COMPUTATIONAL SOCIAL CHOICE: BETWEEN VOTING THEORY AND MULTI-AGENT SYSTEMS

Francesca Rossi
Outline

- Preferences and multi-agent preference aggregation
- Voting theory
- Computational social choice
- Computational aspects
  - Resistance to manipulation
- Modelling preferences compactly
  - soft constraints
  - CP nets
- Exploiting compact preference modelling
  - sequential voting

IFAAMAS summer school -- Beijing, August 2013
Why preferences?

○ An intelligent system must be able to handle soft information
  ● different levels of preference or rejection
  ● several levels of tolerance
  ● vagueness
  ● imprecision

○ Information may be non-crisp
  ● intrinsically: the world is not binary
  ● due to information which is only partially available
Preferences

- Ubiquitous in real life
  - I prefer Venice to Rome
- A more tolerant way to set some constraints over the possible scenarios
  - I prefer a blue car
- Constraints can be used when we know what to accept or reject
  - I don’t want to spend more than X
- If all constraints, possibly
  - no solution, or
  - too many of them, all apparently equally good
- Some problems are naturally modelled with preferences
  - I don’t like meat, and I prefer fish to cheese
- Constraints and preferences may be present in the same problem
  - Configuration, timetabling, etc.
Example: University timetabling

Professor

Constraints

I cannot teach on Wednesday afternoon.

I prefer not to teach early in the morning, nor on Friday afternoon.

Preferences

Administration

Constraints

Lab C can fit only 120 students.

Better to not leave 1-hour holes in the day schedule.

Preferences
Several kinds of preferences

- **Positive (degrees of acceptance)**
  - I like ice cream

- **Negative (degrees of rejection)**
  - I don’t like strawberries

- **Unconditional**
  - I prefer taking the bus

- **Conditional**
  - I prefer taking the bus if it’s raining

- **Multi-agent**
  - I like blue, my husband likes green, what color do we buy the car?
Modelling preferences compactly

- **Preference ordering**: an ordering over the whole set of solutions (or candidates, or outcomes, …)

- Solution space with a combinatorial structure ➔ preferences over partial assignments, from which to generate the preference ordering over the solution space
Formalisms to model preferences

- Soft Constraints
  - Quantitative formalism
  - (Negative) preferences

- CP-nets (Conditional Preference Networks)
  - Qualitative formalism
  - Positive preferences

Two different ways to model compactly a preference ordering over a set of objects with a combinatorial structure
Two main ways to model preferences

- Quantitative
  - Numbers, or ordered set of objects
    - My preference for ice cream is 0.8, and for cake is 0.6
  - E.g., soft constraints

- Qualitative
  - Pairwise comparisons
    - Ice cream is better than cake
  - E.g., CP-nets

- Both very natural in some scenarios
- Different expressive power
- Different computational complexity for reasoning with them
Desiderata for an AI preference framework

- Expressive power
- Compactness
- Efficiency
- Suitability for multi-agent settings
Preferences for collective decision making in multi-agent systems

- Several agents
- Common set of possible decisions
- Each agent has its preferences over the possible decisions
- Goal: to choose one of the decisions, based on the preferences of the agents
  - Also a set of decisions, or a ranking over the decisions
- AI scenarios add: imprecision, uncertainty, complexity, etc.
Example

- Three friends need to decide what to cook for dinner
- 4 items (pasta, main, dessert, drink), 5 options for each
- Each friend has his/her own preferences over the meals

Agents = friends
Decisions = all possible dinners
Another example: Doodle

- Several time slots under consideration
- Partecipants accepts or reject each time slot
  - Very simple way to express preferences over time slots
  - Very little information communicated to the system
- Collective choice: a single time slot
  - The one with most acceptance votes from participants
Collective decision scenarios

- IT enabled social environments
  - People are connected all the time
  - Social networks allow us to share a large amount of information
  - More and more, we want to exploit this information to take collective decisions
  - With our friends, colleagues, etc.
- Also committees of agents
  - Search engines
  - Solvers
  - Classifiers
  - Product ranking agents, etc.
How to compute a collective decision?

- Let the agents vote by expressing their preferences over the possible decisions
- Aggregate the votes to get a single decision
- Let’s look at voting theory then!
Preferences vs. constraints

- Constraints are strict requirements
- Preferences as a way to provide more “tolerant” statements
Constraints

- Many real-life problems can be modelled via constraints
- Ex.:
  - “I need at least two bedrooms”
  - “I don’t want to spend more than 100K”
- Constraint = requirement = relation among objects (values for variables) of the problem
- Solution of a constraint problem = object choice (variable assignment) such that all constraints are satisfied
- Constraint programming offers
  - Natural modelling frameworks
  - Efficient solvers
  - Many application domains
    - Scheduling, timetabling, resource allocation, vehicle routing, ...

[Dechter, 2003; Rossi, Van Beek, Walsh, 2006]
Constraints are not flexible

- Constraints are useful when we have a clear yes/no idea
  - A constraint can either be satisfied or violated
- Sometimes, we have a less precise model of the real-life problem
  - Ex.: “Both a skiing and a beach vacation are fine, but I prefer skiing”
- If all constraints, possibly
  - No solution, or
  - Too many solutions, and equally satisfiable
Preferences are everywhere

- **Under-constrained problems** ⇀ many solutions ⇀ we want to choose among solutions

- **Over-constrained problems** ⇀ no solution ⇀ we want to find an acceptable assignment

- Problems which are **naturally modelled with preferences**

- Constraints and preferences may occur together
  - Ex.: configuration, timetabling
Example: University timetabling

Professor

Constraints

I cannot teach on Wednesday afternoon.

I prefer not to teach early in the morning, nor on Friday afternoon.

 Preferences

Administration

Constraints

Lab C can fit only 120 students.

Better to not leave 1-hour holes in the day schedule.

 Preferences

IFAAMAS summer school -- Beijing, August 2013
Several kinds of preferences

- Positive (degrees of acceptance)
  - “I like ice cream”

- Negative (degrees of rejection)
  - “I don’t like strawberries”

- Unconditional
  - “I prefer taking the bus”

- Conditional
  - “I prefer taking the bus if it’s raining”

- Multi-agent
  - “I like blue, my husband likes green, what color do we buy the car?”
Two main ways to model preferences

- Quantitative
  - Numbers or ordered set of objects
  - “My preference for ice cream is 0.8, and for cake is 0.6”
  - E.g., soft constraints

- Qualitative
  - Pairwise comparisons:
    - “Ice cream is better than cake”
  - E.g., CP-nets
Modelling preferences compactly

- Preference ordering: an ordering over the whole set of solutions (or candidates, or outcomes, ...)
- Solution space with a combinatorial structure → preferences over partial assignments, from which to generate the preference ordering over the solution space
Formalisms to model preferences

- Soft Constraints
  - Quantitative formalism
  - (Negative) preferences
- CP-nets (Conditional Preference Networks)
  - Qualitative formalism
  - Positive preferences

Two different ways to model compactly a preference ordering over a set of objects with a combinatorial structure
Multi-agent preferences

- Several agents (people, software agents, etc.) expressing their preferences over a set of objects (solutions, outcomes, etc.)
- We need to aggregate their preferences to obtain a result which satisfies all
- Result can be:
  - A preference ordering over the objects
  - A set of objects (optimal, winners, etc.)
- Preferences (one agent, or result) are expressed via partial orders
Why partial orders?

- When combining the preferences of different agents, incomparability as a means to resolve conflicts
- For a single agent:
  - some objects may naturally be incomparable
  - several possibly conflicting criteria
  - incomparability to model uncertainty
- Many AI formalisms to represent preferences generate partial orders or preorders
- \(\Rightarrow\) POs for describing both the preferences of an agent and the results of preference aggregation
Preference aggregation via voting theory

- We need to aggregate the preferences to
  - Test optimality
  - Find an optimal outcome
  - Order two outcomes
- A proposal: ask each agent dominance queries and then collect the votes as in an election → voting theory
- Voting theory provides machinery to aggregate preferences
  - Need to be adapted to AI scenarios
    - Automated systems
    - Large set of alternatives
    - Uncertainty
    - Incomparability
    - Elicitation
    - Computational issues
    - …
Terminology

- **Agent**
  - Usually assume odd number of agents to reduce ties

- **Vote**
  - Total order over outcomes (or candidates)
  - Extensions include indifference, incomparability, incompleteness

- **Profile**
  - Vote for each agent

- **Voting rule**
  - **Social choice**: mapping of a profile onto a winner(s)
  - **Social welfare**: mapping of a profile onto a total ordering over the candidates
Voting Procedures

- **n voters** (individuals, agents, players)
- **m candidates** (or alternatives)
- goal: collective choice among the candidates
- Each voter gives a **ballot**
  - the name of a single alternative,
  - a ranking (=linear orders of all alternatives …)
- **Profile**: a set of n ballots (one for each voter)
Voting Procedures

- The procedure defines
  - the valid ballots
  - how they are aggregated

- Different types of result
  - Resolute voting procedures: a single winner
  - Voting correspondences: a set of winners
  - Social welfare functions: an ordering over the set of candidates
Plurality

- Each voter states who the preferred candidate is
- Candidate who is preferred by the largest number of voters wins

- With just 2 candidates, equivalent to majority
  - very good rule to use
Plurality

- Ballot: 1 alternative
- Result: alternative(s) with the most vote(s)

Example:
- 6 voters
- Candidates:

<table>
<thead>
<tr>
<th>Ballot</th>
<th>Profile</th>
<th>Winner</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><img src="image1.png" alt="Candidate 1" /></td>
<td><img src="image2.png" alt="Winner" /></td>
</tr>
<tr>
<td></td>
<td><img src="image3.png" alt="Candidate 2" /></td>
<td></td>
</tr>
<tr>
<td></td>
<td><img src="image4.png" alt="Candidate 3" /></td>
<td></td>
</tr>
<tr>
<td></td>
<td><img src="image5.png" alt="Candidate 4" /></td>
<td></td>
</tr>
<tr>
<td></td>
<td><img src="image6.png" alt="Candidate 5" /></td>
<td></td>
</tr>
<tr>
<td></td>
<td><img src="image7.png" alt="Candidate 6" /></td>
<td></td>
</tr>
</tbody>
</table>
Criticisms of plurality

- Ignores preferences other than favourite
- Similar candidates can “split” the vote
- Encourages voters to vote tactically
  - “My candidate cannot win so I’ll put my second favorite first”
An example

- Consider the following vote
  - 49%: A>B>C
  - 20%: B>C>A
  - 20%: B>A>C
  - 11%: C>B>A → B>C>A → Now B wins!

- A wins a plurality vote
  - B is the Condorcet winner (pairwise winner against all)
  - C’s supporters can “manipulate” vote and get a “better” result by voting for B
Plurality with runoff

- Two rounds
  - Eliminate from the profiles all but the 2 candidates with most votes
  - Use plurality to choose the winner among the remaining 2 candidates

- Drawback: Requires voters to list all preferences or to vote twice

- Consider
  - 25 votes: A>B>C
  - 24 votes: B>C>A
  - 46 votes: C>A>B

  - 1st round: B eliminated
  - 2nd round: C>A by 70:25
  - C wins
Plurality with run off is not monotonic

- Moving a candidate up your ballot may not help him

Example

39  A>B>C
35  B>C>A
26  C>A>B
C is eliminated
A wins 65:35

If 10 B supporters put A first…

49  A>B>C
25  B>C>A
26  C>A>B
B is eliminated
C wins 51:49!

… this hurts A!
Plurality with runoff may favor abstention

- Consider again
  - 25 votes: A>B>C
  - 24 votes: B>C>A
  - 46 votes: C>A>B
  - C wins

- Two voters disliking C don’t vote
  - 23 votes: A>B>C
  - 24 votes: B>C>A
  - 46 votes: C>A>B
  - Different result
  - 1st round: A eliminated
  - 2nd round: B>C by 47:46
  - B wins
k-Approval

- **Ballot:** k favorite candidates
- **Procedure:**
  - for each voter
    - Each approved candidate gets one point
  - The score is the sum of all the points. The candidate(s) with the highest score win.
  - May need to tie break

- More informative balloting
2-approval example

Scores

Winner

1 voter 1 voter 1 voter 1 voters 1 voter
Approval

- **Ballot**: a set of favorite candidates

- **Procedure**:
  - for each voter
    - Each approved candidate gets one point
  - The score is the sum of all the points. The candidates with the highest score win.
  - May need to tie break
  - Named so by Weber in 1977
  - Widely used
  - Allows to express very different preferences
approval example

Scores

Winner

1 voter

1 voter

1 voter

1 voters

1 voter
Single transferable vote (STV)

- If one candidate has >50% votes, he is elected.
- Otherwise, the candidate with least votes is eliminated, and we start again.
  - His votes transferred (2nd placed candidate becomes 1st, etc.)
- Identical to plurality with runoff for n-1 candidates.

Example (100 voters):
- 39 votes: A>B>C>D
- 20 votes: B>A>C>D
- 20 votes: B>C>A>D
- 11 votes: C>B>A>D
- 10 votes: D>A>B>C

1. No one has majority
2. Eliminate D (transfer his votes to A)
3. 49 votes for A: still no majority
4. Eliminate C
5. Result: B wins with 51!
STV: example

- At least 4 candidates otherwise is like Plur. with run-off

<table>
<thead>
<tr>
<th>3 voters</th>
<th>3 voters</th>
<th>1 voters</th>
<th>2 voters</th>
<th>1 voters</th>
</tr>
</thead>
</table>

Winner
Borda

- Given m candidates
  - ith ranked candidate scores m-i
  - Candidate with greatest sum of scores wins

- Example
  - 42 votes: A>B>C>D
  - 26 votes: B>C>D>A
  - 15 votes: C>D>B>A
  - 17 votes: D>C>B>A
  - A’s score:
    \[ 3 \times 42 + 0 \times 26 + 0 \times 15 + 0 \times 17 \]
  - B’s score:
    \[ 2 \times 42 + 3 \times 26 + 1 \times 15 + 1 \times 17 \]
  - B wins
Borda rule: example

<table>
<thead>
<tr>
<th>Rank</th>
<th>Borda Count</th>
<th>Winner</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>3</td>
<td>Kermit</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>Miss Piggy</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Fozzie</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>Statler</td>
</tr>
</tbody>
</table>

1 voter 1 voter 1 voter 1 voters 1 voter
Positional voting rules

- Given vector of weights, \( <s_1, \ldots, s_m> \)
  - Candidate scores \( s_i \) for each vote in \( i \)th position
  - Candidate with greatest total score wins

- Generalizes many other voting rules
  - Borda is \( <m-1, m-2, \ldots, 0> \)
  - Plurality is \( <1, 0, \ldots, 0> \)
More voting rules

- **Approval**
  - Each voter approves between 1 and m-1 candidates
  - Candidate with most votes of approval wins

- **Cup (knockout)**
  - Tree of pairwise majority elections

- **Copeland**
  - The winner is the candidate that wins the most pairwise competitions
Difficult choice …

- So many voting rules to choose from..
- Which is best?
  - Social choice theory looks at the (desirable and undesirable) properties they possess (e.g. monotonicity)
  - Bottom line: with more than 2 candidates, there is no best voting rule
Axiomatic approach

- Define desired properties
  - E.g. monotonicity: improving votes for a candidate can only help him win
- Prove whether voting rule has this property
  - In some cases, as we shall see, we’ll be able to prove *impossibility results* (no voting rule has certain combinations of desirable properties)
- Choose one of the voting rules with the desired properties
Anonymous voting rule: names of voters irrelevant

Neutral voting rule: name of candidates irrelevant
Monotonicity

- Another desirable property of a voting rule
  - Monotonic voting rule: if a particular candidate wins, and a voter improves his vote in favor of this candidate, then he still wins

- We have already seen that plurality with run-off is not monotonic
May’s theorem

- Thm: With 2 candidates, a voting rule is anonymous, neutral and monotonic iff it is the plurality rule.


  - Since these properties are uncontroversial, this decides what to do with 2 candidates!
Condorcet’s paradox

- Collective preference may be cyclic
  - Even when individual preferences are not

- Consider 3 votes
  - A > B > C
  - B > C > A
  - C > A > B

- Condorcet cycle
  - Majority of voters prefer A to B, and prefer B to C, and prefer C to A!

Marie Jean Antoine Nicolas de Caritat, marquis de Condorcet (1743 – 1794)
Condorcet principle

- Condorcet winner
  - Candidate that beats every other in pairwise elections
  - In general, Condorcet winner may not exist
  - When he exists, he is unique

- Condorcet consistent
  - Voting rule that elects the Condorcet winner when he exists (e.g. Copeland rule)
Condorcet Principle

- Condorcet winner: an alternative that beats every other alternative in pairwise majority contests (if exists, unique)

| 49% | 20% | 20% | 11% |

Condorcet winner: 51% vs 89%
Condorcet principle

- Plurality rule is not Condorcet consistent
  - 35 votes: A>B>C
  - 34 votes: C>B>A
  - 31 votes: B>C>A

- B is the Condorcet winner, but plurality elects A
Other desirable properties

- **Free**
  - Every result is possible

- **Pareto (unanimity)**
  - If everyone prefers A to B, then A is preferred to B in the result

- **Independent to irrelevant alternatives**
  - Result between A and B only depends on the agents’ preferences between A and B (and not A and C, or C and B, ...)

- **Non-dictatorial**
  - Absence of a dictator
  - Dictator: voter whose vote always coincides with the result
Arrow’s theorem

- Thm: If there are at least two voters and three or more candidates, then it is impossible for any voting rule to be at the same time:
  - Pareto
  - Independent to irrelevant alternatives
  - Non-dictatorial

Nobel Prize in Economics
1972
Arrow’s theorem: weaker version

- If free & monotonic & IIA then Pareto

- Thm: If there are at least two voters and three or more candidates, then it is impossible for any voting rule to be:
  - Free
  - Monotonic
  - Independent to irrelevant alternatives
  - Non-dictatorial

- Weaker since
  - If Pareto then free
  - If free & Pareto & IIA then not necessarily monotonic
Arrow’s theorem: ways around

- How do we get “around” this impossibility
  - Limit domain
    - Only two candidates → majority
  - Limit votes
    - Single peaked votes
  - Limit properties
    - Drop IIA
    - ...

IFAAMAS summer school -- Beijing, August 2013
Single peaked preferences

- If there is a linear order of the candidates such that for each agent $i$ there is a candidate $x^*$ such that the more a candidate is far from $x^*$ in the order the more he is disliked.

- E.g. Linear order (A, B, C)
  - A>B>C
  - C>B>A
  - B>A>C
Manipulation

- **Constructive**
  - Can we change result so a given candidate wins?

- **Destructive**
  - Can we change result so a given candidate does not win?
Manipulation

- Means to manipulate
  - Our vote
  - The votes of a coalition of voters
  - The votes of other voters
    - Bribery
  - Chair person (control)
    - Agenda
    - Adding/deleting candidates
    - Adding/deleting votes
    - ..
The Gibbard-Satterthwaite theorem

- All “reasonable” voting rules are manipulable under weak assumptions
- One of social choice’s most fundamental results
- Only limited ways to escape GS
  - Restrict how people can vote
  - Ensure it is (computationally) difficult to manipulate result

IFAAMAS summer school -- Beijing, August 2013
The Gibbard-Satterthwaite Theorem

- **Assumptions**
  - 2 or more agents
  - 3 or more candidates
  - Voting rule is onto (free)
    - Every candidate is able to win
  - Voting rule is strategy-proof
    - Voting insincerely does not help
    - More precisely, an agent does not improve the result by mis-reporting his preferences
Gibbard-Satterthwaite

- Assumptions
  - 2 or more agents
  - 3 or more candidates
  - Voting rule is onto (=free)
  - Voting rule is strategy-proof

- Conclusion
  - Voting rule is dictatorial
    - One agent dictates the result

- Impossible to have ontoness + strategy-proof + non-dictatorial

- If onto there is either a cheater or a dictator!
Circumventing Gibbard Sattertwhaite

- Limit candidates
  - With 2 candidates, plurality is strategy-proof and lacks a dictator
- Restrict vote
  - For example, only permit single peaked votes
References for voting theory

Classical voting theory useful for Multi Agent Systems

- Many voting theory results can directly be useful for MAS
- Properties such as IIA, monotonicity, … make sense also in automated systems
- Axiomatic approach even more important when using automated agents
  - We are willing to use an automated system if certain properties are assured
- Some new issues …
Some new issues

- Not just total orders
- Large set of alternatives
- Compact preference modelling languages
  - Soft constraints, CP-nets, …
- Ballot language may be different from preference language
- Uncertainty, vagueness
- Elicitation effort
- Computational issues
  - How difficult it is to aggregate the preferences?
    - Easy is better
  - How difficult it is to manipulate the aggregation rule?
    - Difficult is better!
SOCIAL CHOICE
SCENARIOS VS. MULTI-AGENT SYSTEMS
Is social choice all we need for collective decision making in multi-agent systems?

After all …

- Voters = agents
- Candidates = decisions
- Preferences
- Winner = chosen decision

But …
Main differences

- In multi-agent AI scenarios, we usually have
  - Large sets of candidates (w.r.t. number of voters)
  - Combinatorial structure for candidate set
  - Knowledge representation formalisms to model preferences
  - Incomparability
  - Uncertainty, vagueness
  - Computational concerns
Large set of candidates

- In AI scenarios, usually the set of decisions is much larger than the set of agents expressing preferences over the decisions
  - Many web pages, few search engines
  - Many solutions of a constraint problem, few solvers
Combinatorial structure for the set of decisions

- Car (or PC, or camera) = several features, each with some instances

- Dinner example:
  - Three friends need to decide what to cook for dinner
  - 4 items (pasta, main, dessert, drink)
  - 5 options for each $\Rightarrow 5^4 = 625$ possible dinners

- In general: Cartesian product of several variable domains
  - Variables = items of the menu, domain= 5 options

IFAAMAS summer school -- Beijing, August 2013
Formalisms to model preferences compactly

- Preference ordering over a large set of decisions $\Rightarrow$ need to model them compactly
  - Otherwise too much space and time to handle such preferences

- Two examples:
  - Soft constraints
  - CP-nets
Preferences do not always induce a total order. Some items are naturally incomparable. Not because the information is missing. It depends also on the combinatorial structure (multi-criteria and Pareto dominance). To model uncertainty. As a means to resolve conflicts. Many AI formalisms to model preferences allow for partial orders.
Uncertainty, vagueness

- **Missing preferences**
  - Too costly to compute them
  - Privacy concerns
  - Ongoing elicitation process

- **Imprecise preferences**
  - Preferences coming from sensor data
  - Too costly to compute the exact preference
  - Estimates
Computational concerns

- We would like to avoid very costly ways to
  - Model the preferences
  - Compute the winner
  - Reason with the agents’ preferences

- On the other hand, we need a computational barrier against bad behaviours (such as manipulation)
Computational social choice

- Between multi-agent systems and social choice
  - AI, economics, mathematics, political science, etc.

- New concerns
  - Preference modelling
  - Algorithms, complexity
  - Uncertainty, preference elicitation

- Cross-fertilization in both directions
COMPUTATIONAL ASPECTS
Computational concerns about voting rules

- We want to avoid spending too much time to
  - Elicit preferences
  - Compute the winner (winners, ranking)
- On the other hand, we want a computational barrier against manipulation
  - Given the impossibility result, we want to avoid rules which are computationally easy to manipulate
Manipulation

- A rule is manipulable if an agent can gain by lying about its preferences
  - Gain = obtain a result which is more preferred by the agent
  - Strategy-proof rule: there is no incentive to misrepresent the preferences
- Gibbard-Satterwaite impossibility result
  - With at least three agents and two candidates, it is impossible for a voting rule to be at the same time surjective, strategy-proof, and non-dictatorial
  - We cannot give up non-dictatoriality and surjectivity!
Preventing manipulation?

- A **successful manipulation** is a way of misreporting one’s preferences that leads to a better result for oneself.
- Gibbard-Satterthwaite only tells us that successful manipulations exist for any rule.
  - It does not tell us what these manipulations are.
- Perhaps we can use complexity as a barrier?
  - Do voting rules exist for which manipulations are computationally hard to find?

[Bartholdi, Tovey, Trick 1989]
Intractable manipulation

- If manipulation is computationally intractable for F, then F might be considered resistant (albeit still not immune) to manipulation.

- Most interesting for voting procedures for which winner determination is tractable.
  - complexity gap between manipulation (undesired behaviour) and winner determination (desired functionality).

IFAAMAS summer school -- Beijing, August 2013
Manipulability as a decision problem

Manipulability(F)

Instance: Set of ballots for all but one voter; alternative x.

Question: Is there a ballot for the final voter such that x wins?

How difficult it is to answer this question?

- A manipulator would have to solve Manipulability(F) for all alternatives, in its preference ordering, to understand if there is a way he can get something better.
- If Manipulability(F) is computationally intractable, then manipulability may be considered less of a worry for procedure F.

Remark: We assume that the manipulator knows all the other ballots.

- This unrealistic assumption is reasonable for intractability results: if manipulation is intractable even under such favorable conditions, then all the better.
A formal computational problem

- The simplest version of the manipulation problem:
- **CONSTRUCTIVE-MANIPULATION:**
  - We are given a voting rule $R$, the (unweighted) votes of the other voters, and a candidate $p$
  - We are asked if we can cast our (single) vote to make $p$ win
- E.g. for the Borda rule:
  - Voter 1 votes $A > B > C$
  - Voter 2 votes $B > A > C$
  - Voter 3 votes $C > A > B$
- Borda scores are now: $A: 4$, $B: 3$, $C: 2$
- Can we (voter 4) make $B$ win with our single vote?
- Answer: YES. Vote $B > C > A$ (Borda scores: $A: 4$, $B: 5$, $C: 3$)
Plurality is easy to manipulate

**Manipulability(Plurality) ∈ P**

- Simply vote for x, the alternative to be made winner by means of manipulation. If manipulation is possible at all, this will work. Otherwise not.
- In general: Manipulability(F) ∈ P for any rule F with polynomial winner determination and polynomial number of ballots.

[Bartholdi, Tovey, Trick, 1989]
Manipulating Borda is easy

\[ \text{MANIPULABILITY(Borda)} \in \mathbb{P} \]

- Place \( x \) (the alternative to be made winner through manipulation) at the top of your ballot.
- Then inductively proceed as follows: Check if any of the remaining alternatives can be put next on the ballot without preventing \( x \) from winning. If yes, do so. If no, manipulation is impossible.

[Bartholdi, Tovey, Trick, 1989]
Bad news: Borda is easy to manipulate

- **Greedy** algorithm which finds a manipulation (if one exists)
  - Place $p$ at top of your vote
  - (Repeat) Check every other candidate to see if there is one that can be placed next in order without defeating $p$. If so, place him next, otherwise declare no manipulation exists

- Hence, we can decide if Borda can be manipulated in polynomial time
Bad news: plurality is easy to manipulate by coalition (or single voter)

- If want $p$ to win, the best thing to do is vote for $p$
  - If $p$ then wins, we have manipulated vote
  - If $p$ does not win, there is no manipulation

- Hence, we can decide if plurality can be manipulated in polynomial time
2nd order Copeland is difficult to manipulate

- **Theorem.** CONSTRUCTIVE-MANIPULATION is NP-complete for the second-order Copeland rule. [Bartholdi, Tovey, Trick 1989]
  - Copeland score = number of victories – number of defeats in pairwise contests
    - The winner is the candidate that wins the most pairwise competitions
  - Second order Copeland = tie-break with sum of Copeland scores of alternatives that are defeated
Inverse plurality is NP-hard to manipulate with 3 or more candidates

- **Plurality**
  - each voter has one vote, candidate with most votes wins
- **Inverse plurality**
  - each voter has one veto, candidate with fewest vetoes wins
  - Sometimes called anti-plurality or negative voting
- **Remember that plurality is easy to manipulate!**
Manipulating STV is difficult

- **MANIPULABILITY(STV) ∈ NP-complete**

- NP-membership is clear: checking whether a given ballot makes x win can be done in polynomial time (just try it, STV is polynomial to compute).

- NP-hardness: by reduction from another NP-complete problem (3-Cover). The basic idea is to build a large election instance introducing all sorts of constraints on the ballot of the manipulator, such that finding a ballot meeting those constraints solves a given instance of 3-Cover.

[Bartholdi, Orlin, 1991]
Good news: STV is hard to manipulate

- **Theorem.** CONSTRUCTIVE-MANIPULATION is NP-complete for the STV rule. [Bartholdi, Orlin 1991]

  - Single Transferable Vote repeatedly eliminates the least popular candidate
  - Votes for the least popular candidate are transferred to the next most preferred candidate
Coalitional constructive manipulation

- Manipulation by a coalition of agents
- Constructive: to make some candidate win (not just to get a better result)
### Weighted-coalitional constructive manipulation

<table>
<thead>
<tr>
<th>Number of candidates</th>
<th>2</th>
<th>3</th>
<th>4,5,6</th>
<th>≥ 7</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Borda</strong></td>
<td>P</td>
<td>NP-c</td>
<td>NP-c</td>
<td>NP-c</td>
</tr>
<tr>
<td><strong>veto</strong></td>
<td>P</td>
<td>NP-c*</td>
<td>NP-c*</td>
<td>NP-c*</td>
</tr>
<tr>
<td><strong>STV</strong></td>
<td>P</td>
<td>NP-c</td>
<td>NP-c</td>
<td>NP-c</td>
</tr>
<tr>
<td><strong>plurality with runoff</strong></td>
<td>P</td>
<td>NP-c*</td>
<td>NP-c*</td>
<td>NP-c*</td>
</tr>
<tr>
<td><strong>Copeland</strong></td>
<td>P</td>
<td>P*</td>
<td>NP-c</td>
<td>NP-c</td>
</tr>
<tr>
<td><strong>maximin</strong></td>
<td>P</td>
<td>P*</td>
<td>NP-c</td>
<td>NP-c</td>
</tr>
<tr>
<td><strong>randomized cup</strong></td>
<td>P</td>
<td>P*</td>
<td>P*</td>
<td>NP-c</td>
</tr>
<tr>
<td><strong>regular cup</strong></td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td><strong>plurality</strong></td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
</tr>
</tbody>
</table>
## Results for constructive manipulation

<table>
<thead>
<tr>
<th>Number of candidates</th>
<th>2</th>
<th>3</th>
<th>4,5,6</th>
<th>≥ 7</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Borda</strong></td>
<td>P</td>
<td>NP-c</td>
<td>NP-c</td>
<td>NP-c</td>
</tr>
<tr>
<td><strong>veto</strong></td>
<td>P</td>
<td>NP-c*</td>
<td>NP-c*</td>
<td>NP-c*</td>
</tr>
<tr>
<td><strong>STV</strong></td>
<td>P</td>
<td>NP-c</td>
<td>NP-c</td>
<td>NP-c</td>
</tr>
<tr>
<td><strong>plurality with runoff</strong></td>
<td>P</td>
<td>NP-c*</td>
<td>NP-c*</td>
<td>NP-c*</td>
</tr>
<tr>
<td><strong>Copeland</strong></td>
<td>P</td>
<td>P*</td>
<td>NP-c</td>
<td>NP-c</td>
</tr>
<tr>
<td><strong>maximin</strong></td>
<td>P</td>
<td>P*</td>
<td>NP-c</td>
<td>NP-c</td>
</tr>
<tr>
<td><strong>randomized cup</strong></td>
<td>P</td>
<td>P*</td>
<td>P*</td>
<td>NP-c</td>
</tr>
<tr>
<td><strong>regular cup</strong></td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td><strong>plurality</strong></td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
</tr>
</tbody>
</table>

Complexity of constructive CW-manipulation

[Conitzer, Sandholm, Lang, JACM 53 (3), 2007]
Destructive manipulation

- Exactly the same as constructive manipulation, except:
  - Instead of a preferred candidate, we now have a hated candidate
  - Our goal is to make sure that the hated candidate does not win (whoever else wins)

- Destructive manipulation can be easy even though constructive manipulation is hard

- If destructive manipulation is hard, then so is constructive manipulation
  - In other words: if constructive manipulation is easy, then also destructive manipulation is easy
    - To make A lose, we make somebody else win

- Reverse does not hold
  - There are voting rules that are polynomial to manipulate destructively but NP-hard to manipulate constructively
Weighted-coalitional destructive manipulation

<table>
<thead>
<tr>
<th>Number of candidates</th>
<th>2</th>
<th>≥ 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>STV</td>
<td>P</td>
<td>NP-c*</td>
</tr>
<tr>
<td>plurality with runoff</td>
<td>P</td>
<td>NP-c*</td>
</tr>
<tr>
<td>randomized cup</td>
<td>P</td>
<td>?</td>
</tr>
<tr>
<td>Borda</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>veto</td>
<td>P</td>
<td>P*</td>
</tr>
<tr>
<td>Copeland</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>maximin</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>regular cup</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>plurality</td>
<td>P</td>
<td>P</td>
</tr>
</tbody>
</table>
# Results for destructive manipulation

<table>
<thead>
<tr>
<th>Number of candidates</th>
<th>2</th>
<th>≥ 3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>STV</strong></td>
<td>P</td>
<td>NP-c*</td>
</tr>
<tr>
<td><em>plurality with runoff</em></td>
<td>P</td>
<td>NP-c*</td>
</tr>
<tr>
<td><em>randomized cup</em></td>
<td>P</td>
<td>?</td>
</tr>
<tr>
<td><strong>Borda</strong></td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td><strong>veto</strong></td>
<td>P</td>
<td>P*</td>
</tr>
<tr>
<td><strong>Copeland</strong></td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td><strong>maximin</strong></td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td><strong>regular cup</strong></td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td><strong>plurality</strong></td>
<td>P</td>
<td>P</td>
</tr>
</tbody>
</table>

Complexity of destructive CW-manipulation

[Conitzer, Sandholm, Lang, JACM 53 (3), 2007]
Bartholdi, Tovey, Trick. The computational difficulty of manipulating an election. Social Choice and Welfare, 6(3):227-241, 1989


Walsh. Where are the really hard manipulation problems? The phase transition in manipulating the veto rule. IJCAI-2009.
MODELLING PREFERENCES COMPACTLY: SOFT CONSTRAINTS
Preferences vs. constraints

- Constraints are strict requirements
- Preferences as a way to provide more “tolerant” statements
Constraints

- Many real-life problems can be modelled via constraints
- Ex.:
  - “I need at least two bedrooms”
  - “I don’t want to spend more than 100K”
- Constraint = requirement = relation among objects (values for variables) of the problem
- Solution of a constraint problem = object choice (variable assignment) such that all constraints are satisfied
- Constraint programming offers
  - Natural modelling frameworks
  - Efficient solvers
  - Many application domains
    - Scheduling, timetabling, resource allocation, vehicle routing, ...

[Dechter, 2003; Rossi, Van Beek, Walsh, 2006]

IFAAMAS summer school -- Beijing, August 2013
Constraints are not flexible

- Constraints are useful when we have a clear yes/no idea
  - A constraint can either be satisfied or violated
- Sometimes, we have a less precise model of the real-life problem
  - Ex.: “Both a skiing and a beach vacation are fine, but I prefer skiing”
- If all constraints, possibly
  - No solution, or
  - Too many solutions, and equally satisfiable
Soft Constraints: the c-semiring framework

- Variables \( \{X_1, \ldots, X_n\} = X \)
- Domains \( \{D(X_1), \ldots, D(X_n)\} = D \)
- Soft constraints
  - each constraint involves some of the variables
  - a preference is associated with each assignment of the variables
- Set of preferences \( A \)
  - Totally or partially ordered (induced by +)
  - Combination operator (x)
  - Top and bottom element \((1, 0)\)
  - Formally defined by a c-semiring \( <A, +, x, 0, 1> \)

Soft constraints

- Soft constraint: a pair $c = <f, con>$ where:
  - Scope: $con = \{X^c_1, ..., X^c_k\}$ subset of $X$
  - Preference function:
    $$f: D(X^c_1) \times ... \times D(X^c_k) \rightarrow A$$
    tuple $(v_1, ..., v_k) \rightarrow p$ preference

- Hard constraint: a soft constraint where for each tuple $(v_1, ..., v_k)$
  - $f(v_1, ..., v_k) = 1$ the tuple is allowed
  - $f(v_1, ..., v_k) = 0$ the tuple is forbidden
Soft Constraints: the C-semiring framework

- Some properties:
  - for all \( a \) in \( A \), \( 0 \leq a \leq 1 \)
  - for all \( a,b \) in \( A \), \( a \times b \leq a \)
  - \( <A,\leq> \) lattice
    - \( + \) is lub
    - \( x \) is glb if \( x \) idempotent
  - \( + \) and \( x \) monotone on \( \leq \)
Complete assignments and their evaluation

- Complete assignment: one value for each variable
- Global evaluation: preference associated to a complete assignment
- How to obtain a global evaluation?
  - By combining (via $x$) the preferences of the partial assignments given by the constraints
Example: weighted constraints

- \( A = N \cup +\infty, + = \text{min}, x = +, 0 = +\infty, 1 = 0 \)

- Values in \([0, +\infty]\)
  - Best value = 0
  - Worst value = +\infty

- Comparison with min
  - A better than B iff \( \min(A, B) = A \)

- Composition with +
  - Goal is to minimize sum
Example: fuzzy constraints

- \( A = [0, 1], + = \max, x = \min, 0 = 0, 1 = 1 \):
  - Preferences between 0 and 1
  - Higher values denote better preferences
    - 0 is the worst preference
    - 1 is the best preference
  - Combination is taking the smallest value

- Optimization criterion = maximize the minimum preference

Pessimistic approach, useful in critical application (e.g., space and medical settings)

[Fuzzy CSPs: Schiex UAI' 92, Ruttkay FUZZ-IEEE '94]

IFAAMAS summer school -- Beijing, August 2013
Fuzzy-SCSP example

{Fish, Meat} → {White, red}

Main Course → Wine

(Fish, white) → 1
(Fish, red) → 0.8
(Meat, white) → 0.3
(Meat, red) → 0.7

{12 pm, 1 pm} → {2 pm, 3 pm}

Lunch → Swim

(12 pm, 2 pm) → 1
(12 pm, 3 pm) → 1
(1 pm, 2 pm) → 0
(1 pm, 3 pm) → 1

Solution S

Lunch= 1 pm
Main course = meat
Wine= white
Swim = 2 pm
pref(S)=min(0.3,0)=0

Solution S'

Lunch= 12 pm
Main course = fish
Wine= white
Swim = 2 pm
pref(S)=min(1,1)=1

Fuzzy semiring

$S = <A , + , \times , 0, 1>$

$S_{FCSP} = <[0,1],\max,\min,0,1>$
Instances of semiring-based soft constraints

- Each instance is characterized by a c-semiring \(<A, +, \times, 0, 1>\)
- Classical constraints: \(<\{0,1\}, \text{logical or}, \text{logical and}, 0, 1>\)
  - Satisfy all constraints
- Fuzzy constraints: \(<[0,1], \text{max}, \text{min}, 0, 1>\)
  - Maximize the minimum preference
- Lexicographic CSPs: \(<[0,1]^k, \text{lex-max}, \text{min}, 0^k, 1^k>\)
  - Order the preferences lexicographically and then maximize the minimum preference
- Weighted constraints (N): \(<\mathbb{N} \cup +\infty, \text{min}, +, +\infty, 0>\)
  - Minimize the sum of the costs (naturals)
- Weighted constraints (R): \(<\mathbb{R} \cup +\infty, \text{min}, +, +\infty, 0>\)
  - Minimize the sum of the costs (reals)
- Max CSP: weight =1 if constraint is not satisfied and 0 if satisfied
  - Minimize the number of violated constraints
- Probabilistic constraints: \(<[0,1], \text{max}, \times, 0, 1>\)
  - Maximize the joint probability of being a constraint of the real problem
- Valued CSPs: any totally ordered c-semiring
- Multi-criteria problems: Cartesian product of semirings
Multi-criteria problems

- One semiring for each criteria
- Given $n$ c-semirings $S_i = \langle A_i, +_i, x_i, 0_i, 1_i \rangle$, we can build the c-semiring $\langle A_1, \ldots, A_n \rangle, +, x, \langle 0_1, \ldots, 0_n \rangle, \langle 1_1, \ldots, 1_n \rangle$
- $+$ and $x$ obtained by pointwise application of $+_i$ and $x_i$ on each semiring
- A tuple of values associated with each variable instantiation
- A tuple is better than another if it is better or equal on all elements, and better in at least one
- A partial order even if all the criteria are totally ordered
  - Pareto-like approach
Example

- The problem: choosing a route between two cities
- Each piece of highway has a preference and a cost
- We want to both minimize the sum of the costs and maximize the preference
- Semiring: by putting together one fuzzy semiring and one weighted semiring:
  - $<[0,1],\text{max},\text{min},0,1>$
  - $<\mathbb{N},\text{min},+,+\infty,0>$
- Best solutions: routes such that there is no other route with a better semiring value
  - $<0.8,10>$ is better than $<0.7,15>$
- Two total orders, but the resulting order is partial:
  - $<0.6,10>$ and $<0.4,5>$ are not comparable

IFAAMAS summer school -- Beijing, August 2013
Solution ordering

- A soft CSP induces an ordering over the solutions, from the ordering of the semiring
- Totally ordered semiring $\Rightarrow$ total order over solutions (possibly with ties)
- Partially ordered semiring $\Rightarrow$ total or partial order over solutions (possibly with ties)
- Any ordering can be obtained!
Expressive power

- \( A \rightarrow B \) iff from a problem \( P \) in \( A \) it is possible to build in \textit{polynomial} time a problem \( P' \) in \( B \) s.t. the optimal solutions are the same (but not necessarily the solution ordering!)
  - \( B \) is at least as expressive as \( A \)

- \( A \rightarrow B \) iff from a problem \( P \) in \( A \) it is possible to build in \textit{polynomial} time a problem \( P' \) in \( B \) s.t. \( \text{opt}(P') \subseteq \text{opt}(P) \)
Expressive power

Semiring-based

Valued

weighted_R

Prob

weighted_N

Fuzzy → Lexicographic

Classical
Interesting questions for soft CSPs

- Find an optimal solution
- Find the next solution in a linearization of the solution ordering
- Is $s$ an optimal solution?
- Is $s$ better than $s'$?
Finding an optimal solution

- Difficult in general
  - Branch and bound + constraint propagation
  - Local search
  - Bucket elimination
  - ...
- Easy for some classes of soft constraints
- Ex.: tree-shaped problems
  - Bucket elimination: directional arc-consistency + backtrack-free search
  - Also for problems with bounded treewidth
Finding the next solution

- Next where? In a linearization of the solution ordering
- Ties and incomparable sets should be linearized (any way is fine)
- Difficult for CSPs in general (so also for SCSPs)
- At least as difficult as finding an optimal solution
- Easy for tree-shaped CSPs and tree-shaped fuzzy CSPs
- Difficult for tree-shaped weighted CSPs

[Brafman, Rossi, Venable, Walsh, 2009]

IFAAMAS summer school -- Beijing, August 2013
Is \( s \) an optimal solution?

- **Difficult in general**: same complexity as finding an optimal solution
  - We have to find the optimal preference level
  - Easy for classical CSPs (optimal preference level is 1)
Is $s$ better then $s'$?

- **Easy**: Linear in the number of constraints
  - Compute the two preference levels and compare them
  - Assumption: $+$ and $x$ easy to compute
Systematic search : Branch and bound

- Backtracking \(\rightarrow\) Branch and Bound

- Main idea:
  - visit each assignment that may be a solution
  - skip only assignments that are shown to be dominated by others

- **Search tree** to represent the space of all assignments
Systematic search: Branch and bound

- **Lower bound** = preference of best solution so far (0 at the beginning)

- **Upper bound for each node**: upper bound to the preference of any assignment in the subtree rooted at the node

- If **ub is worst than lb** ⇒ prune subtree

IFAAMAS summer school -- Beijing, August 2013
\[
S_{WCSP} = \langle [0, +\infty], \text{min}, +, +, +, \infty, 0 \rangle
\]

\(ub = x\) preferences from constraints on assigned variables

**Iron quality**
- bad \(\rightarrow 10\) €
- high \(\rightarrow 20\) €

**Wood quality**
- bad \(\rightarrow 10\) €
- high \(\rightarrow 20\) €
- medium \(\rightarrow 30\) €
- high \(\rightarrow 50\) €

**Processing time**
- 2 days \(\rightarrow 0\) €
- 3 days \(\rightarrow 0\) €
- (b, b) \(\rightarrow 0\) €
- (h, m) \(\rightarrow 30\) €
- (h, h) \(\rightarrow 0\) €
- (b, 2) \(\rightarrow 40\) €
- (m, 2) \(\rightarrow 50\) €
- (m, 3) \(\rightarrow 70\) €
- (h, 3) \(\rightarrow 70\) €

\(lb = 100\)
\(ub = +\infty\)
Inference: Constraint propagation

- Constraint propagation (ex. arc-consistency):
  - Deletes an element $a$ from the domain of a variable $x$ if, according to a constraint between $x$ and $y$, it does not have any compatible element $b$ in the domain of $y$
  - Iterate until stability

- Polynomial time

- Very useful at each node of the search tree to prune subtrees
No matter what the other constraints are, $X=b$ cannot participate in any solution. So we can delete it without changing the set of solutions.
Properties

- Equivalence: each step preserves the set of solutions
- Termination (with finite domains)
- Order-independence
Fundamental operations with soft constraints

- **Projection**: eliminate one or more variables from a constraint obtaining a new constraint preserving all the information on the remaining variables
  Formally: If $c=<f, \text{con}>$, then $c|_I = <f', I \cap \text{con}>$
  - $f'(t') = + (f(t))$ over tuples of values $t$ s.t. $t|_I \cap \text{con} = t'$

- **Combination**: combine two or more soft constraints obtaining a new soft constraint “synthesizing” all the information of the original ones
  Formally: If $c_i=<f_i, \text{con}_i>$, then $c_1 \times c_2 = <f, \text{con}_1 \cup \text{con}_2>$
  - $f(t) = f_1(t|_{\text{con}_1}) \times f_2(t|_{\text{con}_2})$
Projection: fuzzy example

If $c = <f, \text{con}>$, then $c|_l = <f', l \cap \text{con}>$

$f'(t') = + (f(t))$ over tuples of values $t$ s.t. $t|_l \cap \text{con} = t'$

c = $<f, \{\text{mc, w}\}>$

$S_{FCSP} = <[0,1], \max, \min, 0, 1>$

Main Course

{Fish, Meat}

Wine

{White, Red}

Main Course

Fish $\rightarrow$ max($f$(fish, white), $f$(fish, red)) $= \max(1, 0.8) = 1$

Meat $\rightarrow$ max($f$(meat, white), $f$(meat, red)) $= \max(0.3, 0.7) = 0.7$

IFAAMAS summer school -- Beijing, August 2013
Projection: weighted example

If \( c = \langle f, \text{con} \rangle \), then \( c|_I = \langle f', I \cap \text{con} \rangle \)

\[ f'(t') = + (f(t)) \text{ over tuples of values } t \text{ s.t. } t|_I \cap \text{con} = t' \]

\( c = \langle f, \{\text{wq, pt}\} \rangle \)

<table>
<thead>
<tr>
<th>Wood quality</th>
<th>Processing time</th>
</tr>
</thead>
<tbody>
<tr>
<td>{bad, med, high}</td>
<td>{2, 3}</td>
</tr>
</tbody>
</table>

- \( (b, 2) \rightarrow 40 \text{ €} \)
- \( (m, 2) \rightarrow 50 \text{ €} \)
- \( (m, 3) \rightarrow 70 \text{ €} \)
- \( (h, 3) \rightarrow 70 \text{ €} \)

\( S_{WCSP} = \langle [0, +\infty], \text{min, +, +, } +\infty, 0 \rangle \)

\( c|_{\text{wq}} \)

- bad \( \rightarrow \min(f(b,2), f(b,3)) = \min(40, +\infty) = 40 \)
- med \( \rightarrow \min(f(m,2), f(m,3)) = \min(50, 70) = 50 \)
- high \( \rightarrow \min(f(h,2), f(h,3)) = \min(+\infty, 70) = 70 \)

IFAAMAS summer school -- Beijing, August 2013
Combination: fuzzy example

If ci=<fi,coni>, then: c1 x c2 = <f, con1 ∪ con2>

\[ f(t) = f1(t|_{con1}) \times f2(t|_{con2}) \]

\[ S_{FCSP}=<[0,1],\text{max},\text{min},0,1> \]

{slow, fast} → {256, 512, 1024} → P

VGA ⊗ MB

\(<s,256> \rightarrow 0.6 \>
\(<s,512> \rightarrow 0.7 \>
\(<s,1024> \rightarrow 0.9 \>
\(<f,256> \rightarrow 0.1 \>
\(<f,512> \rightarrow 0.9 \>
\(<f,1024> \rightarrow 1 \>

\{P4,AMD\} → <256,P4> \rightarrow 0.5 \>
\(<512,P4> \rightarrow 0.7 \>
\(<1024,P4> \rightarrow 0.9 \>
\(<256,AMD> \rightarrow 0.5 \>
\(<512,AMD> \rightarrow 0.5 \>
\(<1024,AMD> \rightarrow 0.5 \>

\[ f(s,256,P4) = \text{min}(0.6,0.5) = 0.5 \]
\[ f(f,1024,P4)=\text{min}(0.9,0.9)=0.9 \]

IFAAMAS summer school -- Beijing, August 2013
Combination: weighted example

If \( c_i = <f, con_i> \), then: \( c_1 \times c_2 = <f, con_1 \cup con_2> \)

\[
\begin{align*}
\text{f(t)} &= f_1(t|_{con_1}) \times f_2(t|_{con_2}) \\
\end{align*}
\]

**Example 1:**
- Iron quality: \{bad, high\}
- Wood quality: \{bad, med, high\}
- Processing time:
  - (b, b) \( \rightarrow \) 0 €
  - (h, m) \( \rightarrow \) 30 €
  - (h, h) \( \rightarrow \) 0 €

**Example 2:**
- Processing time:
  - 2 days \( \rightarrow \) 20 €
  - 3 days \( \rightarrow \) 30 €

**Example 3:**
- Processing time:
  - f(b, b, 2) = 0 + 20 = 20
  - f(h, m, 3) = 30 + 30 = 60
  - ...
Soft constraint propagation

- Deleting a value means passing from 1 to 0 in the semiring $\langle \{0,1\}, \text{or}, \text{and}, 0, 1 \rangle$
- In general, constraint propagation can change preferences to lower values in the ordering
- **Soft arc-consistency**: given $c_x$, $c_{xy}$, and $c_y$, compute $c_x := (c_x \times c_{xy} \times c_y) |_x$
- Iterate until stability
Example: fuzzy arc-consistency

\[ c_x := (c_x \times c_{xy} \times c_y)_x \]

Fuzzy semiring = \([0,1], \text{max}, \min, 0, 1\) 
\[ \rightarrow + = \text{max an } x = \text{min} \]

VGA = s \rightarrow \max(\min(0.2, 0.6, 0.5), \min(0.2, 0.7, 0.8), \min(0.2, 0.9, 0.7)) = \max(0.2, 0.2, 0.2) = 0.2

VGA = f \rightarrow \max(\min(0.9, 0.1, 0.5), \min(0.9, 0.9, 0.8), \min(0.9, 1, 0.7)) = \max(0.1, 0.8, 0.7) = 0.8

IFAAMAS summer school -- Beijing, August 2013
weighted arc-consistency?

Weighted semiring $= \langle [0, +\infty], \min, +, +\infty, 0 \rangle$

Not equivalent!

$\text{pref}_1(a,a) = 10$
$\text{pref}_2(a,a) = 20$
Properties

- If \( x \) idempotent (ex.: fuzzy, classical):
  - Equivalence
  - Termination
  - Order-independence

- If \( x \) not idempotent (ex.: weighted CSPs, prob.), we could count more than once the same constraint \( \Rightarrow \) we need to compensate by subtracting appropriate quantities somewhere else \( \Rightarrow \) we need an additional property (fairness=presence of -)
  - Equivalence
  - Termination
  - Not order-independence

[Schiiex, CP 2000]
References for preferences and soft constraints

- Handbook of constraint programming, Rossi, Van Beek, Walsh eds., Elsevier, 2006
- Possibilistic Constraint Satisfaction Problems or "How to Handle Soft Constraints?". T. Schiex, UAI 1992
MODELLING PREFERENCES COMPACTLY: CP NETS
Qualitative and conditional preferences

- Soft constraints model quantitatively unconditional preferences
- Many problems need statements like
  - “I like white wine if there is fish” (conditional)
  - “I like white wine better than red wine” (qualitative)
- Quantitative ➔ a level of preference for each assignment of the variables in a soft constraint ➔ possibly difficult to elicitate preferences from user
Preference statements in CP nets

- Conditional preference statements
  - "If it is fish, I prefer white wine to red wine"
  - syntax:
    fish: white wine > red wine

- Ceteris paribus interpretation
  - all else being equal
  - \{fish, white wine, ice cream\} > (preferred to) \{fish, red wine, ice cream\}
  - \{fish, white wine, ice cream\} ? \{fish, red wine, fruit\}

[Boutelier, Brafman, Domshlak, Hoos, Poole. JAIR 2004]
[Domshlak, Brafman KR02]
CP nets

- Variables \( \{X_1, \ldots, X_n\} \) with domains
- For each variable, a total order over its values
- Independent variable:
  - \( X = v_1 > X = v_2 > \ldots > X = v_k \)
- Conditioned variable: a total order for each combination of values of some other variables (conditional preference table)
  - \( Y = a, Z = b: X = v_1 > X = v_2 > \ldots > X = v_k \)
  - \( X \) depends on \( Y \) and \( Z \) (parents of \( X \))
- Graphically: directed graph over \( X_1, \ldots, X_n \)
  - Possibly cyclic
CP nets: an example

Independent feature

Main course

fish > meat

Conditional Preference Table

<table>
<thead>
<tr>
<th>Main course</th>
<th>Wine</th>
</tr>
</thead>
<tbody>
<tr>
<td>fish</td>
<td>white &gt; red</td>
</tr>
<tr>
<td>meat</td>
<td>red &gt; white</td>
</tr>
</tbody>
</table>

Dependent feature

Wine

peaches > strawberries

Independent feature

Fruit

IFAAMAS summer school -- Beijing, August 2013
**CP-net semantics**

- **Worsening flip**: changing the value of an attribute in a way that is less preferred in some statement. Example:

  (fish, white wine, peaches)

  ![worsening flip]

  (fish, red wine, peaches)

- An outcome $O_1$ is preferred to $O_2$ iff there is a sequence of worsening flips from $O_1$ to $O_2$

- Optimal outcome: if no other outcome is preferred
Preorder over solutions

- A CP net induces an ordering over the solutions (directly)
- In general, a preorder
- Some solutions can be in a cycle: for each of them, there is another one which is better
- Acyclic CP net: one optimal solution
- Not all orderings can be obtained with CP nets
  - Outcomes which are one flip apart must be ordered
Solution ordering

Solution ordering diagram showing the optimal solution with the order of main courses, wine, and fruit.

- Fish > Meat
- Peaches > Strawberries

**Main Course**
- Fish: White > Red
- Meat: Red > White

**Wine**
- Fish: White, Peaches
- Fish: Red, Peaches
- Fish: White, Berries
- Fish: Red, Berries
- Meat: White, Peaches
- Meat: White, Berries

**Fruit**
- Peaches > Strawberries

Optimal solution: Fish, white, peaches
Interesting questions in CP nets

- Find an optimal outcome
  - In general, difficult (as solving a CSP)
  - Easy for acyclic networks
    - always have exactly one optimal solution
    - sweep forward in linear time

- Find the next solution in a linearization of the solution ordering
  - Easy for acyclic CP-nets

- Does O1 dominate O2?
  - Difficult even for acyclic CP nets

- Is O optimal?
  - Easy: test O against a CSP
How to find optimal solutions in CP nets

- Acyclic CP-nets: sweep forward algorithm
  - Follow the dependency graph
  - For each variable, assign the most preferred value in the context of the parents’ assignment
Sweep forward algorithm

Main course

- Fish > Meat

Wine

- White > Red
- Red > White

Fruit

- Peaches > Strawberries

1. F = Peaches
2. M = Fish
3. Since M = Fish, W = White

Optimal solution

- Fish, white, peaches
- Fish, white, berries
- Fish, red, peaches
- Fish, red, berries
- Meat, red, peaches
- Meat, white, peaches
- Meat, red, berries
- Meat, white, berries

IFAAMAS summer school -- Beijing, August 2013
Cyclic CP nets

- Given a (cyclic) CP net, we can generate in polynomial time a set of constraints \( P \) such that the solutions of \( P \) coincides with the set of optimal solutions of the CP net.

- For each \( Y=a, Z=b: X=v_1 > X=v_2 > ... > X=v_k \), we build the constraint \( Y=a, Z=b \Rightarrow X=v_1 \)
Optimal solutions in cyclic CP nets

Main course

Fish: white > red
Meat: red > white

White: fish > meat
Red: meat > fish

Wine

peaches > strawberries

Fruit

Constraints:
F = peaches
M = fish → W = white
M = meat → W = red
W = white → M = fish
W = red → M = meat

Optimal solutions

Fish: white, peaches
Meat: red, peaches

Fish: white, berries
Meat: red, berries

Fish: red, peaches
Meat: red, berries

Fish: red, berries
Meat: white, berries

IFAAMAS summer school -- Beijing, August 2013
The next best solution

- Also important: given a solution $s$, find the next one
  - Top $k$ solutions in web search
  - Next most preferred option in stable marriage proposal-based algorithms

- Next where? In a linearization of the preference ordering
  - Compatible with the preference ordering
  - Has to linearize incomparability

Next($P, s, l$)

- $P$: Preference representation
- $s$: Solution
- $l$: Linearization of the preference ordering

The solution following $s$ according to the preferences in $P$ and linearization $l$

Next on a CP-net: example

IFAAMAS summer school -- Beijing, August 2013
Next on acyclic CP-nets is easy for conditional lex linearization

- Acyclic CP-nets generate a partial order with one top element
- Assume Boolean vars (for simplicity)
- Main idea: Boolean vector for each solution
  - Position $i$ for variable $x_i$: 0 if $x_i$ has its most preferred value given its parents, otherwise 1
- Lex order over the vectors is a linearization
- Next is just Boolean vector incrementation
  - Given $s$, compute its vector $v$
  - Increment the vector obtaining $v'$
  - Given $v'$, obtain the corresponding solution $s'$

Solution Ordering

Loc-A > Loc-B

WHERE

Loc-A: Analyze > Image
Loc-B: Image > Analyze

WHAT

St2 > St1

DLINK

Expressive power

If interested in the optimal solutions:

Semiring-based

Valued

weighted \( R \) \( \leftrightarrow \) Prob

weighted \( N \)

Fuzzy \( \rightarrow \) Lexicographic \( \rightarrow \) Classical

CP nets

IFAAMAS summer school -- Beijing, August 2013
Given a CP net, it is always possible to build in polynomial time a classical CSP with the same set of optimal solutions

For each $Y=a$, $Z=b$: $X=v_1 > X=v_2 > \ldots > X=v_k$, we build the constraint $Y=a$, $Z=b \rightarrow X=v_1$

For some CSP, it is not possible to build a CP net with the same set of optimals

Ex.: two (optimal) solutions $<X=a,Y=b,Z=c>$ and $<X=a,Y=b,Z=d>$ $\rightarrow$ they must be ordered in a CP net

IFAAMAS summer school -- Beijing, August 2013
Expressive power

If interested in maintaining the solution ordering:

CP nets

Semiring-based

Valued

weighted$_R$

weighted$_N$

Prob

Fuzzy

Lexicographic

Classical

IFAAMAS summer school -- Beijing, August 2013
CP nets vs. Soft Constraints
(solution ordering)

- There are CP nets whose ordering cannot be modelled (in poly time) by a soft CSP
  - Otherwise dominance testing would be easy in CP-nets

- There are soft CSPs whose orderings cannot be modelled by a CP net
  - Not all orderings can be represented by CP nets
## Soft constraints vs. CP-nets

<table>
<thead>
<tr>
<th>Preference orderings</th>
<th>Soft CSPs</th>
<th>Tree-like soft CSPS</th>
<th>CP-nets</th>
<th>Acyclic CP-nets</th>
</tr>
</thead>
<tbody>
<tr>
<td>all</td>
<td>all</td>
<td>some</td>
<td>some</td>
<td></td>
</tr>
<tr>
<td>difficult</td>
<td>easy</td>
<td>difficult</td>
<td>easy</td>
<td></td>
</tr>
<tr>
<td>easy</td>
<td>easy</td>
<td>difficult</td>
<td>difficult</td>
<td></td>
</tr>
<tr>
<td>difficult</td>
<td>difficult for weighted, easy for fuzzy</td>
<td>difficult</td>
<td></td>
<td></td>
</tr>
<tr>
<td>difficult</td>
<td>easy</td>
<td>easy</td>
<td>easy</td>
<td></td>
</tr>
</tbody>
</table>

**IFAAMAS summer school -- Beijing, August 2013**
## Soft constraints vs. CP-nets

<table>
<thead>
<tr>
<th>Preference orderings</th>
<th>Soft constraints</th>
<th>CP nets (acyclic)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>all</td>
<td>some</td>
</tr>
<tr>
<td></td>
<td>difficult</td>
<td>easy</td>
</tr>
<tr>
<td></td>
<td>easy</td>
<td>difficult</td>
</tr>
<tr>
<td></td>
<td>difficult</td>
<td>easy</td>
</tr>
<tr>
<td></td>
<td>difficult</td>
<td>easy</td>
</tr>
</tbody>
</table>

IFAAMAS summer school -- Beijing, August 2013
Approximating CP nets via Soft Constraints

- We can **approximate** the ordering of a CP net via a soft constraint problem
  - Weighted or fuzzy soft constraints
  - For ordered outcomes, same ordering
  - For incomparable outcomes, tie or order \(\Rightarrow\) more ordered
  - Easy dominance test

CP statements \(\Rightarrow\) Soft constraints \(\Rightarrow\) Soft constraint solver

Hard constraints \(\Rightarrow\) approx. Soft constraints

Soft constraints

IFAAMAS summer school -- Beijing, August 2013

[Domshlak, Rossi, Venable, Walsh, IJCAI 2003]
Constrained CP-net

A **Constrained CP-net** on variables $X = \{X_1, \ldots, X_n\}$ is a pair $<N,C>$ where:
- $N$ is a CP-net on variables $X$
- $C$ is a set of Hard or Soft Constraints on $X$

**Constrained CP-net semantics:**

$O_1 \geq O_2$ iff
- $\text{Pref}(O_1) > \text{pref}(O_2)$ in $C$, or
- $\text{Pref}(O_1) = \text{pref}(O_2)$ in $C$ and there is a *chain of worsening flips* from $O_1$ to $O_2$ through outcomes with equal or higher preference
- $O$ optimal if feasible and undominated in the CP net (not necessarily optimal in the CP net)
## Softly Constrained CP net: example

### CP net

<table>
<thead>
<tr>
<th>Main course</th>
<th>Wine</th>
</tr>
</thead>
<tbody>
<tr>
<td>fish</td>
<td>white &gt; red</td>
</tr>
<tr>
<td>meat</td>
<td>red &gt; white</td>
</tr>
<tr>
<td>peaches</td>
<td>strawberries</td>
</tr>
</tbody>
</table>

### Main course

- Fish, red, peaches
- Fish, white, peaches
- Fish, red, berries
- Fish, white, berries
- Meat, red, peaches
- Meat, white, peaches
- Meat, red, berries
- Meat, white, berries

### Soft Constraint

- White → 0.2
- Red → 1

### Optimal

- Fish, white, peaches
- Fish, white, berries
- Meat, white, peaches
- Meat, white, berries
How to obtain an optimal outcome of a constrained CP net $<N,C>$

- From $N$ to optimality constraints $OC$
- If $\text{Sol}(OC \cup C)$ is not empty, then they are (some of the) optimal outcomes $\Rightarrow$ take one of them
  $\Rightarrow$ only hard constraint solving
- Otherwise, dominance testing between feasible outcomes (more costly)

[ Prestwich, Rossi, Venable, Walsh, AAAI 2005]

IFAMMAS summer school -- Beijing, August 2013
(Conditional + qualitative + quantitative) preferences + constraints

CP net
Hard constraints
Soft constraints

Soft constraint Solver (+ dominance test in CP net)

Optimal Solutions

IFAAMAS summer school -- Beijing, August 2013
References for CP-nets

- Extended semantics and optimization algorithms for CP-networks, R. Brafman and Y. Dimopoulos, Computational Intelligence, 20(2), 2004
- Constraint-based Preferential Optimization, S. Prestwich, F. Rossi, K. B. Venable, T. Walsh, AAAI 2005
VOTING WITH COMBINATORIAL DOMAINS
Multiple issues

- Until now we have considered voting over one issue only
- Now we consider several issues
- Example:
  - 3 referendum (yes/no)
  - Each voter has to give his preferences over triples of yes and no
  - Such as: YYY>NNN>YNY>YNN>etc.
- With k issues, k-tuples ($2^k$ if binary issues)
Paradox of multiple elections

- 13 voters are asked to each vote yes or no on 3 issues:
  - 3 voters each vote for YNN, NYN, NNY
  - 1 voter votes for YYY, YYN, YNY, NYY
  - No voter votes for NNN

- Majority on each issue: the winner is NNN!
  - Each issue has 7 out of 13 votes for no
What is a paradox?

- Given
  - A voting rule
  - A profile of ballots
  - A property applicable to both profiles and outcomes
- Each ballot satisfies the property, but the outcome does not
- Example: no ballot is for NNN, but NNN is the outcome of the election
- (applies also to Condorcet paradox)

- What can we do then?
Plurality on combinations

- Ask each voter for her most preferred combination and apply plurality
  - Avoids the paradox, computationally light
  - Almost random decisions
  - Example: 10 binary issues, 20 voters $\Rightarrow 2^{10} = 1024$ combinations to vote for but only 20 voters, so very high probability that no combination receives more than one vote
  - tie-breaking rule decides everything

- Similar also for voting rules that use only a small part of the voters’ preferences (ex.: k-approval with small k)
Other rules on combinations

- Vote on combinations and use other voting rules that use the whole preference ordering on combinations
- Avoids the arbitrariness problem of plurality
- Not feasible when there are large domains
- Example:
  - Borda (needs the whole preference ordering)
  - 6 binary issues \(2^6 = 64\) possible combinations
  - each voter has to choose amongst 64! possible ballots
Sequential voting

- Vote separately on each issue, but do so sequentially.
- This gives voters the opportunity to make their vote for one issue depend on the decisions on previous issues.
Condorcet losers

- Condorcet loser (CL): candidate that loses against any other candidate in a pairwise contest
- Electing a CL is very bad, but Plurality sometimes elects it
- Example:
  - 2 votes for $X > Y > Z$
  - 2 votes for $Y > X > Z$
  - 3 votes for $Z > X > Y$
  - $Z$ is the Plurality winner and the Condorcet loser
Sequential voting and Condorcet losers

- Sequential voting avoids the problem of electing Condorcet losers

- Thm.: Sequential plurality voting over binary issues never elects a Condorcet loser
  - Proof: Consider the election for the final issue. The winning combination cannot be a CL, since it wins at least against the other combination that was still possible after the penultimate election
  - [Lacy, Niou, J. of Theoretical Politics, 2000]

- But no guarantee that sequential voting elects the Condorcet winner (Condorcet consistency).
SEQUENTIAL VOTING WITH SOFT CONSTRAINTS
Profiles via soft constraints

- Agents expressing preferences via soft constraints
- Over a common set of decisions/options
  - options = complete variable assignments
- Same vars and var domains for all agents, different soft constraints
- Profile = preferences of all agents
  - Explicit profile: preference orderings are given
  - Implicit profile: compact representation of the preferences
- We will select a decision using a voting rule
  - Decision = solution for the agents soft constraint satisfaction problems (sof CSP)
  - Voting rule: function from an explicit profile to a decision
- In the dinner example:
  - Each friend has his own soft CSP to express the preferences over the dinners
  - We need to select one dinner over the 625 possible ones
Dinner example, three agents

Agent 1

Agent 2

Agent 3

IFAAMAS summer school -- Beijing, August 2013
How to select a decision?

- **One step approach:**
  - Given the implicit profile, compute the explicit profile and apply a voting rule

- **Problems:**
  - The explicit profile needs exponential space
  - Computing the explicit profile may be very expensive in time
    - Both optimal and next solution are difficult to compute in general for soft constraints

- **Sequential approach**
  - For each variable
    - compute an explicit profile over the variable domain
    - apply a voting rule to this explicit profile
    - add the information about the selected variable value

- **Similar approach used for CP-nets in** [Lang, Xia, 2009]
Dinner example using plurality

Plurality

Pasta = Pesto

Plurality

Drink = Beer

Winner
Local vs. sequential properties

- If each $r_i$ has the property, does the sequential rule have the property?
- If some $r_i$ does not have the property, does the sequential rule not have it?
  - If the sequential rule has a property, do all the $r_i$ have it?
# Properties

<table>
<thead>
<tr>
<th></th>
<th>Local to sequential</th>
<th>Sequential to local</th>
</tr>
</thead>
<tbody>
<tr>
<td>Condorcet consistency</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Anonymity</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Neutrality</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Consistency</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Participation</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Efficiency</td>
<td>yes if single most preferred option for all agents</td>
<td>yes</td>
</tr>
<tr>
<td>Monotonicity</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>IIA</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Non-dictatorship</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Strategy-proofness</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>

IFAAMAS summer school -- Beijing, August 2013

[Dalla Pozza, Pini, Rossi, Venable, IJCAI 2011]
Complexity of coalitional constructive manipulation

- Constructive Coalitional Manipulation $CC(d,C,P,r)$
  - Given voting rule $r$, how difficult it is for coalition of voters $C$ to make candidate $d$ win, knowing the other agents’ preferences $P$?
    - Easy for Copeland with 3 candidates and for Plurality [Conitzer et al., 2007]
    - Difficult for Copeland [Faliszewski et al., 2008]

- Thms:
  - Easy for all local rules $\rightarrow$ Easy for sequential (if soft constraints are tractable)
  - Hard for one local rule $\rightarrow$ Hard for the sequential procedure

[Conitzer et al., 2007]
[Faliszewski et al., 2008]
[Dalla Pozza, Pini, Rossi, Venable, 2011]
Experimental setting

- Randomly generated tree-shaped soft implicit profiles
  - \( n \): number of variables
  - \( m \): number of agents
  - \( d \): domain size
  - \( t \): tightness

- Same rule \( r \) for all steps

- Comparison between two voting rules
  - \( \text{seq}(r) \), from the implicit profile to a solution
  - \( r \), from the explicit profile to a solution
  - baseline

- We measure the quality of returned solution \( s \)
  - for each agent, distance between preference of \( s \) and of its most preferred solutions, averaged over all agents
- Sequential rule much faster (no need to compute the explicit profile)
- Result of about the same quality
- Price to pay to search an agreement with others
The sequential approach behaves like the non-sequential one

- independently of the variable ordering
- independently of the amount of consensus among agents
- also on best and worst cases
SEQUENTIAL VOTING WITH CP-NETS
Profiles via compatible CP-nets

- n voters, voting by giving a CP-net each
  - Same variables, different dependency graph and CP tables
- Compatible CP-nets: there exists a linear order on the variables that is compatible with the dependency graph of all CP-nets (that is, it completes the DAG)
- Then vote sequentially in this order
- Thm.: Under these assumptions, sequential voting is Condorcet consistent if all local voting rules are
  - (Lang and Xia, Math. Social Sciences, 2009)
Example

3 Rovers must decide:
• Where to go: Location A or Location B
• What to do: Analyze a rock or Take a picture
• Which station to downlink the data to: Station 1 or Station 2

![Diagram of decision-making process]
STABLE MARRIAGE PROBLEMS
Preferences over agents

- Until now, agents expressed preferences over alternative decisions (different from the agents).
- Goal: to choose one of the decisions based on the agents’ preferences.
- Now, we consider agents expressing preferences over other agents.
  - Bipartite set of agents.
- Goal: to choose a matching among the agents based on their preferences.
  - Matching: set of pairs \((A_1, A_2)\), where \(A_1\) comes from the first set and \(A_2\) from the second one.
Looking for a job

- **Assume**
  - As many positions as the number of people looking for them
  - Each person sends his cv to all companies

- **Preferences**
  - Each person will rank all the openings
  - Each company will rank all the students

- **How to do the matching in such a way that “everybody is happy”?**

- **Notice**
  - Bipartite set of agents
  - Preferences over other agents, not over alternatives
Other practical scenarios

- Assigning projects
- Job hunting
- Matching students with schools
- Matching doctors with hospitals
- Matching sailors to ships
- Matching producers to consumers
- Choosing roommates
- …
Stable marriage formulation

- Two sets of agents: men and women
- Idealized model
  - Same number of men and women
  - All men totally order all women, and vice-versa
Stable marriage

- Given preferences of n men
  - Greg: Amy > Bertha > Clare
  - Harry: Bertha > Amy > Clare
  - Ian: Amy > Bertha > Clare

- Given preferences of n women
  - Amy: Harry > Greg > Ian
  - Bertha: Greg > Harry > Ian
  - Clare: Greg > Harry > Ian

- Find a *stable marriage*
Stable marriage

- Assignment of men to women (or equivalently of women to men)
  - Idealization: everyone marries at the same time
- No pair (man, woman) not married to each other would prefer to run off together
  - Blocking pair: pair (m, w) such that the marriage contains (m, w') and (m', w), but m prefers w to w', and w prefers m to m'
An example of an unstable marriage

- Greg: Amy > Bertha > Clare
- Harry: Bertha > Amy > Clare
- Ian: Amy > Bertha > Clare

- Amy: Harry > Greg > Ian
- Bertha: Greg > Harry > Ian
- Clare: Greg > Harry > Ian

Bertha & Greg would prefer to be together
An example of a stable marriage

- Greg: Amy > Bertha > Clare
- Harry: Bertha > Amy > Clare
- Ian: Amy > Bertha > Clare

- Amy: Harry > Greg > Ian
- Bertha: Greg > Harry > Ian
- Clare: Greg > Harry > Ian

*Men do ok, women less well*
Another stable marriage

- Greg: Amy > Bertha > Clare
- Harry: Bertha > Amy > Clare
- Ian: Amy > Bertha > Clare

- Amy: Harry > Greg > Ian
- Bertha: Greg > Harry > Ian
- Clare: Greg > Harry > Ian

Women do ok, men less well
Many stable marriages

- Given any stable marriage problem
  - There is at least one stable marriage!
  - There may be many stable marriages
  - They form a lattice, ordered according to men’s (or women’s) preferences
    - The higher in the lattice, the more men are happy: SM1 > SM2 if in SM1 all men are at least as happy as in SM2
    - At least as happy: married to the same or a more preferred woman
Gale Shapley algorithm

- Initialize every person to be free
- While exists a free man
  - Find best woman he hasn’t proposed to yet
  - If this woman is free, declare them engaged
    - Else, if this woman prefers this proposal to her current partner, then declare them engaged (and “free” her current partner)
    - Else, this woman prefers her current partner and she rejects the proposal
Gale Shapley algorithm

- Initialize every person to be free
- While exists a free man
  - Find best woman he hasn’t proposed to yet
  - If this woman is free, declare them engaged
    - Else if this woman prefers this proposal to her current partner then declare them engaged (and “free” her current partner)
    - Else this woman prefers her current partner and she rejects the proposal

<table>
<thead>
<tr>
<th>Greg: Amy &gt; Bertha &gt; Clare</th>
</tr>
</thead>
<tbody>
<tr>
<td>Harry: Bertha &gt; Amy &gt; Clare</td>
</tr>
<tr>
<td>Ian: Amy &gt; Bertha &gt; Clare</td>
</tr>
<tr>
<td>Amy: Harry &gt; Greg &gt; Ian</td>
</tr>
<tr>
<td>Bertha: Greg &gt; Harry &gt; Ian</td>
</tr>
<tr>
<td>Clare: Greg &gt; Harry &gt; Ian</td>
</tr>
</tbody>
</table>
Gale Shapley algorithm

- Greg proposes to Amy, who accepts \( \rightarrow (G, A) \)
- Harry proposes to Bertha, who accepts \( \rightarrow (H, B) \)
- Ian proposes to Amy
- Amy is with Greg, and she prefers Greg to Ian, so she refuses
- Ian proposes to Bertha
- Bertha is with Harry, and she prefers Harry to Ian, so she refuses
- Ian proposes to Claire, who accepts \( \rightarrow (I, C) \)

Greg: Amy>Bertha>Clare
Harry: Bertha>Amy>Clare
Ian: Amy>Bertha>Clare
Amy: Harry>Greg>Ian
Bertha: Greg>Harry>Ian
Clare: Greg>Harry>Ian
Gale Shapley algorithm terminates with everyone married

- Suppose some man is not married at the end
- Then some woman is also unmarried
- But once a woman is married, she only “trades” up
- Hence this woman was never proposed to
  - But if a man is unmarried, he has proposed to and been rejected by every woman
- This is a contradiction as he has never proposed to the unmarried woman!
Gale Shapley algorithm terminates with a stable marriage

- Suppose there is a blocking pair m-w not married
  - Marriage contains (m,w’) and (m’,w)
  - m prefers w to w’, and w prefers m to m’
- Case 1. m never proposed to w
  - Not possible because men move down with the proposals, and w’ is less preferred than w
- Case 2. m had proposed to w
  - But w rejected m, or left him later
  - However, women only ever trade up
  - Hence w prefers m’ to m
  - So the current pairing is stable!
Other features of Gale Shapley algorithm

Each of $n$ men can make at most $(n-1)$ proposals

- Hence GS runs in $O(n^2)$ time

There may be more than one stable marriage

- GS finds **man optimal** solution: there is no stable matching in which any man does better
- GS finds **woman pessimal** solution: in all stable marriages, every woman does at least as well or better
Gale Shapley finds the male optimal solution

- S1: marriage found by GS
- In S1, consider first step where a man is rejected by his best feasible woman
- Man M has proposed and been rejected by his best feasible woman W, since W prefers her current partner Z
  - Note: W prefers Z to M
  - Note: There exists another stable marriage S2 with man M married to woman W (and man Z to woman W’)
- Man Z has not yet been rejected by his best possible woman
  - Z must prefer W at least as much as his best possible woman
- S2 contains (M,W) (Z,W’) and is not a stable marriage as Z and W would prefer to be together
  - Z prefers W to W’
  - W prefer Z to M
Gale Shapley finds the woman pessimal solution

- Consider stable marriage S1 returned by GS
- Let (M,W) be married in S1 but M is not the worst possible man for woman W
  - There exists another stable marriage S2 with (M’,W) (M,W’) and M’ worse than M for W
  - By male optimality of S1, M prefers W to W’
  - Also, W prefers M to M’
  - Then (M,W) is a blocking pair for S2
Other stable marriages

- GS finds male-optimal (or female-optimal) marriage
- A set of agents is favored over the other one
- Other algorithms find “fairer” marriages
- Ex.: stable marriage which minimizes the maximum regret [Gusfield 1989]
  - regret of a man/woman = distance between his partner in the marriage and his most preferred woman/man
Extensions: ties

- Cannot always make up our minds
- Preference ordering: total order with ties
- Two notions of stability:
  - Weak stability: no pair m-w not married where m strictly prefers w to his partner, and w strictly prefers m to her partner
  - Strong stability: no pair m-w not married where m strictly prefers w to his partner, and w prefers m at least as much as her partner
Existence of stable marriage with ties

- Strongly stable marriage may not exist
  - $O(n^4)$ algorithm for deciding existence
- Weakly stable marriage always exists
  - Just break ties arbitrarily
  - Run GS
  - Resulting marriage is weakly stable
Extensions: incomplete preferences

- There are some people we may be unwilling to marry
- \((m,w)\) blocking pair iff
  - \(m\) and \(w\) do not find each other unacceptable
  - \(m\) is unmarried or prefers \(w\) to current partner
  - \(w\) is unmarried or prefers \(m\) to current partner
Solving stable marriage problems with incomplete preferences

- Just apply GS algorithm
  - Extends easily

- Men and woman partition into two sets
  - Those who have partners in all stable marriages
  - Those who do not have partners in any stable marriage
  - In all stable marriages, the same people are married

→ Stable marriages have all the same number of pairs
Extensions: ties + incomplete prefs

- Weakly stable marriages may have different sizes
  - Unlike with just ties, where they are all complete
- Finding weakly stable marriage of max. cardinality is NP-hard
  - Even if only women declare ties
Strategy proofness

- GS is strategy proof for men
  - Assuming GS male optimal algorithm
  - No man can do better than the male optimal solution

- However, women can profit from lying
  - Assuming male optimal algorithm is run
  - And they know complete preference lists
Manipulation by women

- Greg: Amy > Bertha > Clare
- Harry: Bertha > Amy > Clare
- Ian: Amy > Bertha > Clare
- Amy: Harry > Greg > Ian
- Bertha: Greg > Harry > Ian
- Clare: Greg > Harry > Ian
- Amy lies
- Amy: Harry > Ian > Greg
- Bertha: Greg > Harry > Ian
- Clare: Greg > Harry > Ian
Manipulation by women

- Greg: Amy > Bertha > Clare
- Harry: Bertha > Amy > Clare
- Ian: Amy > Bertha > Clare

Greg proposes to Amy, who accepts
Harry proposes to Bertha, who accepts
Ian proposes to Amy, who accepts (Greg left alone)
Greg proposes to Bertha, who accepts (Harry left alone)
Harry proposes to Amy, who accepts (Ian left alone)
Ian proposes to Bertha, who rejects
Ian proposes to Claire, who accepts
Stable matching obtained: (Greg, Bertha), (Harry, Amy), (Ian, Claire)

Amy lies

- Amy: Harry > Ian > Greg
- Bertha: Greg > Harry > Ian
- Clare: Greg > Harry > Ian

IFAAMAS summer school -- Beijing, August 2013
Impossibility of strategy proofness

- GS can be manipulated
- Every stable marriage procedure can be manipulated if preference lists can be incomplete [Roth ’82]
Impossibility of strategy proofness

- Consider
  - Greg: Amy > Bertha
  - Harry: Bertha > Amy
  - Amy: Harry > Greg
  - Bertha: Greg > Harry

- Two stable marriages:
  - (Greg, Amy) (Harry, Bertha) or (Greg, Bertha) (Harry, Amy)

- Suppose we get the male optimal solution
  - (Greg, Amy) (Harry, Bertha)
  - If Amy lies and says Harry is her only acceptable partner
  - Then, with any sm procedure, we must get (Harry, Amy) (Greg, Bertha), as this is the only stable marriage

- Other cases can be manipulated in a similar way
Making manipulation hard

- For some sm procedure, finding the manipulation is easy
  - Example: GS algorithm
- For others, it is difficult
- Can we make the manipulation hard to find?
  - As with voting, this may be a barrier to mis-reporting of preferences

[Pini, Rossi, Venable, Walsh, AAMAS 09]
Gender swapping

- Basic idea
  - Men have no incentive to manipulate GS
  - But women do
- Construct SM procedure that may swap men with women
Gender swapping: non-deterministic solution

- Toss a coin
  - Heads: men stay men
  - Tails: men become women and vice versa
- No incentive to mis-report preferences
  - 50% chance that it will hurt
- Not everyone likes
  - Randomized procedures
  - Probabilistic guarantees
A deterministic solution

- Pick a set of stable marriages
- Choose between them based on agents’ preferences
  - Make this choice difficult to manipulate!
  - Choice based on voting
    - Complexity of manipulating voting rule => complexity of manipulating SM procedure
A deterministic solution: use STV

- Pick a set of stable marriages
- Choose between them based on agents’ preferences
  - Run a STV election to order men by women’s preferences (and women by men’s preferences)
  - For each SM, compute a male (female) score: vector where position j contains i if man (woman) j is married to the i-th most preferred woman (man)
  - Take lex largest between the two vectors
  - Pick SM with lex smallest vector
- Thm.: NP-hard to manipulate and gender neutral
  - NP-hardness inherited from hardness of manipulating STV
  - [Pini, Rossi, Venable, Walsh, AAMAS 2009]
- Also for other voting rules but not a general result
References for stable marriages

- Gusfield. Three fast algorithms for four problems in stable marriage. SIAM J. of Computing, 16(1), 1987
Conclusions

- Voting theory can be useful for preference aggregation in the context of AI
- Exploit axiomatic approach to choose the rule to use
- Adapt voting concepts to the AI context
Conclusions

- Computational complexity is an important issue in
  - Manipulation
  - Preference elicitation
- Complexity can be a friend
  - Ideally want it to be hard to find manipulation but easy to decide when to stop eliciting preferences!
- But NP-hardness is only worst case
A short book about these issues

“A short introduction to preferences: between Artificial Intelligence and Social Choice”
F. Rossi, K. B. Venable, T. Walsh
Morgan & Claypool
2011
http://www.morganclaypool.com/