Voting over combinatorial domains

FRANCESCA ROSSI

UNIVERSITÀ DEGLI STUDI DI PADOVA

RADCLIFFE INSTITUTE FOR ADVANCED STUDY HARVARD UNIVERSITY

Centralized decision making (rank aggregation)

- Several agents judge a set of items
- A collective decision has to be made
- From the individuals' rankings to a single ranking
- Respecting the judges' preferences as much as possible

Rank	Judge A	Judge B	Judge C	Judge D	Judge E	Aggregation sum	Aggregation average	Aggregation rank
item 1	5	6	2	9	1	23	4.6	4
item 2	7	3	6	5	7	28	5.6	8
item 3	5	3	2	3	3	16	3.2	2
item 4	2	9	6	9	3	29	5.8	9
item 5	7	3	9	1	7	27	5.4	6
item 6	1	9	6	5	6	27	5.4	6
item 7	2	8	1	5	7	23	4.6	4
item 8	2	6	9	4	10	31	6.2	10
item 9	10	1	2	5	1	19	3.8	3
item 10	7	1	2	1	3	14	2.8	1

- Examples:
  - Web search engines ranking web pages
  - Cameras ranking the plausible interpretations of an object
  - A hiring committee
  - Recommender systems
  - Friends deciding where to go for dinner
  - Political elections

- Environment: judges' preferences (or polls)
- Goal: collective preference ranking
- Actions: choose an aggregation method
- Uncertainty: over the rankings
- Centralized approach to achieve consensus or compromise

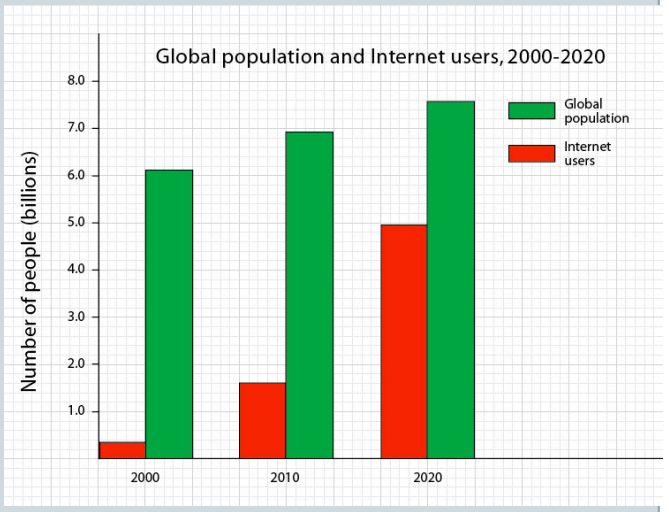
A simple example: friends choosing a dinner

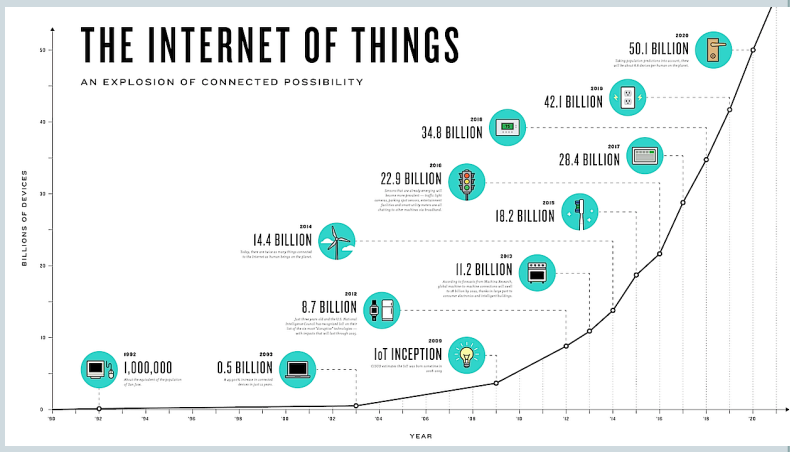
- They want to eat the same meal
  - Pasta, main dish, dessert, drink
- 5 options for each

  ➔ $5^4 = 625$ possible dinners
- Each friend has his own preferences over the possible dinners
- We need to choose one of them
- How do we model their preferences?
- How do we choose the single meal they will all eat?

These scenarios are more and more frequent

- People are connected and want to take decisions with their friends or peers
- Online social networks
- Influences among agents
- Manipulations
- Not just people, also "things"
  - 50 billion connected by 2020





More formal setting

- From
  - M candidates
  - N agents
  - For each agent, a preference order over the candidates
- we want to get
  - A single ordering of the candidates (social welfare function)
  - Or at least a "winning" candidate (social choice function)
- such that
  - the preferences of the agents are "considered"
- We can exploit concepts and techniques from voting theory (social choice)
  - After all, it looks like an election!

Voting rules (social choice functions)

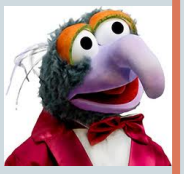
- Plurality: one vote for each agent, score is number of votes
- Borda: each agent give full ranking, score depend on the position on the ranking
- Approval: each agent approves some candidates, score is number of approvals
- K-approval: each agent approves k candidates
- Also many others
  - Kemeny, Single Transferrable Vote, Veto, Copeland, Minimax, Range, Schulze, Banks, Slater, Bucklin, Dogson, …

Plurality

- Vote: 1 candidate
- Result: candidate(s) with the most vote(s)
- Example:
  - 6 voters
  - Candidates:

Winner

Borda

rank	1 voter	rank	1 voter	rank	1 voter	rank	1 voters	rank	1 voter	Borda Count	Winner
3		3		3		3		3		9	
2		2		2		2		2		8	
1		1		1		1		1		7	
0		0		0		0		0		6	

Approval

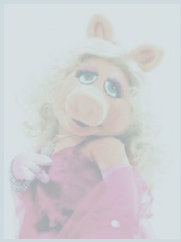








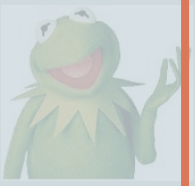










Winner

Scores


4


3


3


1

1 voter | 1 voter | 1 voter | 1 voters | 1 voter

2-approval

Which social choice function should we choose?

- With 2 candidates, easy: majority!
- With more than 2 candidates, look at their properties
- Examples of properties:
  - We don't want one friend to be always deciding the dinner, independently of what the other friends say (non-dictatorship)
  - If everybody says that dinner A is preferred to dinner B, we don't want B to be chosen (Pareto-efficiency)
  - Between A and B, the fact that A is chosen, or B, or none, should depend only on how the friends ordered A and B (independence of irrelevant alternatives)

Which social choice function should we choose?

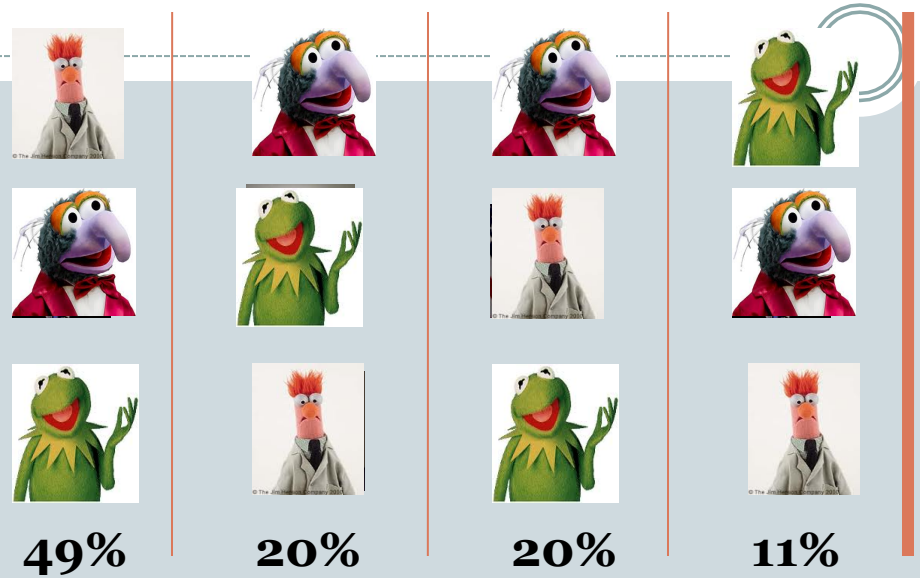
- With 2 candidates, easy: majority!
- With more than 2 candidates, look at their properties
- Examples:
  - We don't want one friend to be always deciding the dinner, independently of what the other friends say (non-dictatorship)
  - If everybody says that dinner A is preferred to dinner B, we don't want B to be chosen (Pareto-efficiency)
  - Between A and B, the fact that A is chosen, or B, or none, should depend only on how the friends ordered A and B (independence of irrelevant alternatives)
- Unfortunately …. No voting rule has these three properties!
  - Kenneth Arrow, Nobel in economics, 1972

Another desirable property: strategy-proofness

- A social choice function is strategy-proof if agents cannot manipulate the result

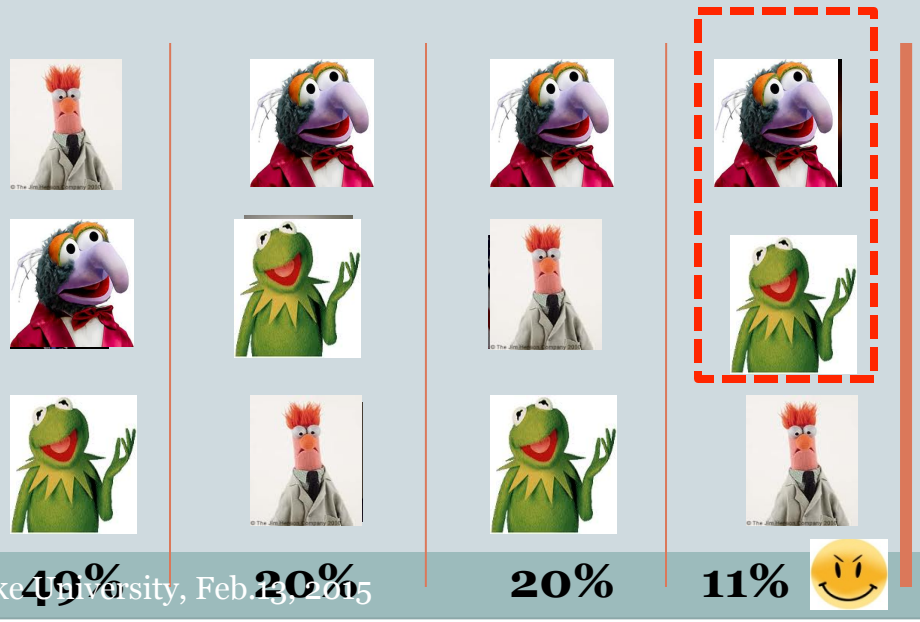
- Manipulation: misreporting his preferences in order to get a better result

Manipulation



49% 20% 20% 11%

Plurality



49% 20% 20% 11%

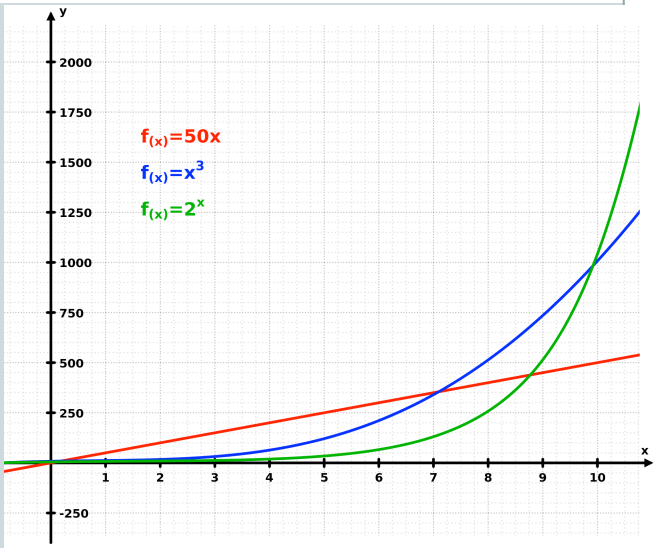
Plurality

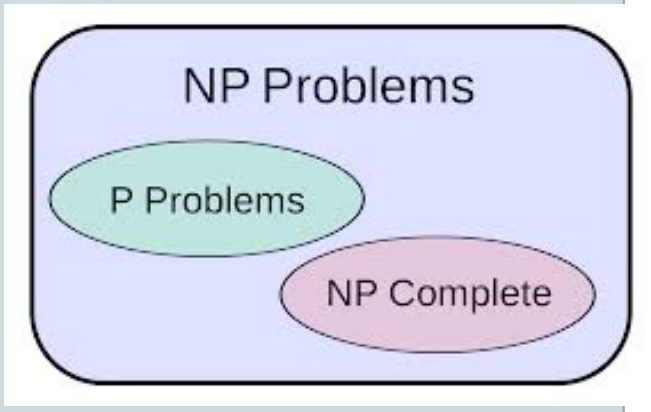
Again not possible

- Unfortunately …. all social choice functions are manipulable!
  - Unless we accept dictatoriality or some candidate that can never win
  - Gibbard-Sattherwite theorems, 1975

Is it difficult to (check if we can) manipulate?

- We consider computational difficulty
  - Exponential time to figure out how to manipulate, knowing how the other agents will vote, in the worst case
  - Easy when it takes always polynomial time
- It depends on the voting rule
  - Easy for Plurality, Borda, and Approval
  - Difficult for other rules, like STV
- We could use voting rules where it is computationally difficult to (check if we can) manipulate!
  - So we mitigate the impossibility result





Is it computationally easy to compute the winner?

- Yes, for most of the rules
  - Plurality, Borda, approval, …
- But not for all
  - Example: Kemeny rule

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

➔ not reasonable to vote independently over the features

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 ➔ $2^{10} = 1024$ 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

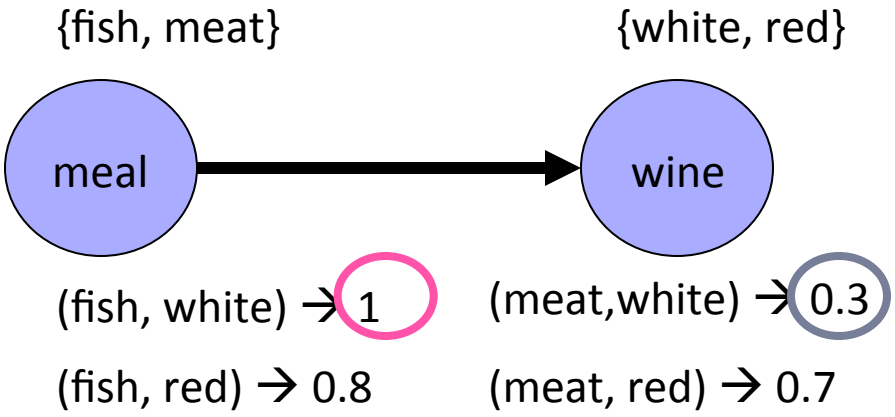
- Sequential voting avoids the problem of electing Condorcet losers
- Thm.: Sequential plurality voting over binary issues never elects a Condorcet loser
  - Proof: Consider the election for the final issue. The winning combination cannot be a CL, since it wins at least against the other combination 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)

How to express preferences compactly over a set of related items?

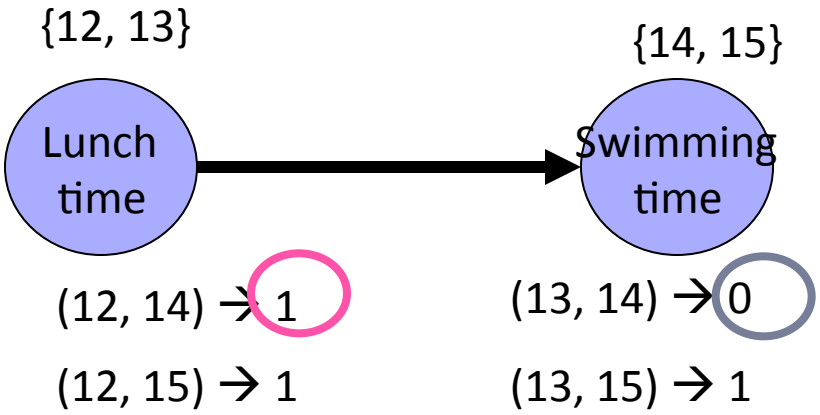
- When there are many candidates, unfeasible to rank them all
  - Think of the dinner example: 625 possible dinners!
- We need a compact way to say what we prefer
- Two main options
  - Quantitative (ex. Soft constraints)
    - Decide on a scale, ex. from 0 to 5, where higher number means more preferred
    - Evaluate some parts of the dinner over this scale
  - Qualitative (ex. CP-nets)
    - Dependence between items, total order on the options for each item

Example: fuzzy constraints

Preference of a decision: minimal preference of its parts
Aim: to find a decision with maximal preference
Preference values: between 0 and 1



Decision *A*	
Lunch time=	13
Meal =	meat
Wine =	white
Swimming time=	14
pref(A)=min(0.3,0)=0	



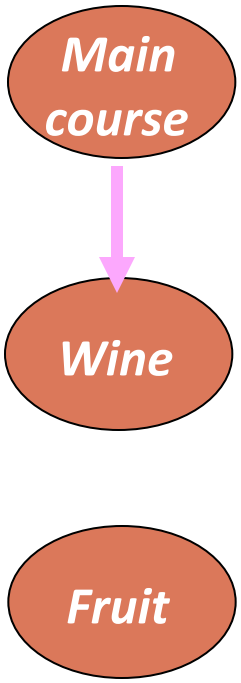
Decision B	
Lunch time =	12
Meal =	fish
Wine =	white
Swimming time =	14
pref(B)=min(1,1)=1	

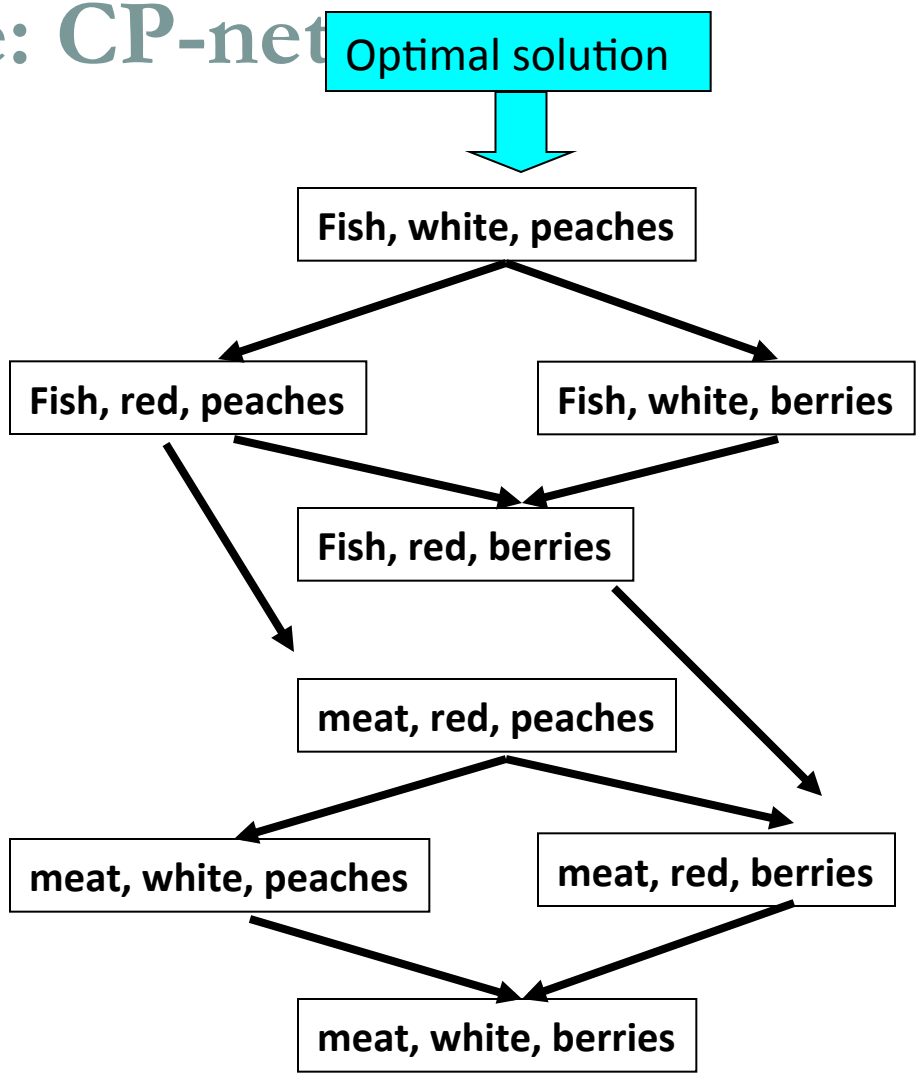
Example: CP-net

fish>meat

Main course	Wine
fish	white > red
meat	red > white

peaches > strawberries





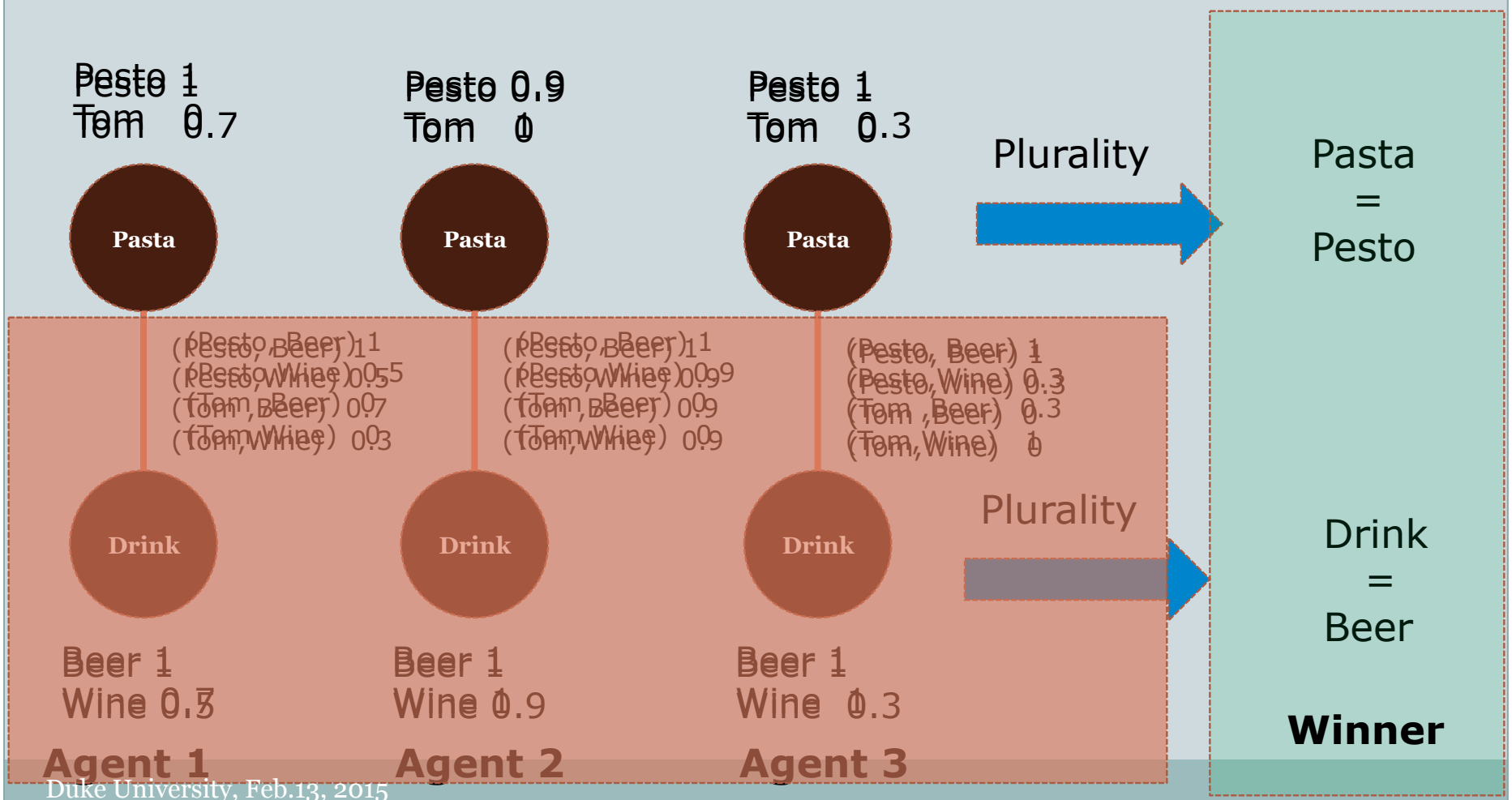
Soft constraints vs. CP-nets

	Soft CSPs	Tree-like soft CSPS	CP-nets	Acyclic CP-nets
Preference orderings	all	all	some	some
Find an optimal decision	difficult	easy	difficult	easy
Compare two decisions	easy	easy	difficult	difficult
Find the next best Decision	difficult	difficult for weighted, easy for fuzzy	difficult	easy
Check if a decision is optimal	difficult	easy	easy	easy

Sequential voting with 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 using plurality



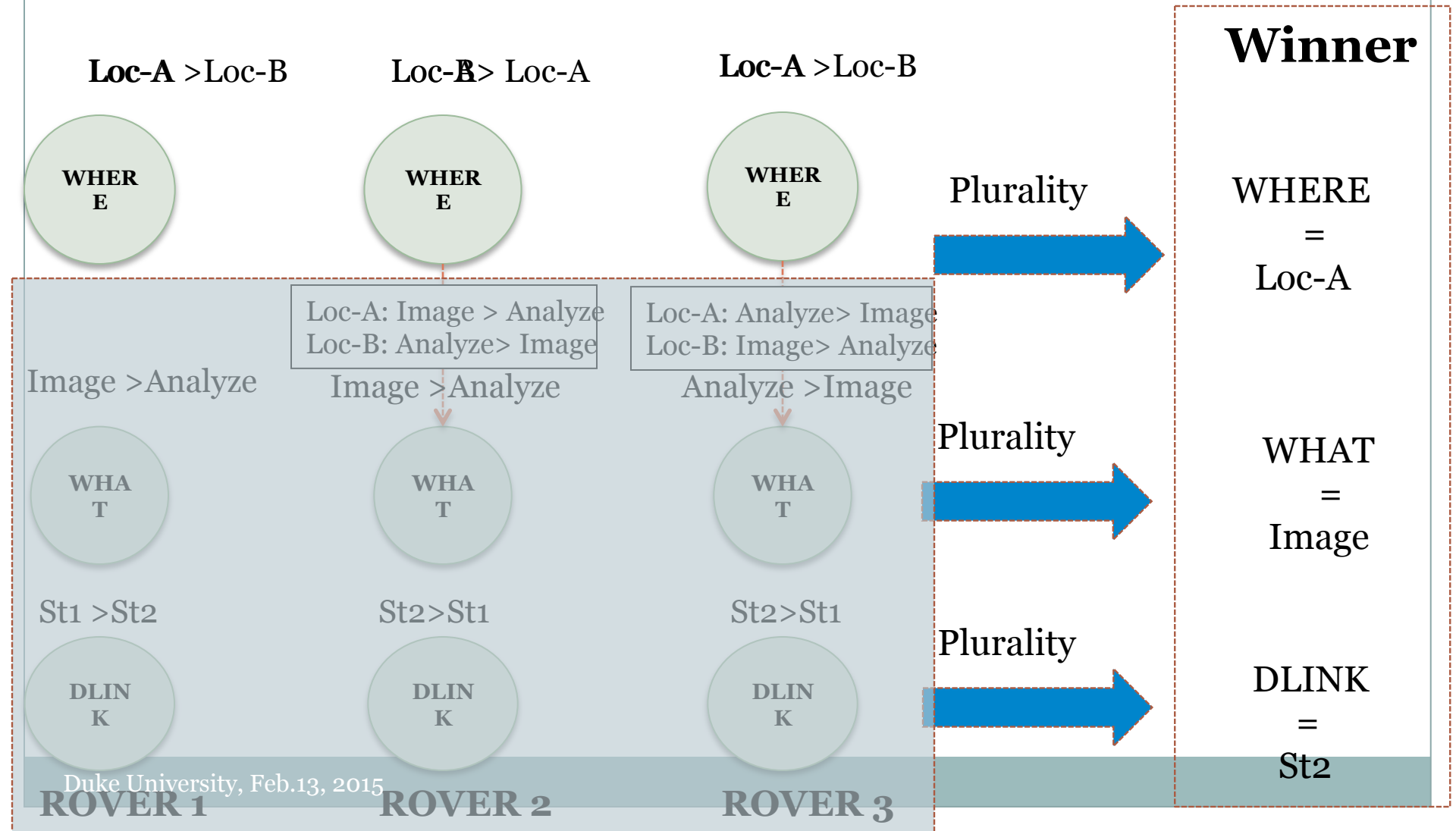
Sequential voting with CP-nets

- 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



Sequential voting: soft constraints vs. CP-nets

- With soft constraints
  - Solving and projecting at each step
  - Tractable in some case (ex. Tree)
  - But no requirement on compatibility among SCSPs
- With CP-nets
  - Directional dependency links
  - Compatibility is required
  - No solving/projecting effort