An Operations Research Look at Voting

Michael A. Trick
Tepper School of Business
Carnegie Mellon
USA

ADT Venice
October 2009

Lets go back to 15th Century

Electing the Doge

Thirty members of the Great Council, chosen by lot,

Lets go back to 15th Century

Electing the Doge

Thirty members of the Great Council, chosen by lot, were reduced by lot to nine;

Lets go back to 15th Century

Electing the Doge

Thirty members of the Great Council, chosen by lot, were reduced by lot to nine; the nine chose forty

Lets go back to 15th Century

Electing the Doge

Thirty members of the Great Council, chosen by lot, were reduced by lot to nine; the nine chose forty and the forty were reduced by lot to twelve,

Lets go back to 15th Century

Electing the Doge

Thirty members of the Great Council, chosen by lot, were reduced by lot to nine; the nine chose forty and the forty were reduced by lot to twelve, who chose twenty-five.

Lets go back to 15th Century

Electing the Doge

Thirty members of the Great Council, chosen by lot, were reduced by lot to nine; the nine chose forty and the forty were reduced by lot to twelve, who chose twenty-five. The twenty-five were reduced by lot to nine

Lets go back to 15th Century

Electing the Doge

Thirty members of the Great Council, chosen by lot, were reduced by lot to nine; the nine chose forty and the forty were reduced by lot to twelve, who chose twenty-five. The twenty-five were reduced by lot to nine and the nine elected forty-five.

Lets go back to 15th Century

Electing the Doge

Thirty members of the Great Council, chosen by lot, were reduced by lot to nine; the nine chose forty and the forty were reduced by lot to twelve, who chose twenty-five. The twenty-five were reduced by lot to nine and the nine elected forty-five. Then the forty-five were once more reduced by lot to eleven,

Lets go back to 15th Century

Electing the Doge

Thirty members of the Great Council, chosen by lot, were reduced by lot to nine; the nine chose forty and the forty were reduced by lot to twelve, who chose twenty-five. The twenty-five were reduced by lot to nine and the nine elected forty-five. Then the forty-five were once more reduced by lot to eleven, and the eleven finally chose the forty-one

Lets go back to 15th Century

Electing the Doge

Thirty members of the Great Council, chosen by lot, were reduced by lot to nine; the nine chose forty and the forty were reduced by lot to twelve, who chose twenty-five. The twenty-five were reduced by lot to nine and the nine elected forty-five. Then the forty-five were once more reduced by lot to eleven, and the eleven finally chose the forty-one who actually elected the doge.

Enter Lewis Carrol

Charles Dodgson came up with a voting system (perhaps in order to "win" a vote on a belfry in Oxford, but also spurred by decisions on studentships) that involved finding the best way to flip adjacent candidates in preference orders to get a Condorcet winner (a candidate who beats every other candidate one-on-one).

Outline

What is Operations Research?

Operations Research is the art and science of making better decisions through mathematical models.

What is Operations Research?

Operations Research is the art and science of making better decisions through mathematical models.

Generally involves models with objectives, variables, and constraints and techniques such as linear and integer programming (and constraint programming, though most in CP identify with computer science, rather than OR)

What is Operations Research?

Operations Research is the art and science of making better decisions through mathematical models.

Generally involves models with objectives, variables, and constraints and techniques such as linear and integer programming (and constraint programming, though most in CP identify with computer science, rather than OR)

Much overlap with a CS view, particularly the part of CS that is willing to solve NP-complete problems (SAT, CP), but I will point out some directions that are more OR-ish.

Voting through Operations Research Eyes

- See voting rule as an algorithm

Voting through Operations Research Eyes

- See voting rule as an algorithm
- Pretty obvious now, but new idea 20 years ago.

Voting through Operations Research Eyes

- See voting rule as an algorithm
- Pretty obvious now, but new idea 20 years ago.
- 20 years? Really? But (I'll claim) this acts like a five year old idea

Voting through Operations Research Eyes

- See voting rule as an algorithm
- Pretty obvious now, but new idea 20 years ago.
- 20 years? Really? But (I'll claim) this acts like a five year old idea
- And then I'll give you some more recent stuff (including results from last week)

INTRODUCTION

Problem: Given n alternatives, and v voters, each with an preference ordering on the alternatives, aggregate them into either
(a) a "winner" or winners, or
(b) a total ordering of the alternatives (where the "winner" is the first in the ordering).

INTRODUCTION

Problem: Given n alternatives, and v voters, each with an preference ordering on the alternatives, aggregate them into either
(a) a "winner" or winners, or
(b) a total ordering of the alternatives (where the "winner" is the first in the ordering).

Huge number of voting rules. A voting rule should be "fair", decisive, and practical.

EXAMPLE

Suppose the preferences are:

3 voters	$a \succ b \succ e \succ c \succ d$
2 voters	$c \succ a \succ e \succ b \succ d$
4 voters	$d \succ b \succ e \succ c \succ a$
1 voter	$d \succ a \succ e \succ b \succ a$
2 voters	$c \succ a \succ e \succ b \succ d$

Who should be the winner?

EXAMPLE

Suppose the preferences are:

3 voters	$a \succ b \succ e \succ c \succ d$
2 voters	$c \succ a \succ e \succ b \succ d$
4 voters	$d \succ b \succ e \succ c \succ a$
1 voter	$d \succ a \succ e \succ b \succ a$
2 voters	$c \succ a \succ e \succ b \succ d$

Who should be the winner?
d with the most first place votes?

EXAMPLE

Suppose the preferences are:

3 voters	$a \succ b \succ e \succ c \succ d$
2 voters	$c \succ a \succ e \succ b \succ d$
4 voters	$d \succ b \succ e \succ c \succ a$
1 voter	$d \succ a \succ e \succ b \succ a$
2 voters	$c \succ a \succ e \succ b \succ d$

Who should be the winner?
d with the most first place votes?
e who no one dislikes too much?

EXAMPLE

Suppose the preferences are:

3 voters	$a \succ b \succ e \succ c \succ d$
2 voters	$c \succ a \succ e \succ b \succ d$
4 voters	$d \succ b \succ e \succ c \succ a$
1 voter	$d \succ a \succ e \succ b \succ a$
2 voters	$c \succ a \succ e \succ b \succ d$

Who should be the winner?
d with the most first place votes?
e who no one dislikes too much?
a with the most first or second place votes?

EXAMPLE

Suppose the preferences are:

3 voters	$a \succ b \succ e \succ c \succ d$
2 voters	$c \succ a \succ e \succ b \succ d$
4 voters	$d \succ b \succ e \succ c \succ a$
1 voter	$d \succ a \succ e \succ b \succ a$
2 voters	$c \succ a \succ e \succ b \succ d$

Who should be the winner?
d with the most first place votes?
e who no one dislikes too much?
a with the most first or second place votes?
Result from run-off elections (how?)?

VOTING THEORY ISSUES

Devise a voting rule that seems "fair" (for a suitable definition of fairness).

VOTING THEORY ISSUES

Devise a voting rule that seems “fair” (for a suitable definition of fairness).

Unanimity: A voting system satisfies *unanimity* if whenever every voter prefers candidate c to d, then d is not the winner.

VOTING THEORY ISSUES

Devise a voting rule that seems “fair” (for a suitable definition of fairness).

Unanimity: A voting system satisfies *unanimity* if whenever every voter prefers candidate c to d, then d is not the winner.

Independence of Irrelevant Alternatives: A voting system satisfies *independence of irrelevant alternatives* if the decision of c versus d depends only on the relative ranking of c and d in the voter preference profiles.

IMPOSSIBILITY THEOREMS

Theorem (Arrow): The only voting rule that satisfies Unanimity and Independence of Irrelevant Alternatives is Dictatorship

Many, many efforts build on this: is IIA relevant? should we allow any possible input? etc. etc.

ALGORITHMIC QUESTION

How quickly can we determine the result under a particular voting rule?

n candidates, v voters.

Plurality: $O(n)$
Borda and many others: $O(nv)$

Even low order polynomials would be a problem (U.S. election with an $\theta(v^3)$ algorithm?).
Can it get worse?

CONDORCET CRITERION

Definition: Given a voting instance, if a candidate c is preferred to each other candidate by a majority of voters, then c is the *Condorcet winner*.

If every instance had a Condorcet winner, then choosing it would satisfy Unanimity and IIA, but some instances do not
$(a \succ b \succ c, b \succ c \succ a, c \succ a \succ b)$.

(Re-)ENTER LEWIS CARROLL

Dodgson's Rule: The winner of an election is the candidate who requires the fewest preference switches (adjacent) to become the winner.

Theorem (Bartholdi, Tovey, and Trick (BTT), 1989): It is NP-Hard to determine the winner under Dodgson's Method.

IMPRACTICALITY THEOREM

Kemeny's Rule involves finding an ordering that is "closest" to the voters' preferences (so if a beats b by 3 votes, then it costs 3 to reverse this). This rule is also hard to calculate.

IMPRACTICALITY THEOREM

Kemeny's Rule involves finding an ordering that is "closest" to the voters' preferences (so if a beats b by 3 votes, then it costs 3 to reverse this). This rule is also hard to calculate.

Definition. A voting system satisfies *neutrality* if it is symmetric in its treatment of the candidates.

Definition. A voting system satisfies *consistency* if, when two disjoint sets of voters agree on a candidate c, the union of voeters will also choose c.

IMPRACTICALITY THEOREM

Kemeny's Rule involves finding an ordering that is "closest" to the voters' preferences (so if a beats b by 3 votes, then it costs 3 to reverse this). This rule is also hard to calculate.

Definition. A voting system satisfies *neutrality* if it is symmetric in its treatment of the candidates.

Definition. A voting system satisfies *consistency* if, when two disjoint sets of voters agree on a candidate c, the union of voeters will also choose c.

Impracticality Theorem (BTT, 1989). For any voting system that satisfies
(a) neutrality
(b) consistency
(c) Condorcet winner
it is NP-Hard to determine the winner.

IMPLICATIONS

- There exist algorithms for fixed number of candidates.

IMPLICATIONS

- There exist algorithms for fixed number of candidates.
- Can run into problems if number of candidates and voters is reasonably large

IMPLICATIONS

- There exist algorithms for fixed number of candidates.
- Can run into problems if number of candidates and voters is reasonably large
- Gives limits to how quick can be solved (Saari and Merlin reduce Kemeny rule from obvious $n! \left(\begin{array}{c} n \\ 2 \end{array} \right) v$ algorithm to $O(n! + nv)$ algorithm. No use trying to reduce the $n!$.

IMPLICATIONS

- There exist algorithms for fixed number of candidates.
- Can run into problems if number of candidates and voters is reasonably large
- Gives limits to how quick can be solved (Saari and Merlin reduce Kemeny rule from obvious $n! \begin{pmatrix} n \\ 2 \end{pmatrix} v$ algorithm to $O(n! + nv)$ algorithm. No use trying to reduce the $n!$.
- Heuristics probably not useful in many applications.

IMPLICATIONS

- There exist algorithms for fixed number of candidates.
- Can run into problems if number of candidates and voters is reasonably large
- Gives limits to how quick can be solved (Saari and Merlin reduce Kemeny rule from obvious $n!\begin{pmatrix} n \\ 2 \end{pmatrix}v$ algorithm to $O(n! + nv)$ algorithm. No use trying to reduce the $n!$.
- Heuristics probably not useful in many applications.
- Difficult to analyze aspects that require characterization of Kemeny or Dodgson winner for arbitrary sizes

VOTING THEORY ISSUE: MANIPULABILITY

Sometimes a voter can get a preferred result by misrepresenting his preferences. (Example: plurality election between a, b, and c. Without you a and b are tied, and c way behind. You prefer c but instead vote to break tie between a and b).

Definition. A voting system satisfies *non-manipulability* if no voter can ever get a preferred result by misrepresenting his preferences.

Many impossibility theorems (example: No voting system satisfies anonymity, neutrality, Condorcet winner, and non-manipulability). Many, many papers on this: what if everyone is changing? Weakening of conditions, etc.

ALGORITHMIC ISSUE

Can it ever be hard to manipulate?

Yes!

ALGORITHMIC ISSUE

Can it ever be hard to manipulate?

Yes!

Definition. The *Copeland score* of a candidate is the number of pairwise contests won minus the number lost.
Definition. The *Second order Copeland score* of a candidate is the sum of the Copeland scores of each defeated candidate.

ALGORITHMIC ISSUE

Can it ever be hard to manipulate?

Yes!

Definition. The *Copeland score* of a candidate is the number of pairwise contests won minus the number lost.
Definition. The *Second order Copeland score* of a candidate is the sum of the Copeland scores of each defeated candidate.

Theorem (BTT 1989). It is NP-complete for a voter to determine how to manipulate an election under second order Copeland score.

There are others, with Single Tranferable Vote the most natural (Bartholdi and Orlin).

IMPLICATIONS

- There exist algorithms for fixed number of candidates or for fixed number of voters (depending on rule).

IMPLICATIONS

- There exist algorithms for fixed number of candidates or for fixed number of voters (depending on rule).
- Can run into problems if number of candidates and voters is reasonably large

IMPLICATIONS

- There exist algorithms for fixed number of candidates or for fixed number of voters (depending on rule).
- Can run into problems if number of candidates and voters is reasonably large
- Difficult to analyze manipulable instances since no characterization of such.

IMPLICATIONS

- There exist algorithms for fixed number of candidates or for fixed number of voters (depending on rule).
- Can run into problems if number of candidates and voters is reasonably large
- Difficult to analyze manipulable instances since no characterization of such.
- Heuristics probably *are* useful.

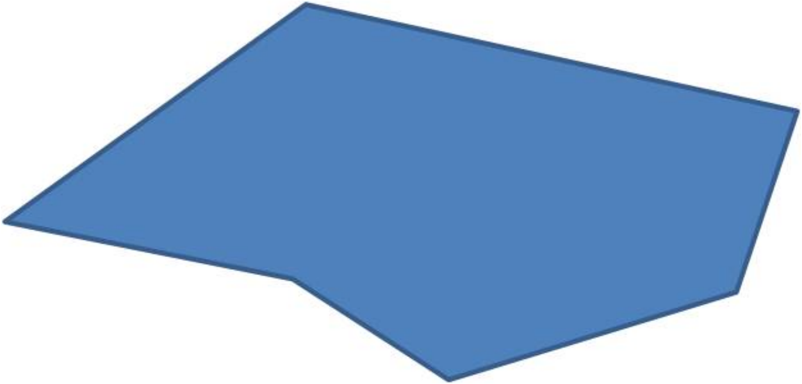
IMPLICATIONS

- There exist algorithms for fixed number of candidates or for fixed number of voters (depending on rule).
- Can run into problems if number of candidates and voters is reasonably large
- Difficult to analyze manipulable instances since no characterization of such.
- Heuristics probably *are* useful.
- Likelihood of manipulation now depends on both opportunity and the recognition of the opportunity.

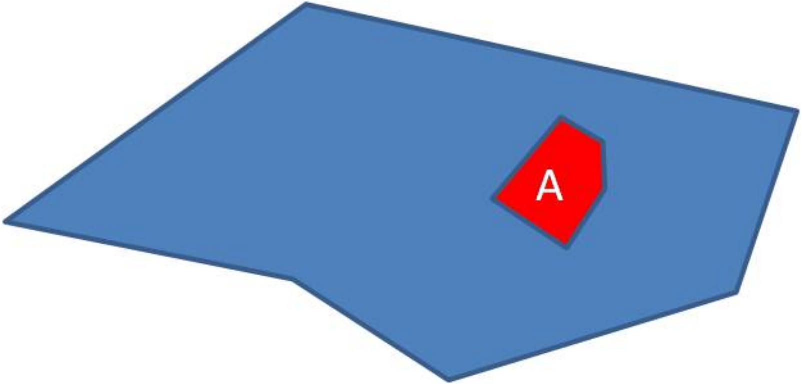
IMPLICATIONS

- There exist algorithms for fixed number of candidates or for fixed number of voters (depending on rule).
- Can run into problems if number of candidates and voters is reasonably large
- Difficult to analyze manipulable instances since no characterization of such.
- Heuristics probably *are* useful.
- Likelihood of manipulation now depends on both opportunity and the recognition of the opportunity.
- Matches up with intuitive feel for difficulty of problem.

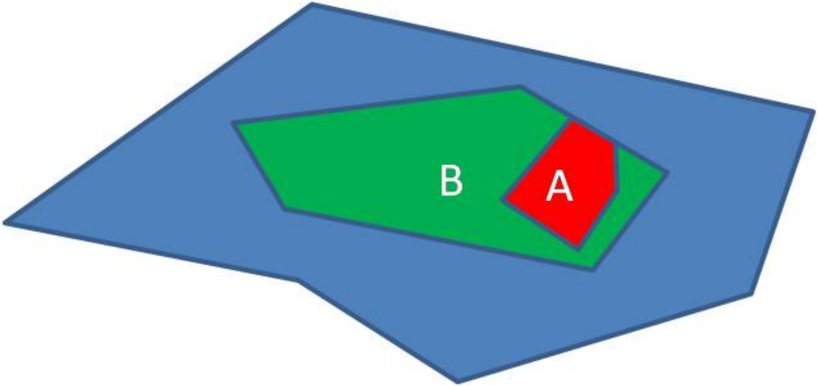
Opportunity/Recognition/Heuristics



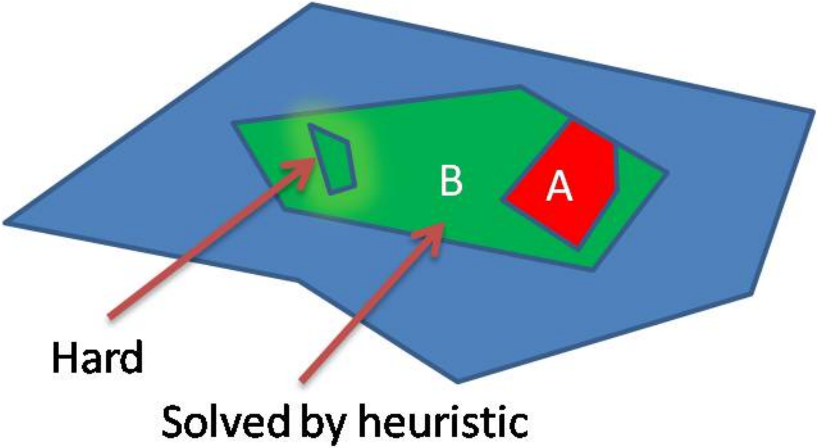
Opportunity/Recognition/Heuristics



Opportunity/Recognition/Heuristics



Opportunity/Recognition/Heuristics



HEURISTICS

Easy to come up with heuristics. For instance, a voter trying to elect c might try the following:

Inititialize Place c at the top of the preference order

HEURISTICS

Easy to come up with heuristics. For instance, a voter trying to elect c might try the following:

Inititialize Place c at the top of the preference order

Iterate Determine if any candidate can be placed in the next lower position (independent of other choices) without preventing c from winning. If so, place such a candidate in the next position; otherwise claim c cannot win

HEURISTICS

Easy to come up with heuristics. For instance, a voter trying to elect c might try the following:

Inititialize Place c at the top of the preference order

Iterate Determine if any candidate can be placed in the next lower position (independent of other choices) without preventing c from winning. If so, place such a candidate in the next position; otherwise claim c cannot win

HEURISTICS

Easy to come up with heuristics. For instance, a voter trying to elect c might try the following:

Inititialize Place c at the top of the preference order

Iterate Determine if any candidate can be placed in the next lower position (independent of other choices) without preventing c from winning. If so, place such a candidate in the next position; otherwise claim c cannot win

Algorithmic issue: For which rules is this heuristic guaranteed to work (i.e. will always correctly determine if c can win)?

HEURISTIC ANALYSIS

Denote an ordering P, where iPj mean i is ordered before j.

Theorem (BTT 1989). Greedy-Manipulation will find an ordering P that will make candidate c a winner or conclude that it is impossible for any voting scheme that can be represented as a function $S(P) : C \rightarrow \mathbf{R}$ that is both
– "responsive": a candidate with the largest $S(P, i)$ is a winner.
– "monotone": for any two preference orders P and P' and for any candidate i, $\{j : iP'j\} \subseteq \{j : iPj\}$ implies that $S(P', i) \leq S(P, i)$.

Shows Plurality, Borda, Copeland, and many others are manipulable quickly.

MANIPULATION BY GROUPS

Suppose you lead a group of 100 people and wish to tell them how to vote in order to get your prefered candidate c to win.

Plurality: Easy! Vote for candidate c. Don't even need to know others preferences: if it works, fine; otherwise you can't make c the winner (need more knowledge to elect your highest possible candidate).

Borda: Much harder. Clearly put c in first slot. But who in second (who will get $n-1$ points)? May have to have some people put a_1 in second slot and others put a_2 in second slot.

Manipulating Borda count by Groups is NP-complete (and needs multiple profiles).

MANIPULATION BY CHAIRS

Chairs of committees may have a number of powers:

Changing the Candidates

- Adding Candidates

MANIPULATION BY CHAIRS

Chairs of committees may have a number of powers:

Changing the Candidates

- Adding Candidates
- Deleting Candidates

MANIPULATION BY CHAIRS

Chairs of committees may have a number of powers:

Changing the Candidates

- Adding Candidates
- Deleting Candidates
- Partitioning Candidates

MANIPULATION BY CHAIRS

Chairs of committees may have a number of powers:

Changing the Candidates

- Adding Candidates
- Deleting Candidates
- Partitioning Candidates

MANIPULATION BY CHAIRS

Chairs of committees may have a number of powers:

Changing the Candidates

- Adding Candidates
- Deleting Candidates
- Partitioning Candidates

Changing the Voters

- Adding Voters

MANIPULATION BY CHAIRS

Chairs of committees may have a number of powers:

Changing the Candidates

- Adding Candidates
- Deleting Candidates
- Partitioning Candidates

Changing the Voters

- Adding Voters
- Deleting Voters

MANIPULATION BY CHAIRS

Chairs of committees may have a number of powers:

Changing the Candidates

- Adding Candidates
- Deleting Candidates
- Partitioning Candidates

Changing the Voters

- Adding Voters
- Deleting Voters
- Partitioning Voters

MANIPULATION BY CHAIRS

Chairs of committees may have a number of powers:

Changing the Candidates

- Adding Candidates
- Deleting Candidates
- Partitioning Candidates

Changing the Voters

- Adding Voters
- Deleting Voters
- Partitioning Voters

MANIPULATION BY CHAIRS

Chairs of committees may have a number of powers:

Changing the Candidates

- Adding Candidates
- Deleting Candidates
- Partitioning Candidates

Changing the Voters

- Adding Voters
- Deleting Voters
- Partitioning Voters

Many "fairness conditions" address the question of whether a voting rule is vulnerable to these sort of manipulations.

COMPARISON of PLURALITY and CONDORCET

Can also ask the algorithmic question: how can a chair determine *how* to optimally use power (BTT 1992).

Control by ...	Plurality	Condorcet
adding candidates	resistant	immune
deleting candidates	resistant	vulnerable
partitioning candidates	resistant	vulnerable
adding voters	vulnerable	resistant
deleting voters	vulnerable	resistant
partitioning voters	vulnerable	resistant

CONCLUSIONS

1. Algorithmic issues abound in voting theory.

CONCLUSIONS

1. Algorithmic issues abound in voting theory.
2. Voting rules differ in their computational needs

CONCLUSIONS

1. Algorithmic issues abound in voting theory.
2. Voting rules differ in their computational needs
3. Complexity analysis can point possible directions for research

CONCLUSIONS

1. Algorithmic issues abound in voting theory.
2. Voting rules differ in their computational needs
3. Complexity analysis can point possible directions for research
4. Many issues left to explore! And they have been over the last years, including

CONCLUSIONS

1. Algorithmic issues abound in voting theory.
2. Voting rules differ in their computational needs
3. Complexity analysis can point possible directions for research
4. Many issues left to explore! And they have been over the last years, including
   - More detailed complexity classes

CONCLUSIONS

1. Algorithmic issues abound in voting theory.
2. Voting rules differ in their computational needs
3. Complexity analysis can point possible directions for research
4. Many issues left to explore! And they have been over the last years, including
   - More detailed complexity classes
   - Average case behavior

CONCLUSIONS

1. Algorithmic issues abound in voting theory.
2. Voting rules differ in their computational needs
3. Complexity analysis can point possible directions for research
4. Many issues left to explore! And they have been over the last years, including
    - More detailed complexity classes
    - Average case behavior
    - Other hard/easy problems

INTERLUDE: Some Further History

1987. We write paper on NP-completeness to determine winner.

INTERLUDE: Some Further History

1987. We write paper on NP-completeness to determine winner. Rejected by *Journal of Economic Theory*, Accepted by *Social Choice and Welfare*

INTERLUDE: Some Further History

1987. We write paper on NP-completeness to determine winner. Rejected by *Journal of Economic Theory*, Accepted by *Social Choice and Welfare*

1987. We write paper on NP-completeness of manipulation.

INTERLUDE: Some Further History

1987. We write paper on NP-completeness to determine winner. Rejected by *Journal of Economic Theory*, Accepted by *Social Choice and Welfare*

1987. We write paper on NP-completeness of manipulation. Rejected by *Journal of Economic Theory*, Accepted by *Social Choice and Welfare*

1990. We write paper on controlling an election.

INTERLUDE: Some Further History

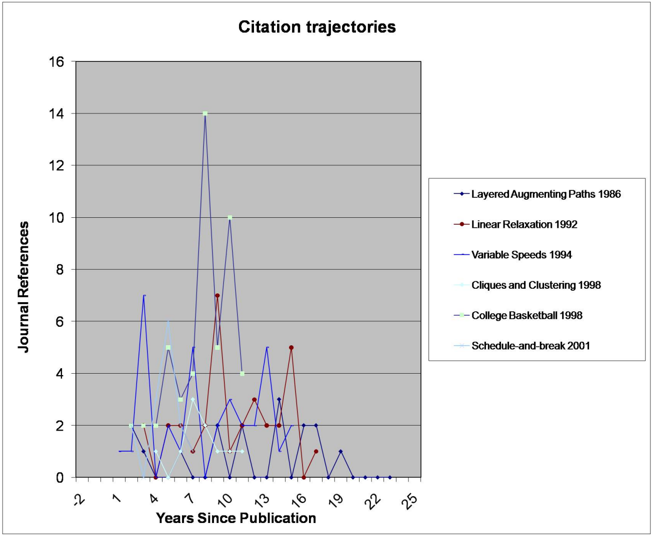
1987. We write paper on NP-completeness to determine winner. Rejected by *Journal of Economic Theory*, Accepted by *Social Choice and Welfare*

1987. We write paper on NP-completeness of manipulation. Rejected by *Journal of Economic Theory*, Accepted by *Social Choice and Welfare*

1990. We write paper on controlling an election. Rejected by *Econometrica* (in 2 weeks), Accepted in special issue of *Mathematical and Computer Modeling*

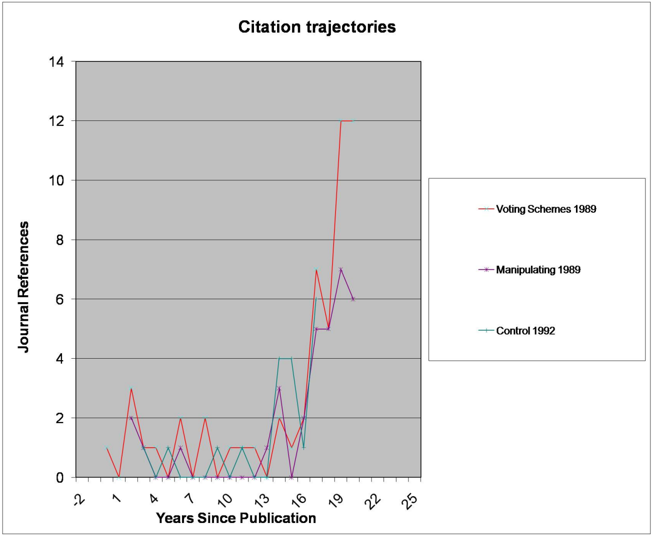
Citation History

Let's check the citation (SSI) history. “Normal” papers



Citation History

Let's check the citation (SSI) history. Voting papers



Current State

Currently the Winner NP-completeness and the Manipulation NP-completeness papers are number 5 and 8 *all-time* from *Social Choice and Welfare* in terms of google scholar citations (164 and 150). Only 3 papers in JET from 1989 have more cites.

Controlling an election is at 71, by far most cited ever in *Mathematics and Computer Modeling*.

Current State

Currently the Winner NP-completeness and the Manipulation NP-completeness papers are number 5 and 8 *all-time* from *Social Choice and Welfare* in terms of google scholar citations (164 and 150). Only 3 papers in JET from 1989 have more cites.

Controlling an election is at 71, by far most cited ever in *Mathematics and Computer Modeling*.

Thanks!

Current State

Currently the Winner NP-completeness and the Manipulation NP-completeness papers are number 5 and 8 *all-time* from *Social Choice and Welfare* in terms of google scholar citations (164 and 150). Only 3 papers in JET from 1989 have more cites.

Controlling an election is at 71, by far most cited ever in *Mathematics and Computer Modeling*.

Thanks!

Don't give up on what you think is a good, but unrecognized, research direction. Its day *may* come.

PRESENT: Agenda Control
Another type of chair manipulation

Suppose you give the chair the opportunity to determine the agenda in pairwise comparisons

"b will go against c and the winner will go against a"

PRESENT: Agenda Control
Another type of chair manipulation

Suppose you give the chair the opportunity to determine the agenda in pairwise comparisons

"b will go against c and the winner will go against a"

"a will go against b, a will go against c and the winners will go against each other"

PRESENT: Agenda Control
Another type of chair manipulation

Suppose you give the chair the opportunity to determine the agenda in pairwise comparisons

"*b* will go against *c* and the winner will go against *a*"

"*a* will go against *b*, *a* will go against *c* and the winners will go against each other"

Requirements: Pairwise comparisons; every candidate appears at least once; agenda set before preferences are known (unlike previous types of manipulation which assumed knowledge of others' preferences).

PRESENT: Agenda Control
Another type of chair manipulation

Suppose you give the chair the opportunity to determine the agenda in pairwise comparisons

"b will go against c and the winner will go against a"

"a will go against b, a will go against c and the winners will go against each other"

Requirements: Pairwise comparisons; every candidate appears at least once; agenda set before preferences are known (unlike previous types of manipulation which assumed knowledge of others' preferences).

Big Open problem: n candidates. a is chair's favorite candidate: can the chair set the agenda so that a wins as long as a is in the top cycle?

PRESENT: Agenda Control
Another type of chair manipulation

Suppose you give the chair the opportunity to determine the agenda in pairwise comparisons

"*b* will go against *c* and the winner will go against *a*"

"*a* will go against *b*, *a* will go against *c* and the winners will go against each other"

Requirements: Pairwise comparisons; every candidate appears at least once; agenda set before preferences are known (unlike previous types of manipulation which assumed knowledge of others' preferences).

Big Open problem: *n* candidates. *a* is chair's favorite candidate: can the chair set the agenda so that *a* wins as long as *a* is in the top cycle?

Medium open problem: 5 candidates. *a* is chair's favorite candidate: can the chair set the agenda so that *a* wins as long as *a* is in the top cycle?

PRESENT: Agenda Control
Another type of chair manipulation

Suppose you give the chair the opportunity to determine the agenda in pairwise comparisons

"b will go against c and the winner will go against a"

"a will go against b, a will go against c and the winners will go against each other"

Requirements: Pairwise comparisons; every candidate appears at least once; agenda set before preferences are known (unlike previous types of manipulation which assumed knowledge of others' preferences).

Big Open problem: n candidates. a is chair's favorite candidate: can the chair set the agenda so that a wins as long as a is in the top cycle?

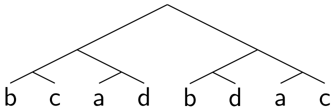
Medium open problem: 5 candidates. a is chair's favorite candidate: can the chair set the agenda so that a wins as long as a is in the top cycle? We'll see the 3 and 4 candidate cases.

Implementable Rules

Voting trees, with candidates at leaves.
Winner is determined by working up from leaves, using majority voting to choose between the two candidates.

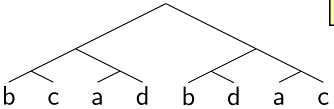
Implementable Rules

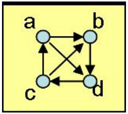
Voting trees, with candidates at leaves.
Winner is determined by working up from leaves, using majority voting to choose between the two candidates.



Implementable Rules

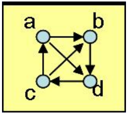
Voting trees, with candidates at leaves.
Winner is determined by working up from leaves, using majority voting to choose between the two candidates.

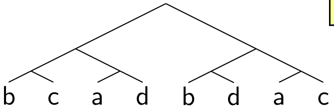


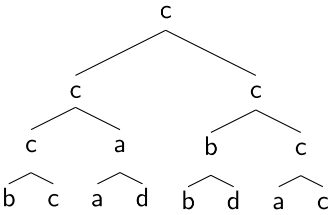


Implementable Rules

Voting trees, with candidates at leaves.
Winner is determined by working up from leaves, using majority voting to choose between the two candidates.

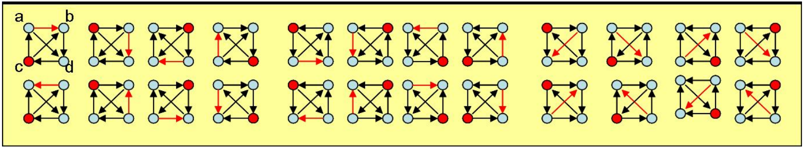






Implementable Rules

Given a set of tournaments $\mathcal{T}$, a voting tree defines a *rule* over $\mathcal{T}$.
Over all tournaments on 4 candidates with all candidates in top cycle, the previous tree gives the following rule:



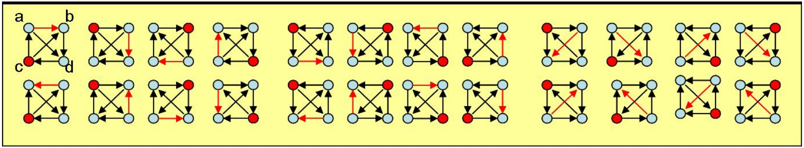
(This is actually the Copeland rule with 2nd-order Copeland tiebreaking)

Question

What rules are implementable by voting trees?

Implementable Rules

Given a set of tournaments $\mathcal{T}$, a voting tree defines a *rule* over $\mathcal{T}$.
Over all tournaments on 4 candidates with all candidates in top cycle, the previous tree gives the following rule:



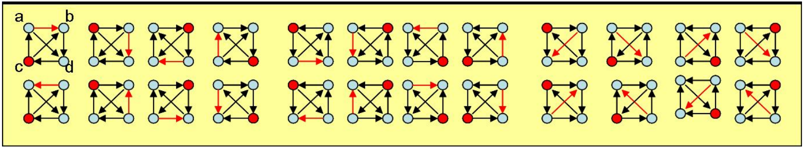
(This is actually the Copeland rule with 2nd-order Copeland tiebreaking)

Question

What rules are implementable by voting trees?

Implementable Rules

Given a set of tournaments $\mathcal{T}$, a voting tree defines a *rule* over $\mathcal{T}$. Over all tournaments on 4 candidates with all candidates in top cycle, the previous tree gives the following rule:



(This is actually the Copeland rule with 2nd-order Copeland tiebreaking)

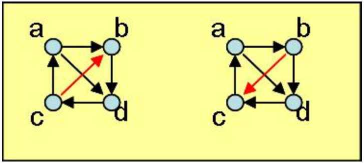
Question

What rules are implementable by voting trees?

In the economics literature, this is known as implementation by backwards induction and is a key open problem in mechanism design.

Implementable Rules

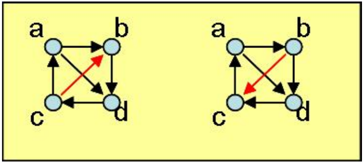
Assumption: all candidates appear in tree (rule is onto).
Clearly, rule must choose from top cycle of each tournament (including choosing Condorcet winner if it exists)
Sufficient?



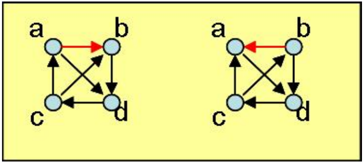
all 16 pairs of winners are implementable.

Implementable Rules

Assumption: all candidates appear in tree (rule is onto).
Clearly, rule must choose from top cycle of each tournament (including choosing Condorcet winner if it exists)
Sufficient?



all 16 pairs of winners are implementable.



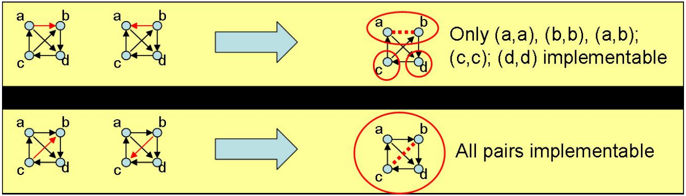
only $(a, a), (b, b), (c, c), (d, d)$ and (a, b) are implementable.

Pairwise Conjecture

Conjecture (Pairwise Conjecture)

A rule defined over all tournaments of n candidates is implementable if and only if it is implementable over all pairs of tournaments. (Srivasta and Trick, 1996)

Srivastava and Trick also give necessary and sufficient conditions for a rule to be implementable over a pair of tournaments.



If true, then this implies there is an agenda-control tree for all n.

Computational Procedure

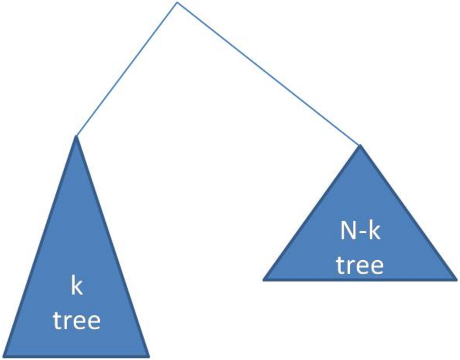
After thirteen years, little progress on conjecture (but no counterexamples either!)

Want a computational procedure to provide verification (or find counterexample)

Algorithm to generate all small rules over small number of candidates (number of rules increases quickly with number of candidates)

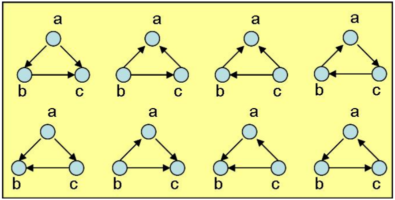
Computational Procedure: Dynamic Programming

Complete enumeration to generate minimum size trees for each rule.
Generate all trees of size up to N
Generate $N+1$ size trees by combining



There has to be a better way!

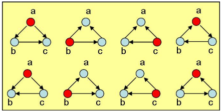
Rules on 3 Candidates

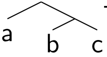


There are 8 tournaments on 3 candidates, so there are $3^8 = 6561$ rules over these tournaments. Of these, only 9 rules are Condorcet. The Pairwise conjecture requires each of these 9 to be implementable, and the computational procedure shows that to be the case.

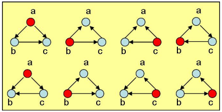
Tournaments on 3 Candidates

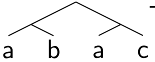
Always choose Condorcet candidate if it exists. Else:



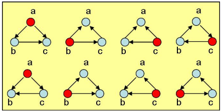


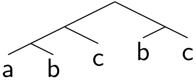
Type I: Choose *a*





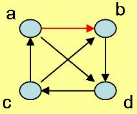
Type II: Choose candidate that beats *a*





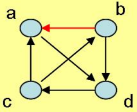
Type III: Choose candidate that loses to *a*

Structure of 4 Candidate Tournaments



Copeland: 2
a, c
Copeland: 1
b, d

Partner



Copeland: 2
b, c
Copeland: 1
a, d

Define
w_1: c
w_2: a(b)
l_1: b(a)
l_2: d

Implementable
w_1 : (c,c)
w_2 : (a,b)
l_2 : (d,d)

(w_2,l_1) : (a,a)
(l_1,w_2) : (b,b)

Number of Rules on 4 Candidates

- There are $2^6 = 64$ tournaments on 4 candidates, so $4^{64} = 3.4 * 10^{38}$ rules.

Number of Rules on 4 Candidates

- There are $2^6 = 64$ tournaments on 4 candidates, so $4^{64} = 3.4 * 10^{38}$ rules.
- This is reduced to $4^{24}3^8 = 1.8 * 10^{18}$ rules that choose from the top cycle (including Condorcet winners).

Number of Rules on 4 Candidates

- There are $2^6 = 64$ tournaments on 4 candidates, so $4^{64} = 3.4 * 10^{38}$ rules.
- This is reduced to $4^{24}3^8 = 1.8 * 10^{18}$ rules that choose from the top cycle (including Condorcet winners).
- Since the 24 tournaments with four candidates can be divided into pairs for which only 5 of the 16 possible pairs of winners is implementable, if the Pairwise Conjecture is true, there are $5^{12}3^8 =$ 1,601,806,640,625 implementable rules.

Number of Rules on 4 Candidates

- There are $2^6 = 64$ tournaments on 4 candidates, so $4^{64} = 3.4 * 10^{38}$ rules.
- This is reduced to $4^{24}3^8 = 1.8 * 10^{18}$ rules that choose from the top cycle (including Condorcet winners).
- Since the 24 tournaments with four candidates can be divided into pairs for which only 5 of the 16 possible pairs of winners is implementable, if the Pairwise Conjecture is true, there are $5^{12}3^8 =$ 1,601,806,640,625 implementable rules.
- If we ignore the tournaments with just 3 candidates in the top cycle (but still require Condorcet rules), that gives $5^{12} =$ 244,140,625 which is at least conceivable for our computational approach to find.

Number of Rules on 4 Candidates

- There are $2^6 = 64$ tournaments on 4 candidates, so $4^{64} = 3.4 * 10^{38}$ rules.
- This is reduced to $4^{24}3^8 = 1.8 * 10^{18}$ rules that choose from the top cycle (including Condorcet winners).
- Since the 24 tournaments with four candidates can be divided into pairs for which only 5 of the 16 possible pairs of winners is implementable, if the Pairwise Conjecture is true, there are $5^{12}3^8 =$ 1,601,806,640,625 implementable rules.
- If we ignore the tournaments with just 3 candidates in the top cycle (but still require Condorcet rules), that gives $5^{12} = 244{,}140{,}625$ which is at least conceivable for our computational approach to find.
- Of these 4096 choose among the Copeland winners, and 1 chooses only Copeland losers (interesting to find these).

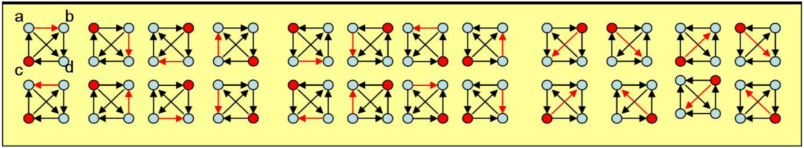
Results so far

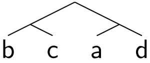
We have found about 66,835,958 rules so far, 3933 of the Copeland winner (out of 4096), and the Copeland Loser Rule (up to 31 leaves in the tree).

Size	Number	Copeland
4	15	3
5	102	0
6	424	0
7	1104	0
8	2377	19
9	5486	4
10	11232	18
11	21768	36
12	40420	36
13	70600	96
14	116670	60
15	187560	96
16	294510	240
17	439102	192

Size	Number	Copeland
18	633986	138
19	895648	292
20	1231551	368
21	1655920	148
22	2188704	240
23	2829882	318
24	3595685	276
25	4464020	296
26	5428012	224
27	6468312	220
28	7542497	366
29	8613668	88
30	9610118	76
31	10486540	84

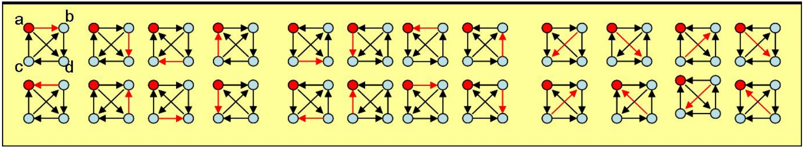
Smallest Rule, Choice from Copeland Winners

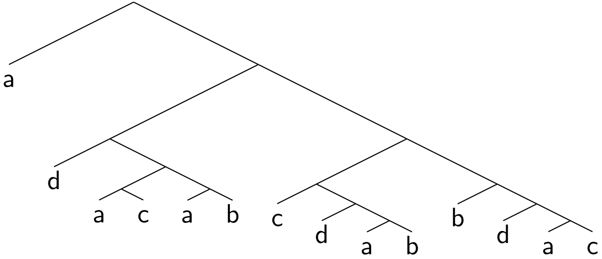




4 Node, Top Cycle, Lexicographic Tiebreak

Always choose a

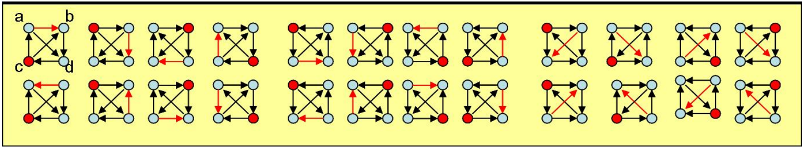


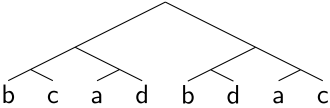


Compare with 3 candidate case! Agenda manipulation possible, but obvious due to complexity of the result.

4 Node, Copeland winner, 2nd Order Copeland tiebreak

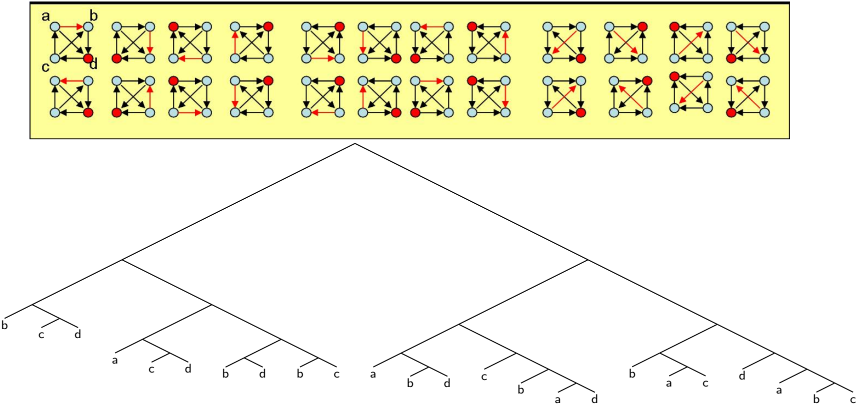
Always choose w_1 (neutral rule)





Copeland Loser in Top Cycle, 2nd Order Copeland tiebreak

Always choose l_2 (neutral rule)



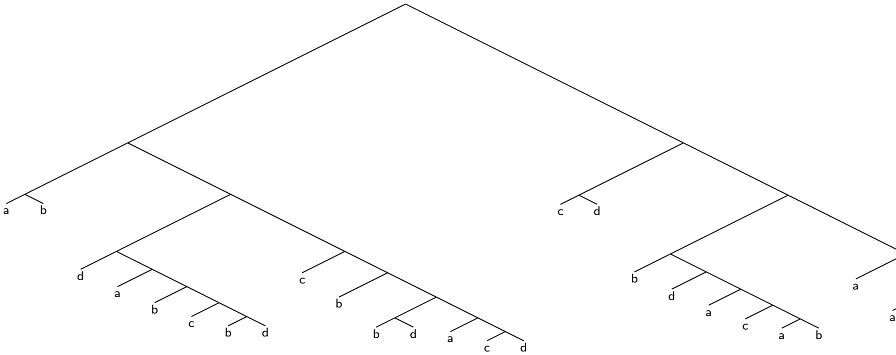
Copeland Winner in Top Cycle, Copeland loser tiebreak

Always choose w_2 (neutral rule)

Not yet found.

Sample 31 node tree

"Mainly" choose w_2 (correct 18 out of 24 times)



Conclusions and Future Directions

- Rules get very complicated very quickly (not surprising literature has few rules)

Conclusions and Future Directions

- Rules get very complicated very quickly (not surprising literature has few rules)
- Computational procedure reasonable approach for 4 candidates

Conclusions and Future Directions

- Rules get very complicated very quickly (not surprising literature has few rules)
- Computational procedure reasonable approach for 4 candidates

- Improving procedure: use of symmetry

Conclusions and Future Directions

- Rules get very complicated very quickly (not surprising literature has few rules)
- Computational procedure reasonable approach for 4 candidates

- Improving procedure: use of symmetry
- Characterization of rules with small trees
- Bounds on tree size

Conclusions and Future Directions

- Rules get very complicated very quickly (not surprising literature has few rules)
- Computational procedure reasonable approach for 4 candidates

- Improving procedure: use of symmetry
- Characterization of rules with small trees
- Bounds on tree size

Conclusions and Future Directions

- Rules get very complicated very quickly (not surprising literature has few rules)
- Computational procedure reasonable approach for 4 candidates

- Improving procedure: use of symmetry
- Characterization of rules with small trees
- Bounds on tree size
- Would appreciate someone proving or disproving Pairwise Conjecture!

FUTURE: Future Directions

- Lots more complexity in economics and finance

FUTURE: Future Directions

- Lots more complexity in economics and finance
  - Hiding toxic assets in finance: Arora, Barak, Brunemeier and Ge (2009): important since choice of instance is endogenous.

FUTURE: Future Directions

- Lots more complexity in economics and finance
  - Hiding toxic assets in finance: Arora, Barak, Brunemeier and Ge (2009): important since choice of instance is endogenous.
- Need for more algorithms

FUTURE: Future Directions

- Lots more complexity in economics and finance
    - Hiding toxic assets in finance: Arora, Barak, Brunemeier and Ge (2009): important since choice of instance is endogenous.
- Need for more algorithms
    - Polyhedral characterizations of voting rules (cf. Groetschel and Wakabayashi on linear ordering polytope)

FUTURE: Future Directions

- Lots more complexity in economics and finance
  - Hiding toxic assets in finance: Arora, Barak, Brunemeier and Ge (2009): important since choice of instance is endogenous.
- Need for more algorithms
  - Polyhedral characterizations of voting rules (cf. Groetschel and Wakabayashi on linear ordering polytope)
  - Integer/constraint programming approaches to hard problems

FUTURE: Future Directions

- Lots more complexity in economics and finance
  - Hiding toxic assets in finance: Arora, Barak, Brunemeier and Ge (2009): important since choice of instance is endogenous.
- Need for more algorithms
  - Polyhedral characterizations of voting rules (cf. Groetschel and Wakabayashi on linear ordering polytope)
  - Integer/constraint programming approaches to hard problems
  - Faster algorithms for mechanism design

FUTURE: Future Directions

- Lots more complexity in economics and finance
  - Hiding toxic assets in finance: Arora, Barak, Brunemeier and Ge (2009): important since choice of instance is endogenous.
- Need for more algorithms
  - Polyhedral characterizations of voting rules (cf. Groetschel and Wakabayashi on linear ordering polytope)
  - Integer/constraint programming approaches to hard problems
  - Faster algorithms for mechanism design
- More and more realistic preference restrictions (beyond single peakedness)

FUTURE: Future Directions

- Lots more complexity in economics and finance
  - Hiding toxic assets in finance: Arora, Barak, Brunemeier and Ge (2009): important since choice of instance is endogenous.
- Need for more algorithms
  - Polyhedral characterizations of voting rules (cf. Groetschel and Wakabayashi on linear ordering polytope)
  - Integer/constraint programming approaches to hard problems
  - Faster algorithms for mechanism design
- More and more realistic preference restrictions (beyond single peakedness)
- Need for more computational work (cf. Walsh's tutorial on Tuesday): where is the complexity?

FUTURE: Future Directions

- Lots more complexity in economics and finance
  - Hiding toxic assets in finance: Arora, Barak, Brunemeier and Ge (2009): important since choice of instance is endogenous.
- Need for more algorithms
  - Polyhedral characterizations of voting rules (cf. Groetschel and Wakabayashi on linear ordering polytope)
  - Integer/constraint programming approaches to hard problems
  - Faster algorithms for mechanism design
- More and more realistic preference restrictions (beyond single peakedness)
- Need for more computational work (cf. Walsh's tutorial on Tuesday): where is the complexity?
- Need for more real instances: does any of this happen in practice?

FUTURE: Future Directions

- Lots more complexity in economics and finance
  - Hiding toxic assets in finance: Arora, Barak, Brunemeier and Ge (2009): important since choice of instance is endogenous.
- Need for more algorithms
  - Polyhedral characterizations of voting rules (cf. Groetschel and Wakabayashi on linear ordering polytope)
  - Integer/constraint programming approaches to hard problems
  - Faster algorithms for mechanism design
- More and more realistic preference restrictions (beyond single peakedness)
- Need for more computational work (cf. Walsh's tutorial on Tuesday): where is the complexity?
- Need for more real instances: does any of this happen in practice?
- Breaking out of OR/CS and changing economics/finance. We have useful formalisms of "bounded rationality".

Questions or Comments?