-crossing Preferences & Intermediate Preferences on Median Graphs

Single-crossing Preferences

Single-crossing Preferences

A profile is *single-crossing* if we can order the voters so that preference between any two candidates a, b changes at most once as we go through the candidates in order:

Single-crossing Preferences

A profile is *single-crossing* if we can order the voters so that preference between any two candidates a, b changes at most once as we go through the candidates in order:

V_1 : Blue > Yellow

V_2 : Blue > Yellow

V_3 : Yellow > Blue

V_4 : Yellow > Blue

Single-crossing Preferences

A profile is *single-crossing* if we can order the voters so that preference between any two candidates a, b changes at most once as we go through the candidates in order:

V_1 : Blue > Yellow

V_2 : Blue > Yellow

↔

V_3 : Yellow > Blue

V_4 : Yellow > Blue

Single-crossing Preferences

A profile is *single-crossing* if we can order the voters so that preference between any two candidates a, b changes at most once as we go through the candidates in order:

V_1 : Blue > Yellow

V_2 : Blue > Yellow

V_3 : Yellow > Blue

V_4 : Yellow > Blue

✓ ✗

V_1 : Blue > Yellow

V_2 : Blue > Yellow

V_3 : Yellow > Blue

V_4 : Blue > Yellow

Single-crossing Preferences

A profile is *single-crossing* if we can order the voters so that preference between any two candidates a, b changes <u>at most once</u> as we go through the candidates in order:

V_1 : Blue > Yellow
V_2 : Blue > Yellow
V_3 : Yellow > Blue
V_4 : Yellow > Blue

✓

✗

V_1 : Blue > Yellow
V_2 : Blue > Yellow
V_3 : Yellow > Blue
V_4 : Blue > Yellow

Some Properties of SC

Some Properties of SC

- Majority relation is acyclic, so Condorcet winner exists.*

*For odd n.

Some Properties of SC

- Majority relation is acyclic, so Condorcet winner exists.*
- Profiles admit a median voter. [Rothstein'91]

*For odd n.

Some Properties of SC

- Majority relation is acyclic, so Condorcet winner exists.*
- Profiles admit a median voter. [Rothstein'91]
- CC can be solved in polynomial time. [Skowron et al.'15]

*For odd n.

Some Properties of SC

- Majority relation is acyclic, so Condorcet winner exists.*
- Profiles admit a median voter. [Rothstein'91]
- CC can be solved in polynomial time. [Skowron et al.'15]

Problem: Not many real elections are SC.

*For odd n.

Some Properties of SC

- Majority relation is acyclic, so Condorcet winner exists.*
- Profiles admit a median voter. [Rothstein'91]
- CC can be solved in polynomial time. [Skowron et al.'15]

Problem: Not many real elections are SC. Extend notion?

*For odd n.

Some Properties of SC

- Majority relation is acyclic, so Condorcet winner exists.*
- Profiles admit a median voter. [Rothstein'91]
- CC can be solved in polynomial time. [Skowron et al.'15]

Problem: Not many real elections are SC. Extend notion?

Difficulty: Preserve Condorcet domain and poly-time solvability of CC.

*For odd n.

Generalized SC

Generalized SC

[Demange'12] introduces *intermediate preferences indexed by a* ***median graph***.

Generalized SC

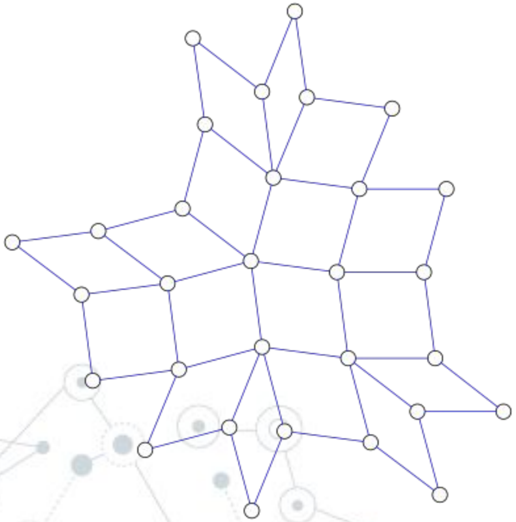
[Demange'12] introduces *intermediate preferences indexed by a* ***median graph***.

[Puppe, Slinko'17] show this is necessary and sufficient to get a Condorcet domain.

Generalized SC

[Demange'12] introduces *intermediate preferences indexed by a* ***median graph***.

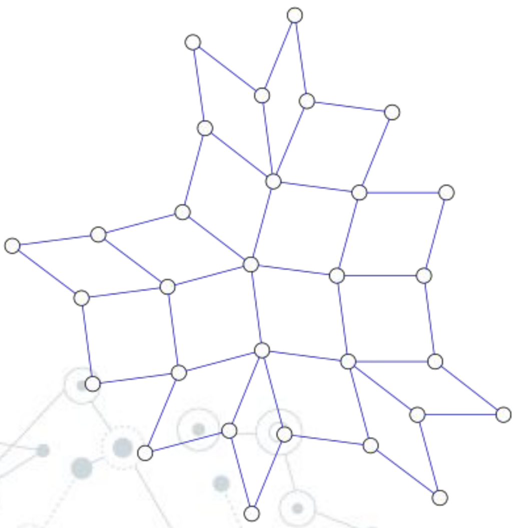
[Puppe, Slinko'17] show this is necessary and sufficient to get a Condorcet domain.

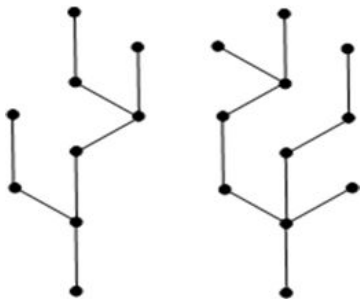


Generalized SC

[Demange'12] introduces *intermediate preferences indexed by a* ***median graph***.

[Puppe, Slinko'17] show this is necessary and sufficient to get a Condorcet domain.

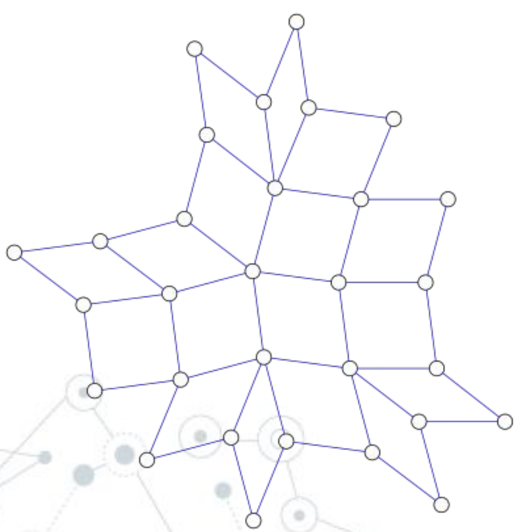


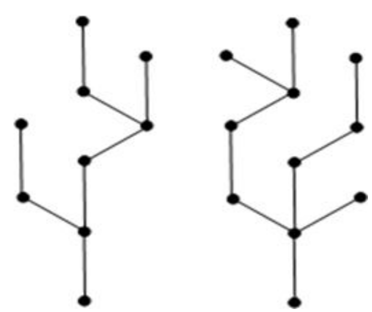


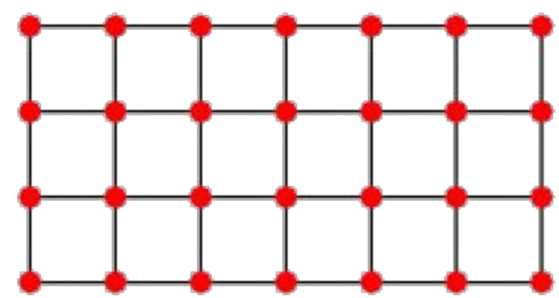
Generalized SC

[Demange'12] introduces *intermediate preferences indexed by a* ***median graph***.

[Puppe, Slinko'17] show this is necessary and sufficient to get a Condorcet domain.





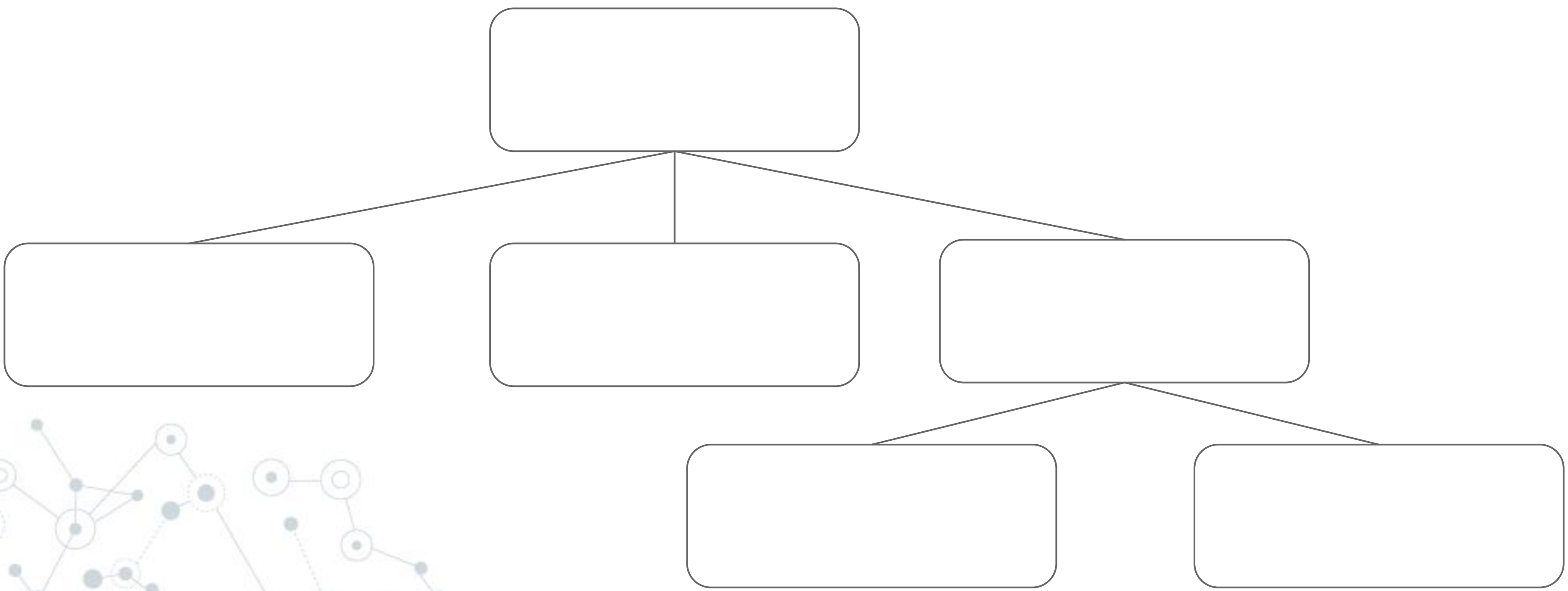


Generalized SC

A preference profile is *single-crossing with respect to a median graph* having the voters as vertices iff the sub-profile induced by any shortest path between any two vertices is classically single-crossing.

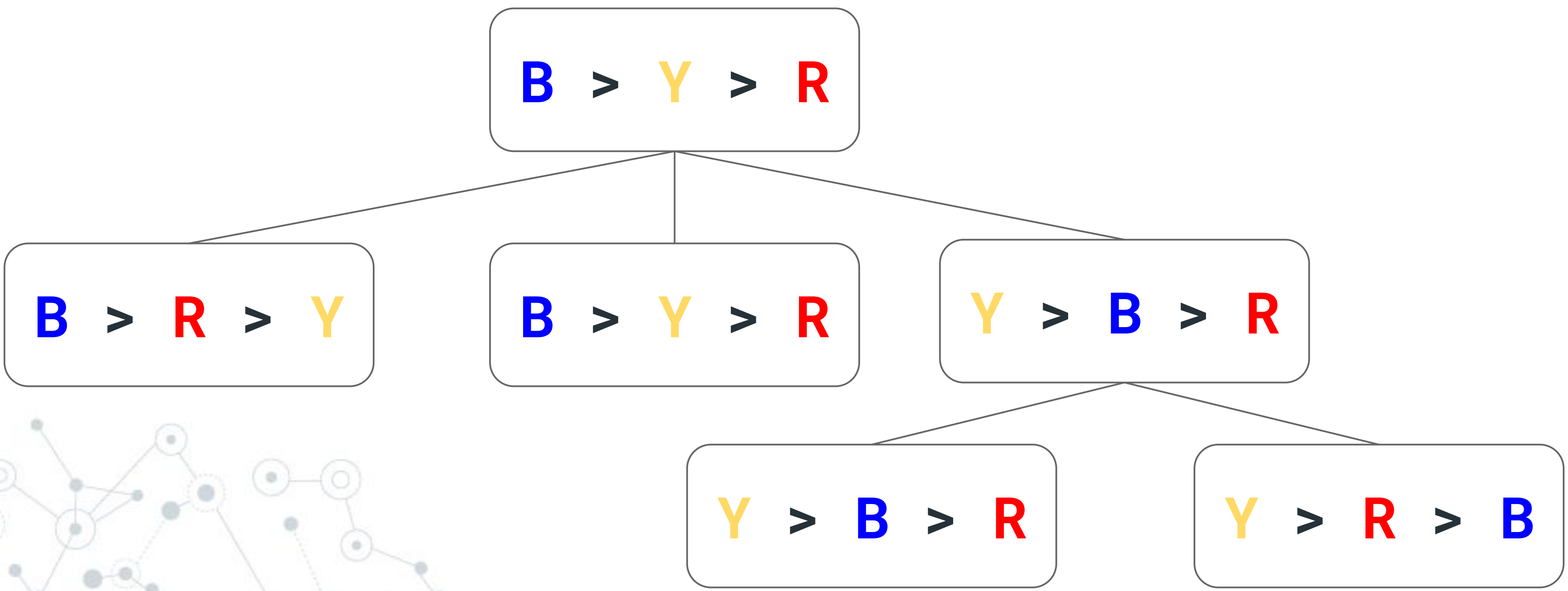
Generalized SC

A preference profile is *single-crossing with respect to a median graph* having the voters as vertices iff the sub-profile induced by any shortest path between any two vertices is classically single-crossing.



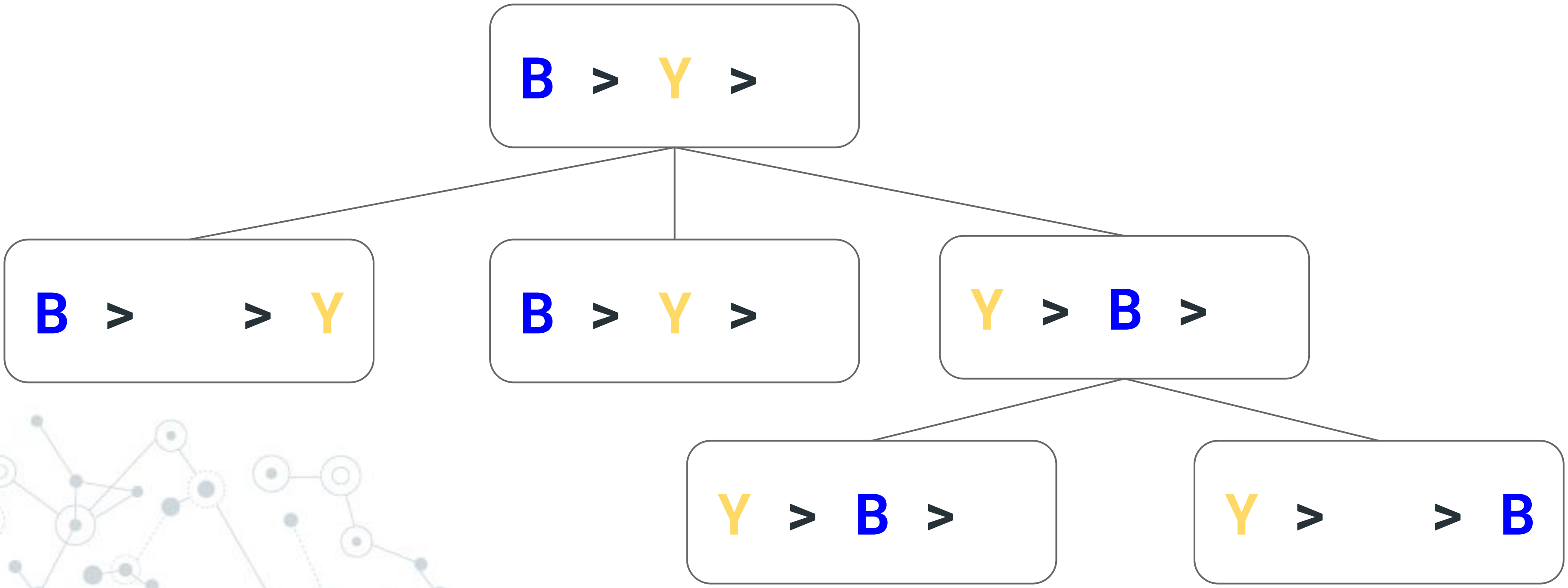
Generalized SC

A preference profile is *single-crossing with respect to a median graph* having the voters as vertices iff the sub-profile induced by any shortest path between any two vertices is classically single-crossing.



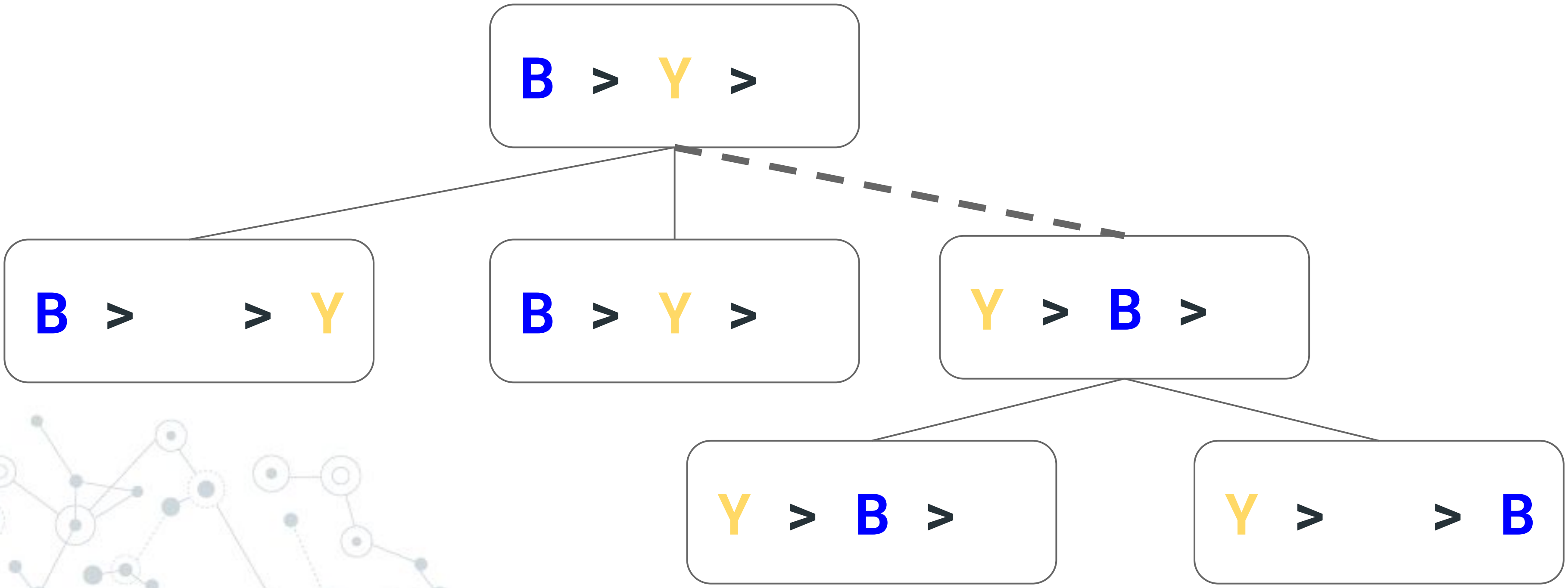
Generalized SC

A preference profile is *single-crossing with respect to a median graph* having the voters as vertices iff the sub-profile induced by any shortest path between any two vertices is classically single-crossing.



Generalized SC

A preference profile is *single-crossing with respect to a median graph* having the voters as vertices iff the sub-profile induced by any shortest path between any two vertices is classically single-crossing.



3.

This Paper

Our Contribution

Our Contribution

Our Contribution

1. We improve the current time complexity of $O(n^2mk)$ for CC under classical-SC achieved by [Skowron et al.'15]:

Our Contribution

1. We improve the current time complexity of $O(n^2mk)$ for CC under classical-SC achieved by [Skowron et al.'15]:

   - A simple tweak gives $O(nmk)$.

Our Contribution

1. We improve the current time complexity of $O(n^2mk)$ for CC under classical-SC achieved by [Skowron et al.'15]:

   - A simple tweak gives $O(nmk)$.

   - Using Monge-concavity we further get $nm2^{O\left(\sqrt{\log k \log\log n}\right)}$.

Our Contribution

1. We improve the current time complexity of $O(n^2mk)$ for CC under classical-SC achieved by [Skowron et al.'15]:

   - A simple tweak gives $O(nmk)$.

   - Using Monge-concavity we further get $nm2^{O\left(\sqrt{\log k \log\log n}\right)}$.

   - For Borda disutilities we get $O(nm \log(nm))$.

Our Contribution

1. We improve the current time complexity of $O(n^2mk)$ for CC under classical-SC achieved by [Skowron et al.'15]:

   - A simple tweak gives $O(nmk)$.

   - Using Monge-concavity we further get $nm2^{O\left(\sqrt{\log k \log \log n}\right)}$.

   - For Borda disutilities we get $O(nm \log(nm))$.

2. [Clearwater et al.'15] proposes an algorithm for CC under tree-SC. Unfortunately, the algorithm is not polynomial as claimed. We give the first polynomial algorithm.

Our Contribution

1. We improve the current time complexity of $O(n^2mk)$ for CC under classical-SC achieved by [Skowron et al.'15]:

   - A simple tweak gives $O(nmk)$.

   - Using Monge-concavity we further get $nm2^{O\left(\sqrt{\log k \log\log n}\right)}$.

   - For Borda disutilities we get $O(nm \log(nm))$.

2. [Clearwater et al.'15] proposes an algorithm for CC under tree-SC. Unfortunately, the algorithm is not polynomial as claimed. We give the first polynomial algorithm.

3. **Not in this talk**: Conjecture DP algorithm for CC under grid-SC.

CC Under Classical-SC

CC Under Classical-SC

Key observation: in any K-committee the candidates representing the voters partition the voters into continuous subsegments.

CC Under Classical-SC

Key observation: in any K-committee the candidates representing the voters partition the voters into continuous subsegments.

e.g. **K = 3**

CC Under Classical-SC

Key observation: in any K-committee the candidates representing the voters partition the voters into continuous subsegments.

e.g. **K = 3**

```
Blue   > Yellow > Red    > Green
Blue   > Red    > Yellow > Green
Red    > Blue   > Green  > Yellow
Red    > Green  > Yellow > Blue
Green  > Red    > Yellow > Blue
```

CC Under Classical-SC

Key observation: in any K-committee the candidates representing the voters partition the voters into continuous subsegments.

e.g. **K = 3**; say we removed Yellow

Blue	>		>	Red	>	Green
Blue	>	Red	>		>	Green
Red	>	Blue	>	Green	>	
Red	>	Green	>		>	Blue
Green	>	Red	>		>	Blue

CC Under Classical-SC

Key observation: in any K-committee the candidates representing the voters partition the voters into continuous subsegments.

e.g. **K = 3**; say we removed `Yellow`

Blue	>		>	Red	>	Green
Blue	>	Red	>		>	Green
Red	>	Blue	>	Green	>	
Red	>	Green	>		>	Blue
Green	>	Red	>		>	Blue

CC Under Classical-SC

Key observation: in any K-committee the candidates representing the voters partition the voters into continuous subsegments.

e.g. **K = 3**; say we removed **Blue**

	>	Yellow	>	Red	>	Green
	>	Red	>	Yellow	>	Green
Red	>		>	Green	>	Yellow
Red	>	Green	>	Yellow	>	
Green	>	Red	>	Yellow	>	

CC Under Classical-SC

Key observation: in any K-committee the candidates representing the voters partition the voters into continuous subsegments.

e.g. **K = 3**; say we removed **Blue**

	>	Yellow	>	Red	>	Green
	>	Red	>	Yellow	>	Green
Red	>		>	Green	>	Yellow
Red	>	Green	>	Yellow	>	
Green	>	Red	>	Yellow	>	

This allows simple interval DP to work [Skowron et al.'15], with more care it can be implemented in $O(nmk)$.

CC Under Classical-SC

Reduction to minimum K-link path in DAG

CC Under Classical-SC

Reduction to minimum K-link path in DAG

- Define $f(i, j)$ for $0 \leq i < j \leq N$ to be the least possible total cost to represent voters $v_{i+1} \ldots v_j$ with a single candidate.

CC Under Classical-SC

Reduction to minimum K-link path in DAG

- Define $f(i, j)$ for $0 \le i < j \le N$ to be the least possible total cost to represent voters $v_{i+1} \dots v_j$ with a single candidate.
- Define a DAG with vertices $0 \dots N$ and edges (i, j) for $0 \le i < j \le N$ of cost $f(i, j)$.

CC Under Classical-SC

Reduction to minimum K-link path in DAG

- Define $f(i, j)$ for $0 \leq i < j \leq N$ to be the least possible total cost to represent voters $v_{i+1} \dots v_j$ with a single candidate.
- Define a DAG with vertices $0 \dots N$ and edges (i, j) for $0 \leq i < j \leq N$ of cost $f(i, j)$.
- Then, our problem is to find the minimum total weight path starting at 0, ending at N, and consisting of exactly K edges.

CC Under Classical-SC

Reduction to minimum K-link path in DAG

- Define f(i, j) for $0 \le i < j \le N$ to be the least possible total cost to represent voters $v_{i+1} \dots v_j$ with a single candidate.
- Define a DAG with vertices $0 \dots N$ and edges (i, j) for $0 \le i < j \le N$ of cost f(i, j).
- Then, our problem is to find the minimum total weight path starting at 0, ending at N, and consisting of exactly K edges.

Blue	>		>	Red	>	Green
Blue	>	Red	>		>	Green

Red	>	Blue	>	Green	>	
Red	>	Green	>		>	Blue

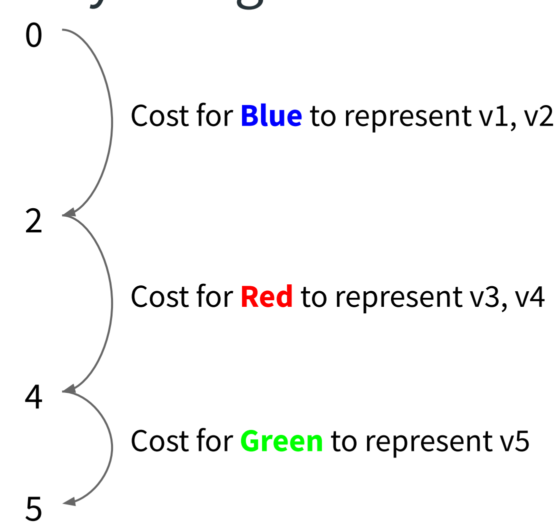
Green	>	Red	>		>	Blue

CC Under Classical-SC

Reduction to minimum K-link path in DAG

- Define f(i, j) for $0 \leq i < j \leq N$ to be the least possible total cost to represent voters $v_{i+1} \ldots v_j$ with a single candidate.
- Define a DAG with vertices $0 \ldots N$ and edges (i, j) for $0 \leq i < j \leq N$ of cost f(i, j).
- Then, our problem is to find the minimum total weight path starting at 0, ending at N, and consisting of exactly K edges.

Blue	>		>	Red	>	Green
Blue	>	Red	>		>	Green
Red	>	Blue	>	Green	>	
Red	>	Green	>		>	Blue
Green	>	Red	>		>	Blue



CC Under Classical-SC

Lemma Assume $a < b < c < d$, then it holds that $f(a, c) + f(b, d) \leq f(a, d) + f(b, c)$ (i.e. the costs f are *Monge-concave*).

CC Under Classical-SC

Lemma Assume $a < b < c < d$, then it holds that $f(a, c) + f(b, d) \leq f(a, d) + f(b, c)$ (i.e. the costs f are *Monge-concave*).

Proof Idea First, show by contradiction that if there is a counterexample, then there is one with **N = 3**.

CC Under Classical-SC

Lemma Assume $a < b < c < d$, then it holds that $f(a, c) + f(b, d) \leq f(a, d) + f(b, c)$ (i.e. the costs f are *Monge-concave*).

Proof Idea First, show by contradiction that if there is a counterexample, then there is one with $\mathbf{N = 3}$.

As a result, we get *Monge-concave* instances of the minimum K-link path problem for DAGs. Relevant work:

CC Under Classical-SC

Lemma Assume $a < b < c < d$, then it holds that $f(a, c) + f(b, d) \leq f(a, d) + f(b, c)$ (i.e. the costs f are *Monge-concave*).

Proof Idea First, show by contradiction that if there is a counterexample, then there is one with **N = 3**.

As a result, we get *Monge-concave* instances of the minimum K-link path problem for DAGs. Relevant work:

- [Bein, Larmore, Park'92], [Aggarwal, Schieber, Tokuyama'94] - Give

O(n log (nU)) algorithm, where U bounds dissatisfactions.

CC Under Classical-SC

Lemma Assume a < b < c < d, then it holds that f(a, c) + f(b, d) ≤ f(a, d) + f(b, c) (i.e. the costs f are *Monge-concave*).

Proof Idea First, show by contradiction that if there is a counterexample, then there is one with **N = 3**.

As a result, we get *Monge-concave* instances of the minimum K-link path problem for DAGs. Relevant work:

- [Bein, Larmore, Park'92], [Aggarwal, Schieber, Tokuyama'94] - Give *O(n log (nU))* algorithm, where U bounds dissatisfactions.
- [Schieber'95] - Gives $n2^{O\left(\sqrt{\log k \log\log n}\right)}$ for $k = \Omega(\log n)$.

CC Under Classical-SC

Lemma Assume a < b < c < d, then it holds that f(a, c) + f(b, d) ≤ f(a, d) + f(b, c) (i.e. the costs f are *Monge-concave*).

Proof Idea First, show by contradiction that if there is a counterexample, then there is one with **N = 3**.

As a result, we get *Monge-concave* instances of the minimum K-link path problem for DAGs. Relevant work:

- [Bein, Larmore, Park'92], [Aggarwal, Schieber, Tokuyama'94] - Give *O(n log (nU))* algorithm, where U bounds dissatisfactions.
- [Schieber'95] - Gives $n2^{O\left(\sqrt{\log k \log \log n}\right)}$ for $k = \Omega(\log n)$.

Need extra factor of *m* due to time to compute f(i, j)!

CC Under Classical-SC

Lemma Assume a < b < c < d, then it holds that f(a, c) + f(b, d) ≤ f(a, d) + f(b, c) (i.e. the costs f are *Monge-concave*).

Proof Idea First, show by contradiction that if there is a counterexample, then there is one with **N = 3**.

As a result, we get *Monge-concave* instances of the minimum K-link path problem for DAGs. Relevant work:

- [Bein, Larmore, Park'92], [Aggarwal, Schieber, Tokuyama'94] - Give *O(n log (nU))* algorithm, where U bounds dissatisfactions.
- [Schieber'95] - Gives $n2^{O\left(\sqrt{\log k \log\log n}\right)}$ for $k = \Omega(\log n)$.

Need extra factor of *m* due to time to compute f(i, j)!

Remark For egalitarian, binary search the answer and then run algorithm on instance with 0-1 dissatisfactions. This gives *O(nm log n log (nm))*.

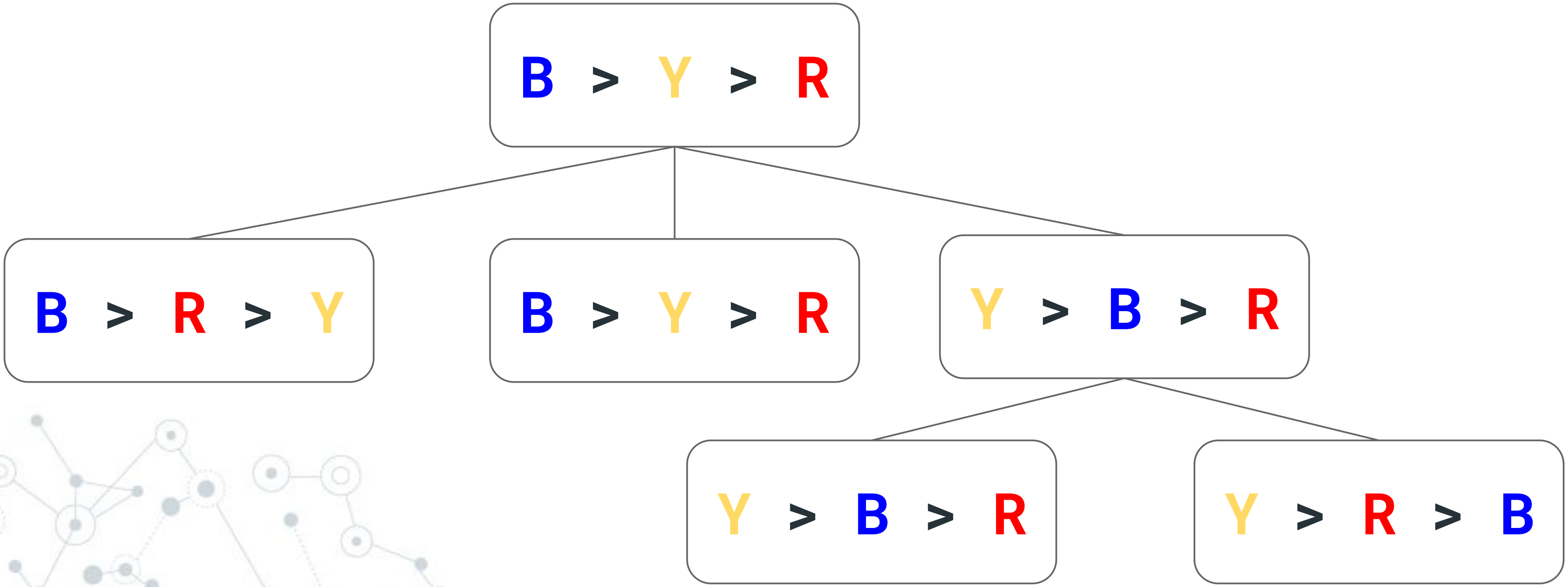
CC Under Tree-SC

CC Under Tree-SC

Similar **Connectivity Observation**: In any K-committee the candidates representing the voters partition the voters into connected subtrees.

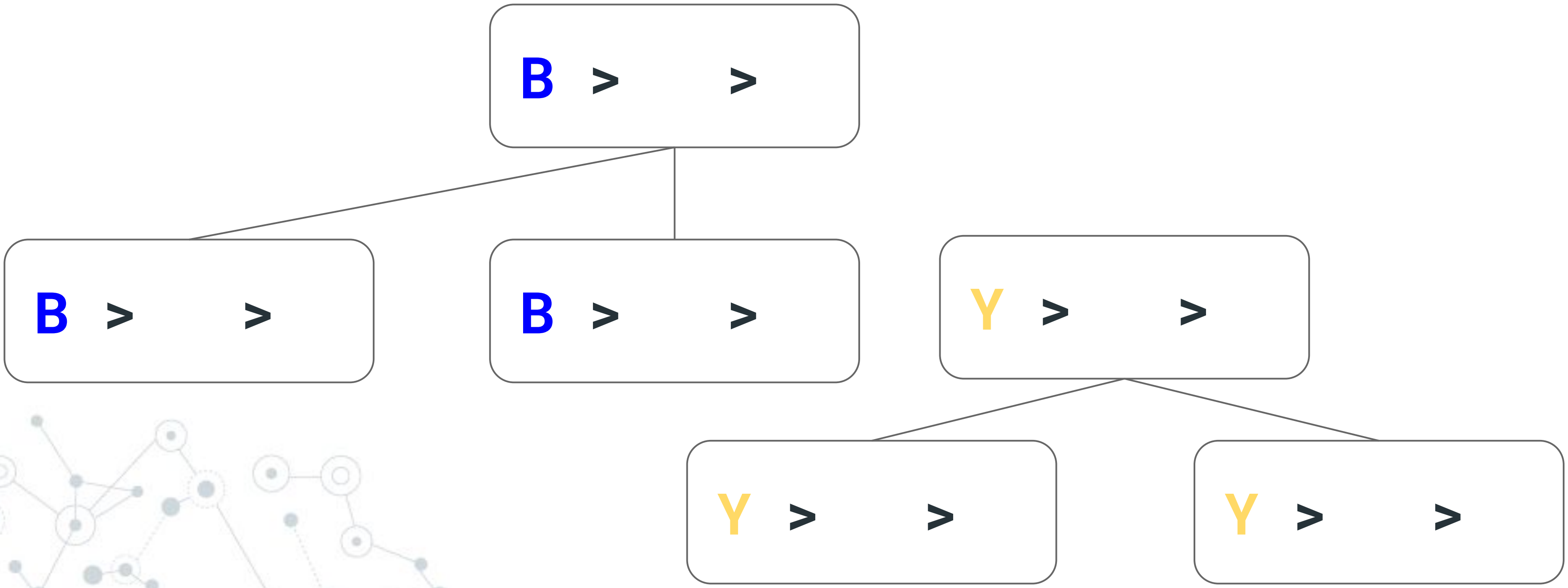
CC Under Tree-SC

Similar **Connectivity Observation**: In any K-committee the candidates representing the voters partition the voters into connected subtrees.



CC Under Tree-SC

Similar **Connectivity Observation**: In any K-committee the candidates representing the voters partition the voters into connected subtrees.

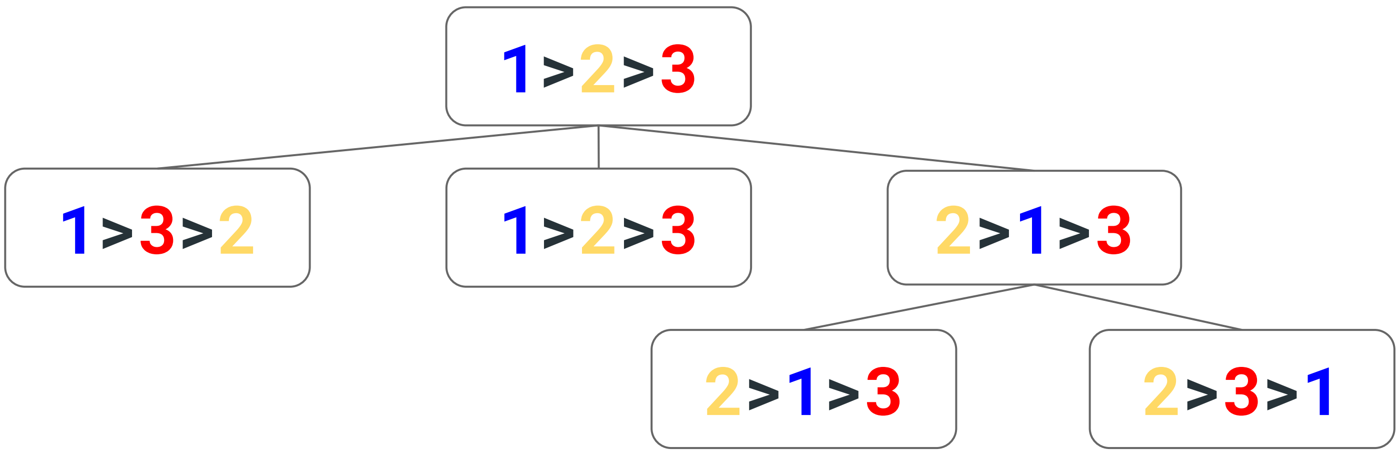


CC Under Tree-SC

Assume candidates are numbered 1, 2, …, M. Root the tree and assume that the root has the order 1 > 2 > … > M.

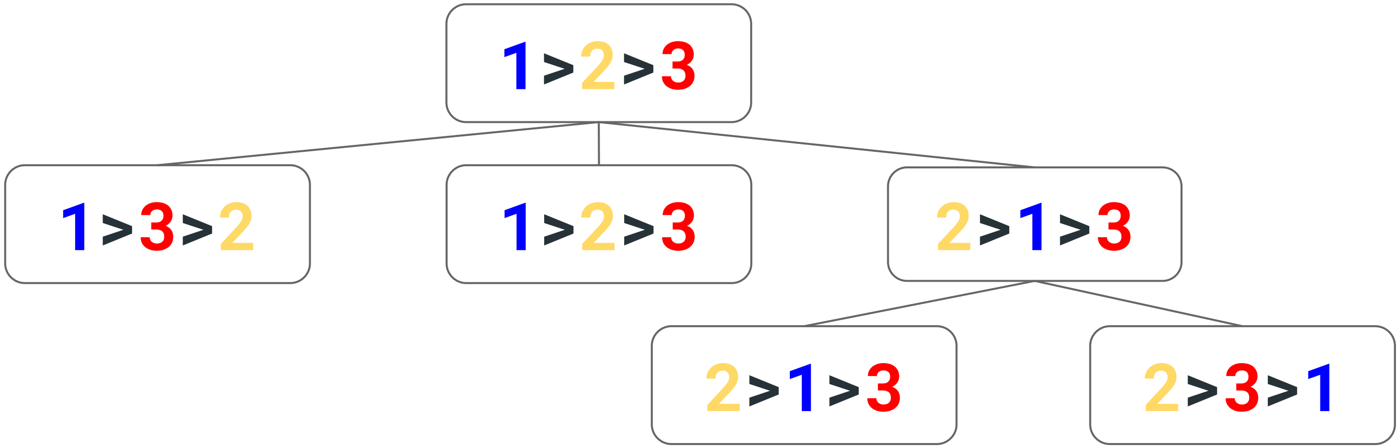
CC Under Tree-SC

Assume candidates are numbered 1, 2, …, M. Root the tree and assume that the root has the order 1 > 2 > … > M.



CC Under Tree-SC

Assume candidates are numbered 1, 2, …, M. Root the tree and assume that the root has the order 1 > 2 > … > M.

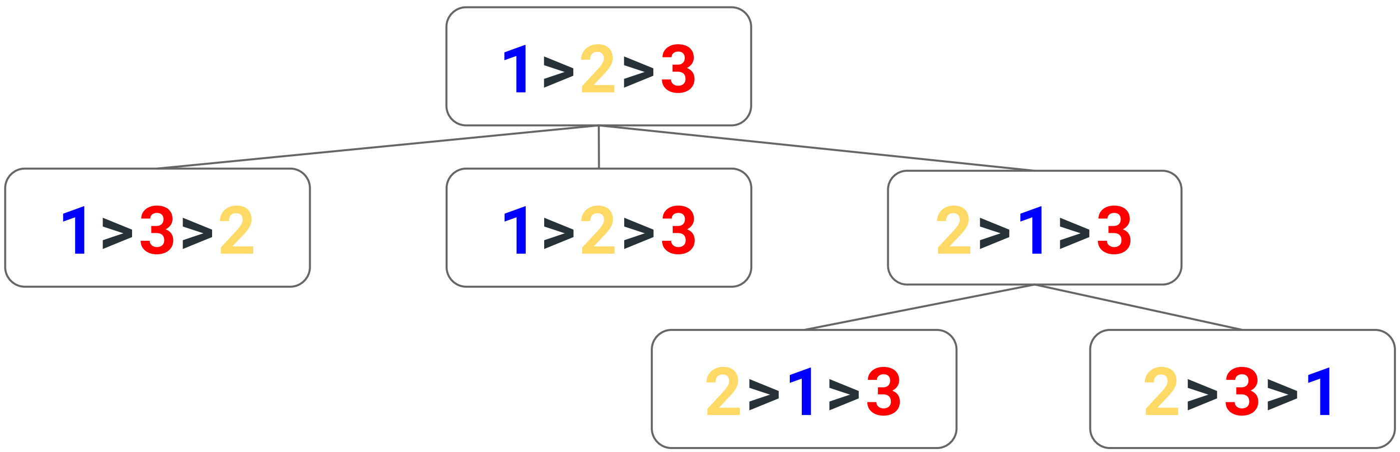


Monotonicity Lemma

In any K-committee, while walking down the tree the representing candidate is non-decreasing.

CC Under Tree-SC

Assume candidates are numbered 1, 2, …, M. Root the tree and assume that the root has the order 1 > 2 > … > M.



Monotonicity Lemma (ins. [Clearwater et al.'15])

In any K-committee, while walking down the tree the representing candidate is non-decreasing.

CC Under Tree-SC

CC Under Tree-SC

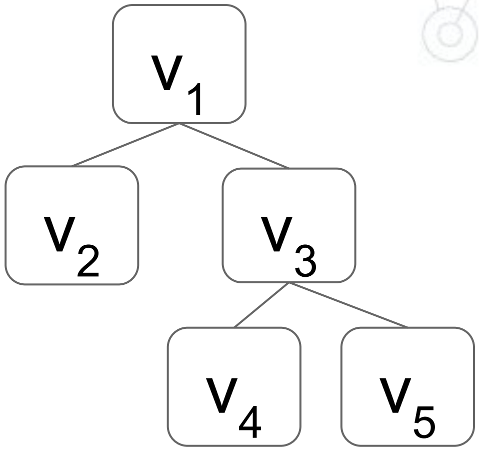
- Assume tree is **binary** to simplify presentation - general case is more tricky.

CC Under Tree-SC

- Assume tree is **binary** to simplify presentation - general case is more tricky.
- Say tree is rooted in v_1.

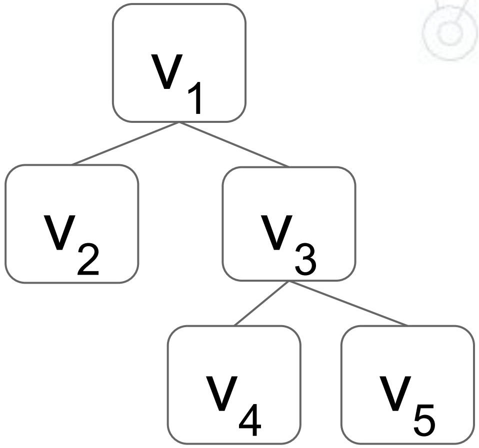
CC Under Tree-SC

- Assume tree is **binary** to simplify presentation - general case is more tricky.
- Say tree is rooted in v_1.



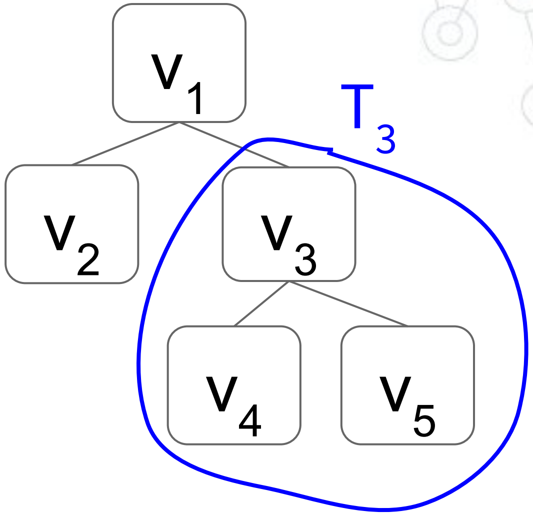
CC Under Tree-SC

- Assume tree is **binary** to simplify presentation - general case is more tricky.
- Say tree is rooted in v_1. Define T_i to be the downwards subtree of v_i.



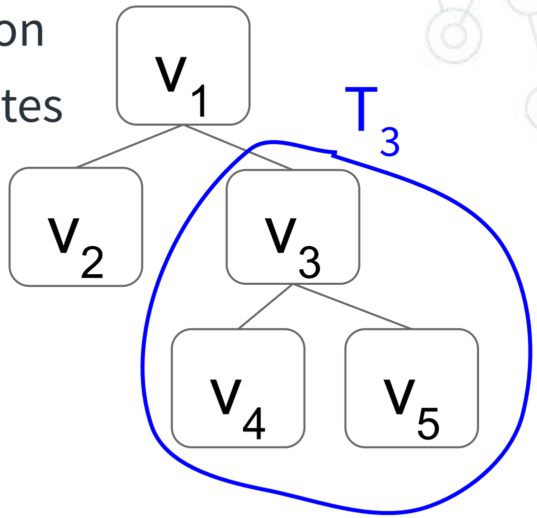
CC Under Tree-SC

- Assume tree is **binary** to simplify presentation - general case is more tricky.
- Say tree is rooted in v_1. Define T_i to be the downwards subtree of v_i.



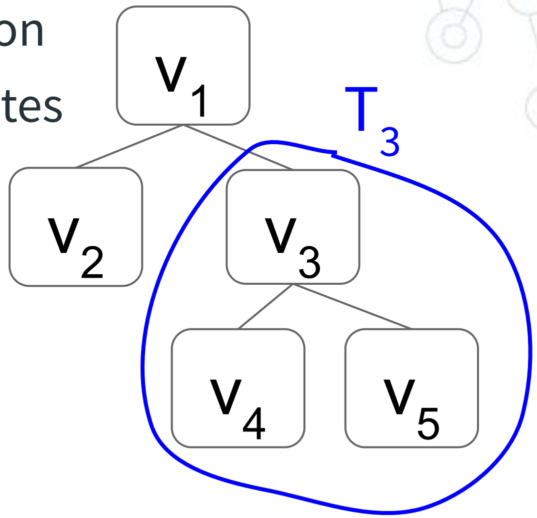
CC Under Tree-SC

- Assume tree is **binary** to simplify presentation - general case is more tricky.
- Say tree is rooted in v_1. Define T_i to be the downwards subtree of v_i.
- Define $dp[v_i][c][k]$ to be the least possible dissatisfaction of voters in T_i if we are allowed to use at most k candidates from the set c, c + 1, …, m;



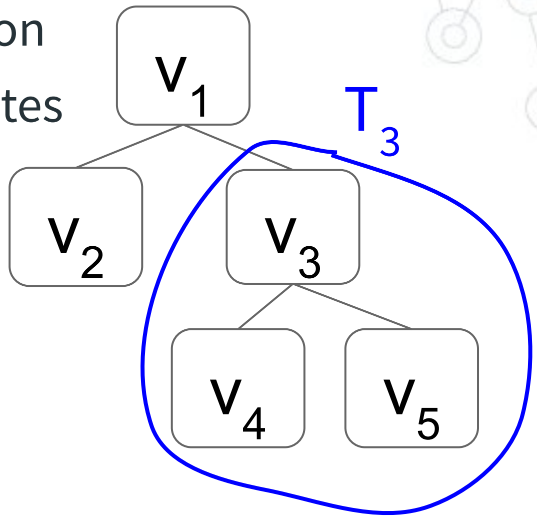
CC Under Tree-SC

- Assume tree is **binary** to simplify presentation - general case is more tricky.
- Say tree is rooted in v_1. Define T_i to be the downwards subtree of v_i.
- Define $dp[v_i][c][k]$ to be the least possible dissatisfaction of voters in T_i if we are allowed to use at most k candidates from the set c, c + 1, …, m; and $dp'[v_i][c][k]$ to be the same, but **enforcing** v_i is represented by candidate c.



CC Under Tree-SC

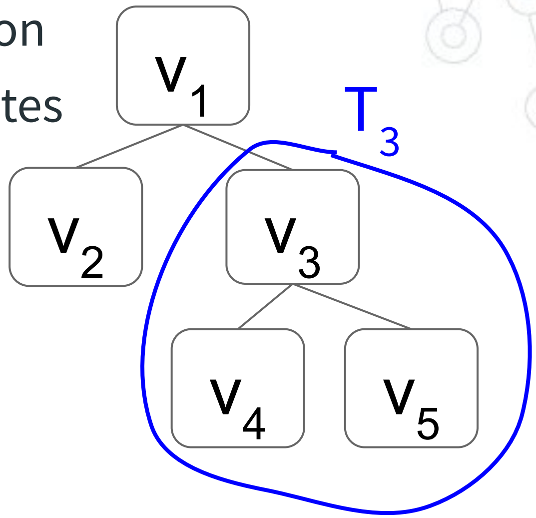
- Assume tree is **binary** to simplify presentation - general case is more tricky.
- Say tree is rooted in v_1. Define T_i to be the downwards subtree of v_i.
- Define $dp[v_i][c][k]$ to be the least possible dissatisfaction of voters in T_i if we are allowed to use at most k candidates from the set c, c + 1, …, m; and $dp'[v_i][c][k]$ to be the same, but **enforcing** v_i is represented by candidate c.



Interesting case: A node v with two children l and r.

CC Under Tree-SC

- Assume tree is **binary** to simplify presentation - general case is more tricky.
- Say tree is rooted in v_1. Define T_i to be the downwards subtree of v_i.
- Define $dp[v_i][c][k]$ to be the least possible dissatisfaction of voters in T_i if we are allowed to use at most k candidates from the set c, c + 1, …, m; and $dp'[v_i][c][k]$ to be the same, but **enforcing** v_i is represented by candidate c.

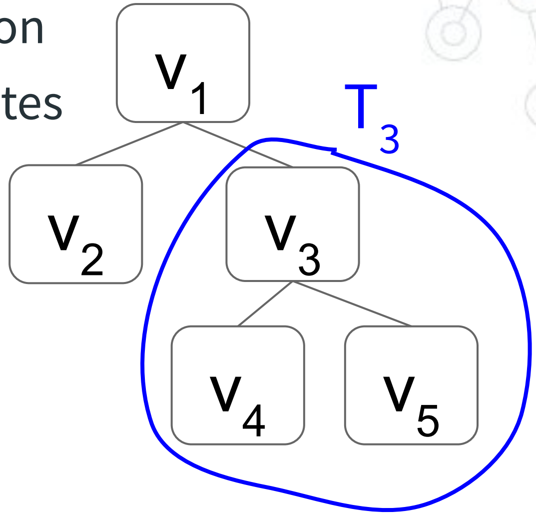


Interesting case: A node v with two children l and r.

```
dp[v][c][k] = min { dp'[v][c][k], dp[v][c + 1][k] }
```

CC Under Tree-SC

- Assume tree is **binary** to simplify presentation - general case is more tricky.
- Say tree is rooted in v_1. Define T_i to be the downwards subtree of v_i.
- Define dp[v_i][c][k] to be the least possible dissatisfaction of voters in T_i if we are allowed to use at most k candidates from the set c, c + 1, …, m; and dp'[v_i][c][k] to be the same, but **enforcing** v_i is represented by candidate c.

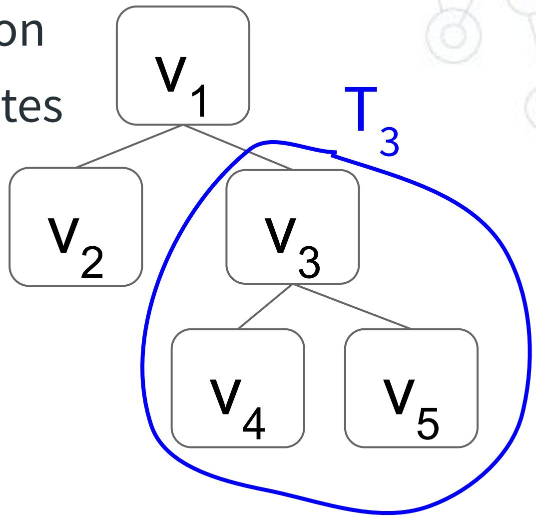


Interesting case: A node v with two children l and r.

```
dp[v][c][k] = min { dp'[v][c][k], dp[v][c + 1][k] }
dp'[v][c][k] = dis(v, c)
```

CC Under Tree-SC

- Assume tree is **binary** to simplify presentation - general case is more tricky.
- Say tree is rooted in v_1. Define T_i to be the downwards subtree of v_i.
- Define dp[v_i][c][k] to be the least possible dissatisfaction of voters in T_i if we are allowed to use at most k candidates from the set c, c + 1, …, m; and dp'[v_i][c][k] to be the same, but **enforcing** v_i is represented by candidate c.

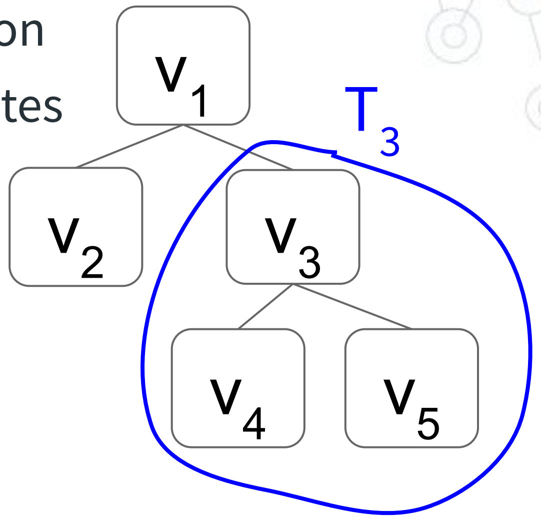


Interesting case: A node v with two children l and r.

```
dp[v][c][k] = min { dp'[v][c][k], dp[v][c + 1][k] }
dp'[v][c][k] = dis(v, c)
  + min { dp'[l][c]    [k'] + dp'[r][c]    [k - k'],
     k'
```

CC Under Tree-SC

- Assume tree is **binary** to simplify presentation - general case is more tricky.
- Say tree is rooted in v_1. Define T_i to be the downwards subtree of v_i.
- Define $dp[v_i][c][k]$ to be the least possible dissatisfaction of voters in T_i if we are allowed to use at most k candidates from the set c, c + 1, …, m; and $dp'[v_i][c][k]$ to be the same, but **enforcing** v_i is represented by candidate c.

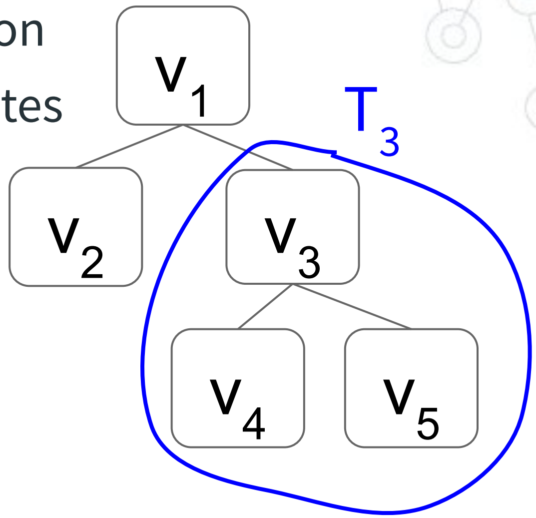


Interesting case: A node v with two children l and r.

```
dp[v][c][k] = min { dp'[v][c][k], dp[v][c + 1][k] }
dp'[v][c][k] = dis(v, c)
  + min { dp'[l][c]     [k'] + dp'[r][c]     [k - k'],
     k'   dp'[l][c]     [k'] + dp [r][c + 1][k - k'],
```

CC Under Tree-SC

- Assume tree is **binary** to simplify presentation - general case is more tricky.
- Say tree is rooted in v_1. Define T_i to be the downwards subtree of v_i.
- Define dp[v_i][c][k] to be the least possible dissatisfaction of voters in T_i if we are allowed to use at most k candidates from the set c, c + 1, …, m; and dp'[v_i][c][k] to be the same, but **enforcing** v_i is represented by candidate c.

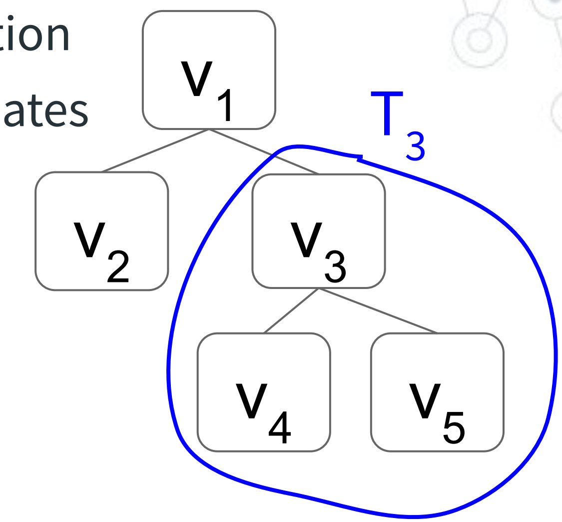


Interesting case: A node v with two children l and r.

```
dp[v][c][k] = min { dp'[v][c][k], dp[v][c + 1][k] }
dp'[v][c][k] = dis(v, c)
  + min { dp'[l][c]    [k'] + dp'[r][c]    [k - k'],
     k'   dp'[l][c]    [k'] + dp [r][c + 1][k - k'],
          dp [l][c + 1][k'] + dp'[r][c]    [k - k'],
```

CC Under Tree-SC

- Assume tree is **binary** to simplify presentation - general case is more tricky.
- Say tree is rooted in v_1. Define T_i to be the downwards subtree of v_i.
- Define $dp[v_i][c][k]$ to be the least possible dissatisfaction of voters in T_i if we are allowed to use at most k candidates from the set c, c + 1, …, m; and $dp'[v_i][c][k]$ to be the same, but **enforcing** v_i is represented by candidate c.

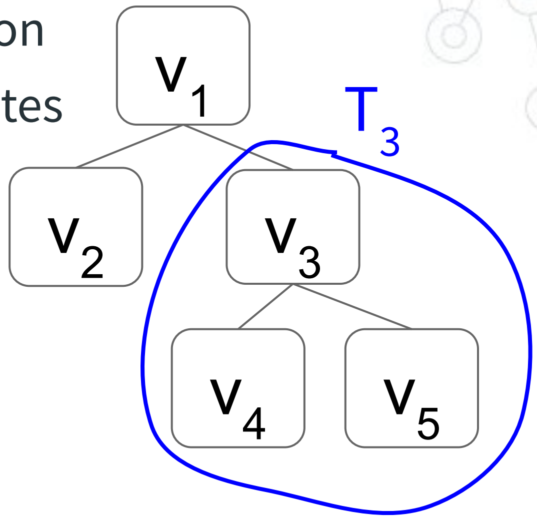


Interesting case: A node v with two children l and r.

```
dp[v][c][k] = min { dp'[v][c][k], dp[v][c + 1][k] }
dp'[v][c][k] = dis(v, c)
  + min { dp'[l][c]    [k'] + dp'[r][c]    [k - k'],
     k'   dp'[l][c]    [k'] + dp [r][c + 1][k - k'],
          dp [l][c + 1][k'] + dp'[r][c]    [k - k'],
          dp [l][c + 1][k'] + dp [r][c + 1][k - k' - 1] }
```

CC Under Tree-SC

- Assume tree is **binary** to simplify presentation - general case is more tricky.
- Say tree is rooted in v_1. Define T_i to be the downwards subtree of v_i.
- Define dp[v_i][c][k] to be the least possible dissatisfaction of voters in T_i if we are allowed to use at most k candidates from the set c, c + 1, …, m; and dp'[v_i][c][k] to be the same, but **enforcing** v_i is represented by candidate c.



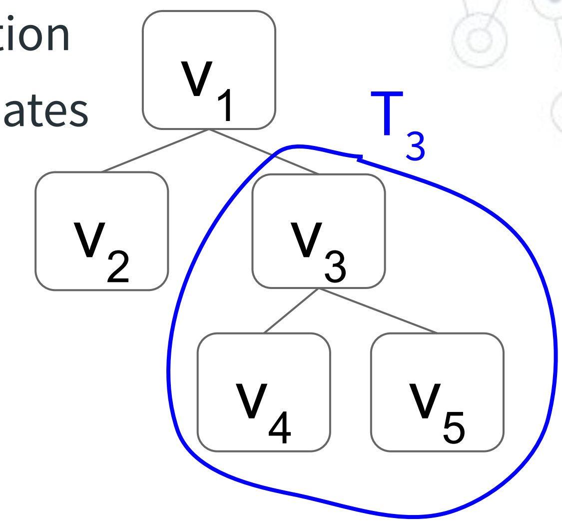
Interesting case: A node v with two children l and r.

```
dp[v][c][k] = min { dp'[v][c][k], dp[v][c + 1][k] }
dp'[v][c][k] = dis(v, c)
  + min { dp'[l][c]     [k'] + dp'[r][c]     [k - k'],
      k'   dp'[l][c]     [k'] + dp [r][c + 1][k - k'],
           dp [l][c + 1][k'] + dp'[r][c]     [k - k'],
           dp [l][c + 1][k'] + dp [r][c + 1][k - k' - 1] }
```

- *O(nmk)* states, but *O(nmk²)* time!

CC Under Tree-SC

- Assume tree is **binary** to simplify presentation - general case is more tricky.
- Say tree is rooted in v_1. Define T_i to be the downwards subtree of v_i.
- Define dp[v_i][c][k] to be the least possible dissatisfaction of voters in T_i if we are allowed to use at most k candidates from the set c, c + 1, …, m; and dp'[v_i][c][k] to be the same, but **enforcing** v_i is represented by candidate c.



Interesting case: A node v with two children l and r.

```
dp[v][c][k] = min { dp'[v][c][k], dp[v][c + 1][k] }
dp'[v][c][k] = dis(v, c)
  + min { dp'[l][c]    [k'] + dp'[r][c]    [k - k'],
     k'   dp'[l][c]    [k'] + dp [r][c + 1][k - k'],
          dp [l][c + 1][k'] + dp'[r][c]    [k - k'],
          dp [l][c + 1][k'] + dp [r][c + 1][k - k' - 1] }
```

- *O(nmk)* states, but *O(nmk²)* time!
- With care can be implemented in *O(nmk)*.

Future Directions

Future Directions

1. How to solve CC for grid-SC?

Future Directions

1. How to solve CC for grid-SC?

2. Does some form of concavity hold for trees?

Future Directions

1. How to solve CC for grid-SC?
2. Does some form of concavity hold for trees?
3. Is CC for median graphs NP-hard?

Hope you enjoyed!

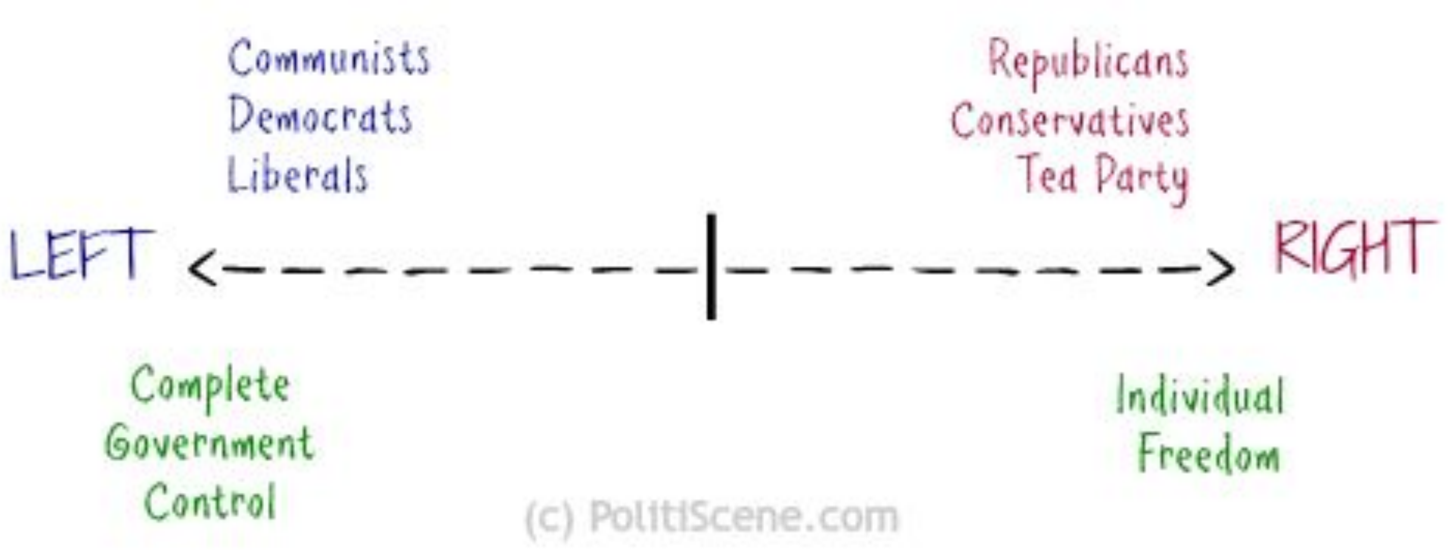
Intuition

Intuition

Imagine with every voter/candidate we associate a real number:

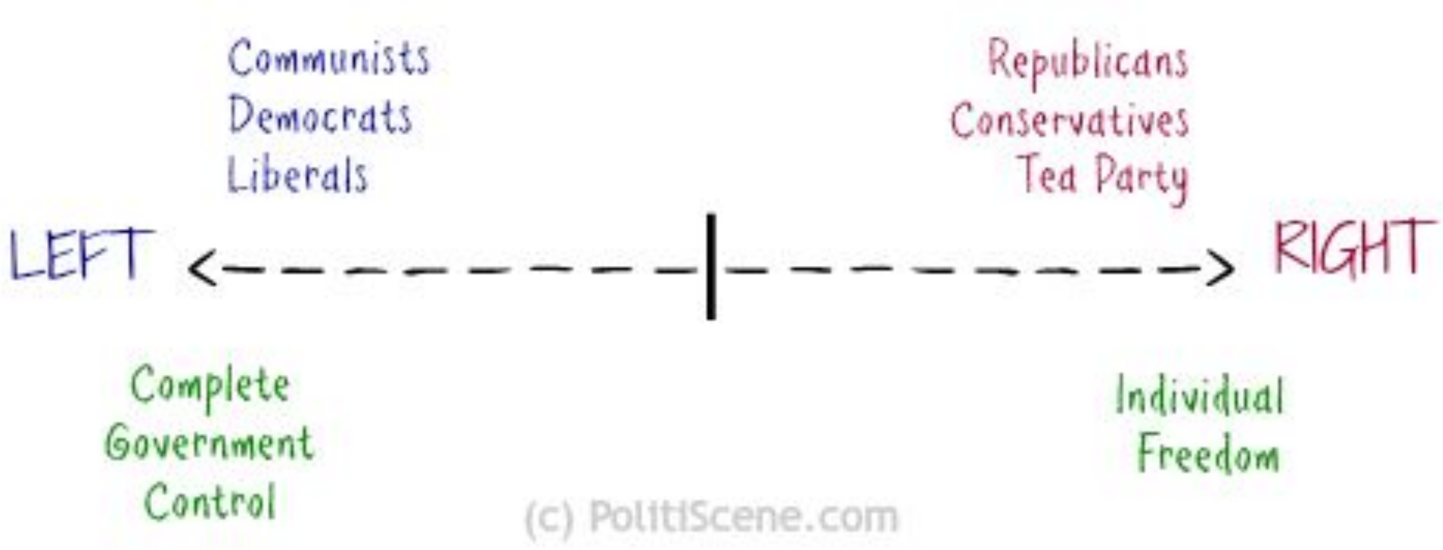
Intuition

Imagine with every voter/candidate we associate a real number:



Intuition

Imagine with every voter/candidate we associate a real number:



Voters vote based on how far off a candidate's number is from their own.