COMPACT PREFERENCE REPRESENTATION AND MATCHING PROBLEMS

Francesca Rossi

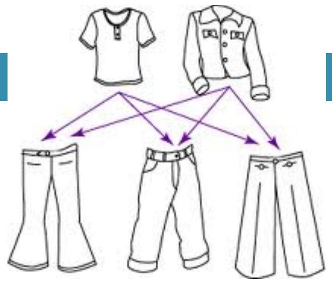
Main differences between social choice and multi-agent AI scenarios

- 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
- Combinatorial structure for the set of decisions
  - Car (or PC, or camera) = several features, each with some instances
  - Dinner = combination of the different dishes

Combinatorial structure for the set of decisions



- Example:
  - Three friends need to decide what to cook for dinner
  - 4 items (pasta, main, dessert, drink)
  - 5 options for each ➔ $5^4 = 625$ possible dinners
- In general: Cartesian product of several variable domains
  - Variables = items of the menu, domain= 5 options

Formalisms to model preferences

- Preference ordering over a large set of decisions ➔ need to model them compactly
  - Otherwise too much space and time to handle such preferences
- Two examples:
  - soft constraints
  - CP-nets

Outline

- Compact representation of preferences
  - Soft constraints
  - CP nets
- Sequential voting
- Stable marriage problems

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]

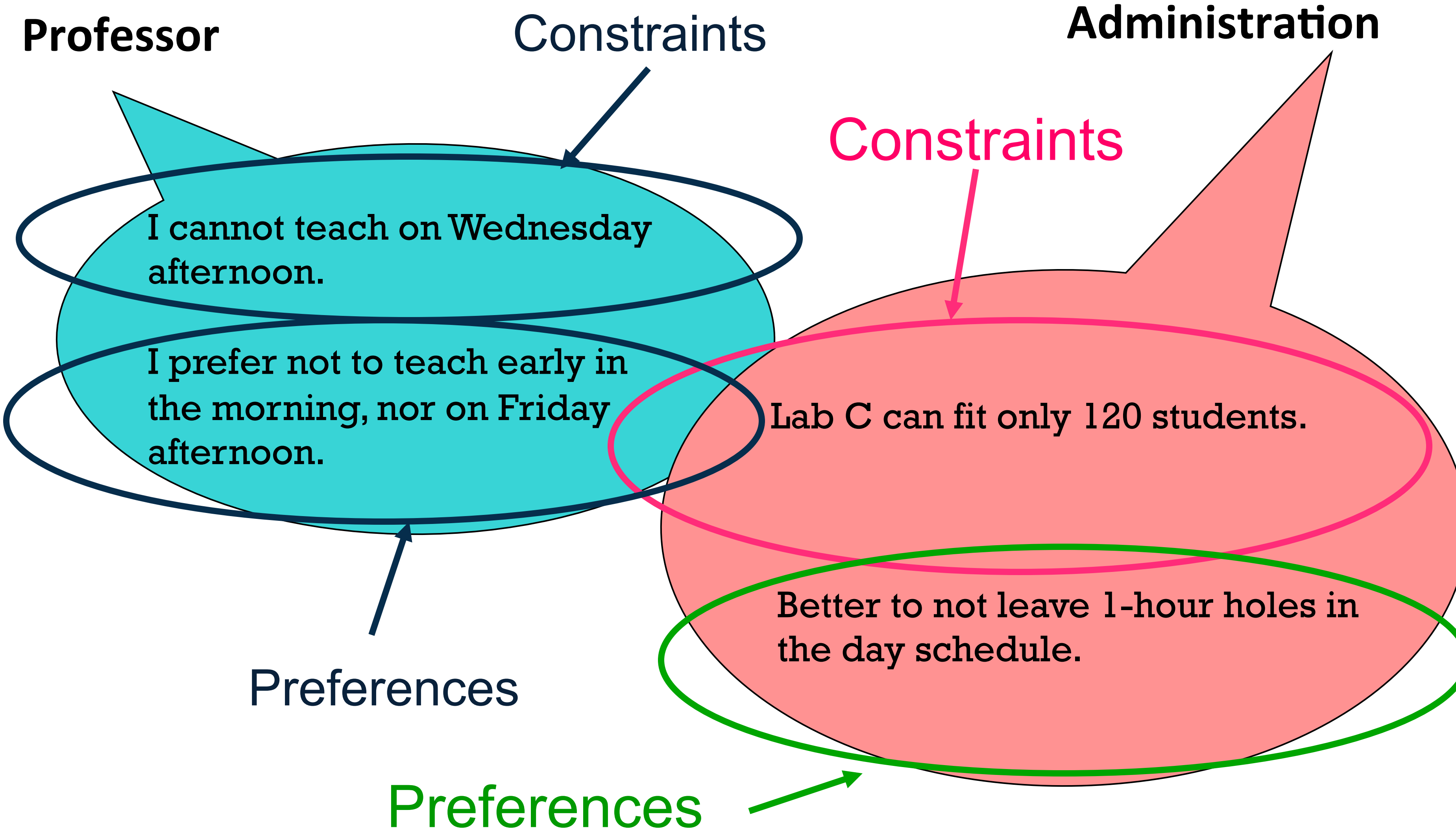
Constraints are not flexible

- 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



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

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 $\langle A,+,x,\mathbf{0},\mathbf{1}\rangle$

[Bistarelli, Montanari, Rossi, IJCAI 1995, JACM 1997]

Soft constraints

- Soft constraint: a pair c=<f,con> where:
  - Scope: con={X^c_1,…, X^c_k} subset of X
  - Preference function :

    f: $D(X^c_1) x \ldots x D(X^c_k) \rightarrow A$

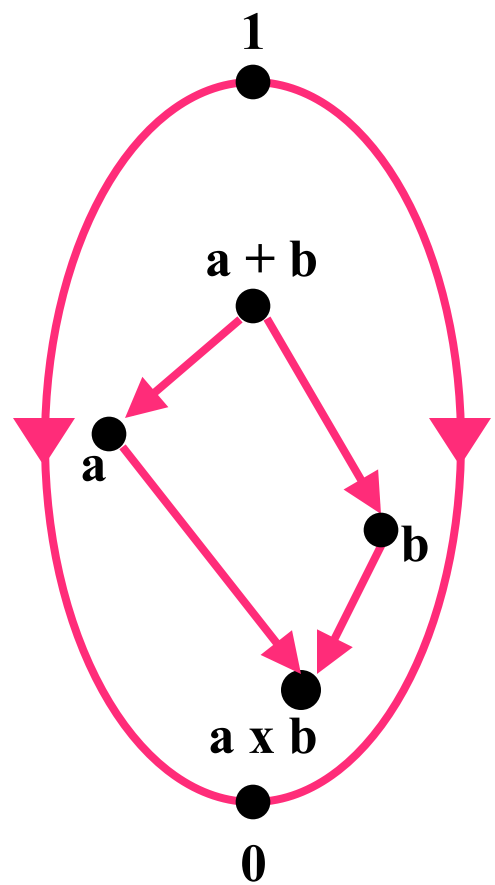
    tuple $(v_1,\ldots, v_k) \rightarrow p$ *preference*
- Hard constraint: a soft constraint where for each tuple $(v_1,\ldots, v_k)$

  $f(v_1,\ldots, v_k)=1$ the tuple is allowed

  $f(v_1,\ldots, v_k)=0$ 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,≤> lattice
    - + is lub
    - x is glb if x idempotent
  - + and x monotone on ≤



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

- $<\mathbf{A} = N \cup +\infty, \mathbf{+} = \min, \mathbf{x} = +, \mathbf{0} = +\infty, \mathbf{1} = 0>$
- 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

- $\langle A = [0,1], + = max, x = min, \mathbf{0} = 0, \mathbf{1} = 1 \rangle$:
  - 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 (eg., space and medical settings)

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

Fuzzy-SCSP example

{Fish, Meat} {White, red}

(Fish, white) → 1

(Fish, red) → 0.8

(Meat, white) → 0.3

(Meat, red) → 0.7

{12 pm, 1 pm} {2 pm, 3 pm}

(12 pm, 2 pm) → 1

(12 pm, 3 pm) → 1

(1 pm, 2 pm) → 0

(1 pm , 3 pm) → 1

Fuzzy semiring

$$S = \langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$$

$$S_{FCSP} = \langle [0,1], max, min, 0, 1 \rangle$$

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

Instances of semiring-based soft constraints

- Each instance is characterized by a c-semiring <A, +, x, 0, 1>
- Classical constraints: <{0,1},logical or,logical and,0,1>
  - Satisfy all constraints
- Fuzzy constraints: <[0,1],max,min,0,1>
  - Maximize the minimum preference
- Lexicographic CSPs: $<[0,1]^k$,lex-max,min,0^k,$1^k>$
  - Order the preferences lexicographically and then maximize the minimum preference
- Weighted constraints (N):$<N \cup +\infty$, min, +,$+\infty$,0>
  - Minimize the sum of the costs (naturals)
- Weighted constraints (R):$<R \cup +\infty$, 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], max, x, 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 $S_i = <A_i, +_i, x_i, 0_i, 1_i>$, we can build the c-semiring

$<<A_1,..., A_n>, +, x, <0_1,...,0_n>,<1_1,...,1_n>>$

- + 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],max,min,0,1>
  - <N, min, +, +∞, 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

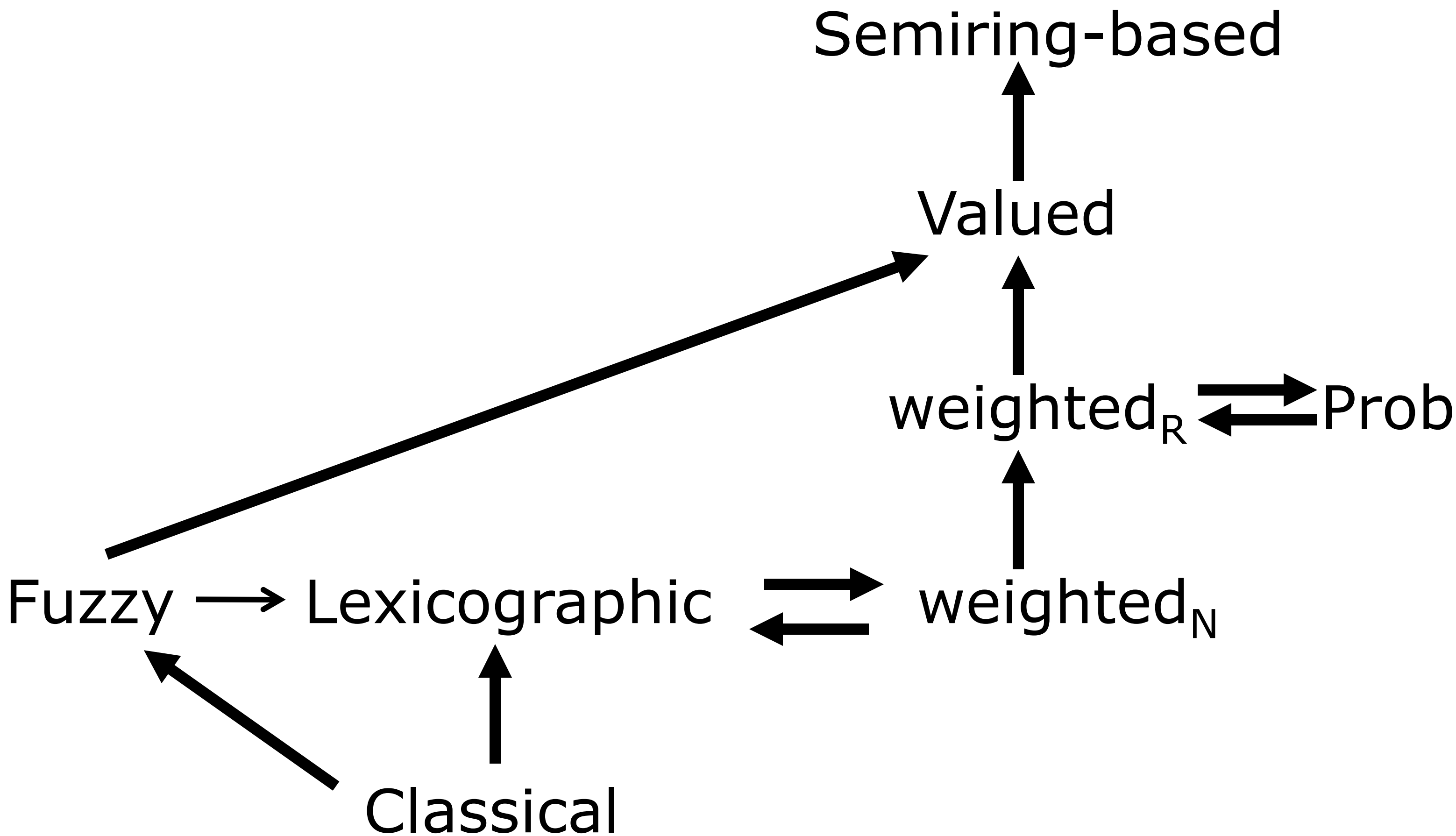
Solution ordering

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

Expressive power

- $A \longrightarrow B$ iff from a problem P in A it is possible to build in 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
- $A \longrightarrow B$ iff from a problem P in A it is possible to build in polynomial time a problem P' in B s.t. $opt(P') \subseteq opt(P)$

Expressive power



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]

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

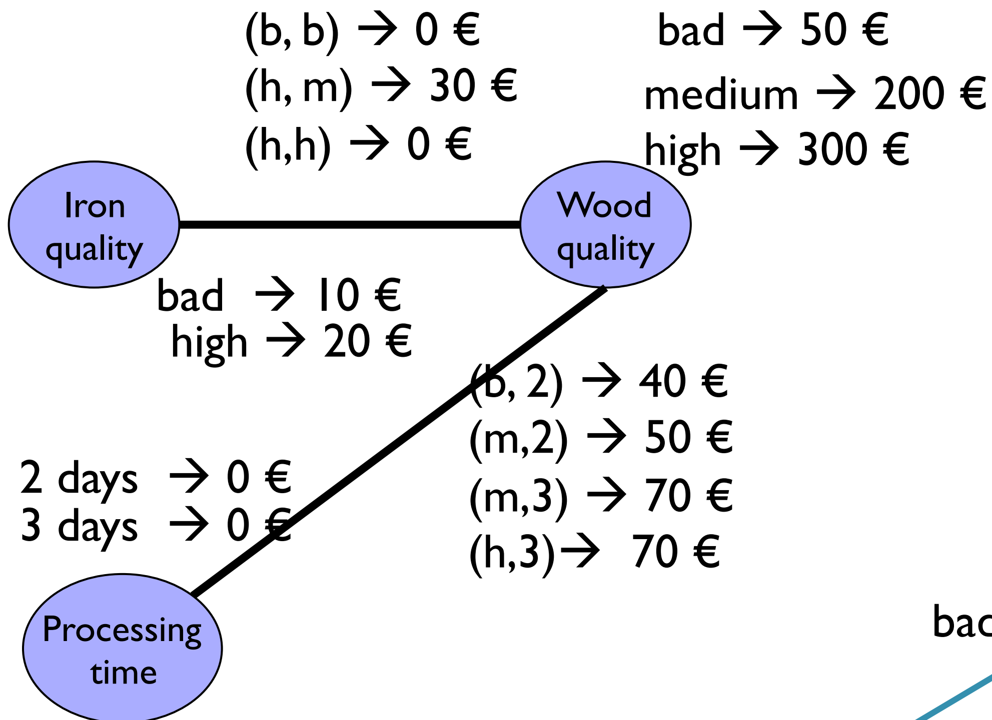
- Backtracking → **Branch and Bound**
- Main idea:
  - visit each assignment that may be a solution
  - skip only assignments that are shown to be dominated by others
- **Search tree** to represent the space of all assignments

Systematic search : Branch and bound

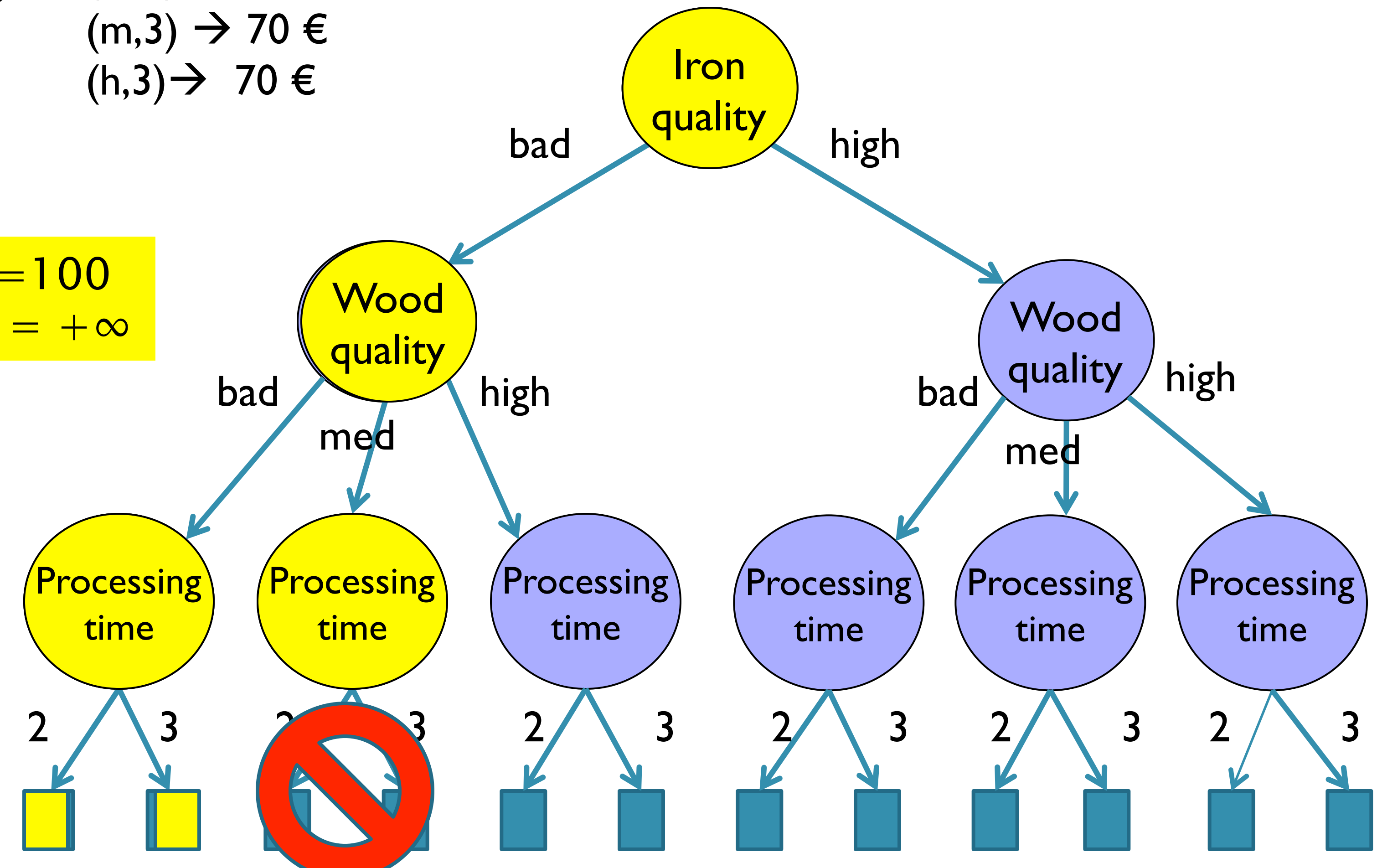
- **Lower bound** = preference of best solution so far (0 at the beginning)
- **Upper bound for each node**: upper bound to the preference of any assignment in the subtree rooted at the node
- If **ub is worst than lb** ➔ prune subtree

$S_{WCSP} = \langle [0,+\infty], \min, +, +\infty, 0 \rangle$

ub = x preferences from constraints on assigned variables



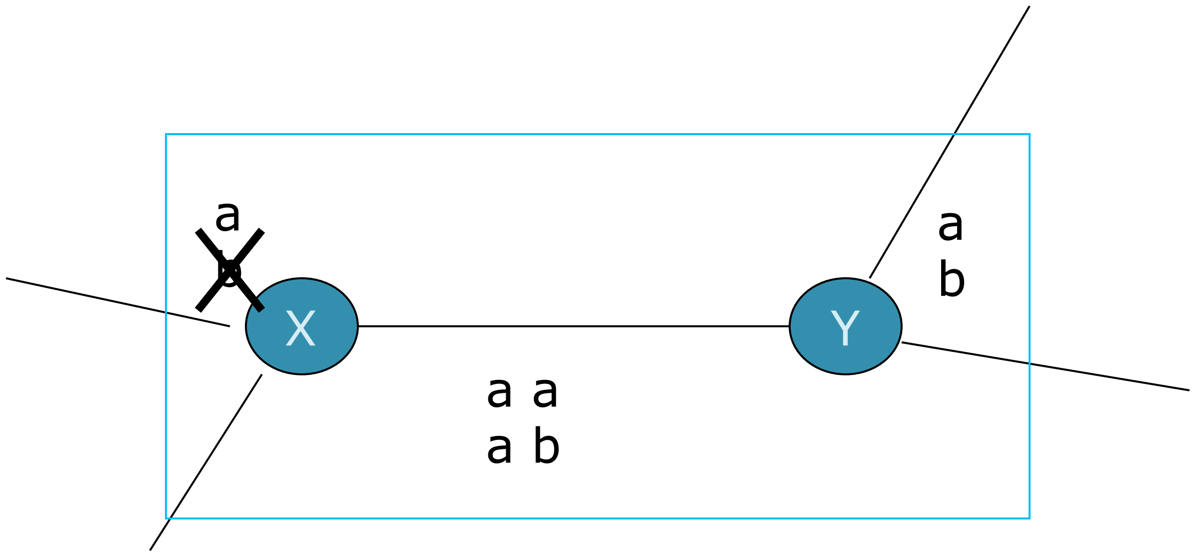
lb = 100
ub = $+\infty$



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

Example



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,con>, then $c|_I$ = <f', $I \cap$ con>
  - $f'(t') = +(f(t))$ over tuples of values t s.t. $t|_{I \cap con} = t'$

- **Combination**: combine two or more soft constraints obtaining a new soft constraint "synthesizing " all the information of the original ones

  Formally: If ci=<fi,coni>, then c1 x c2 = <f, con1 $\cup$ con2>
  - $f(t) = f1(t|_{con1}) \times f2(t|_{con2})$

Projection: fuzzy example

$S_{FCSP}=\langle[0,1],max,min,0,1\rangle$

If $c=\langle f,con\rangle$, then $c|_I = \langle f', I \cap con\rangle$

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

$c=\langle f,\{mc,w\}\rangle$

{Fish,Meat}

{White ,Red}

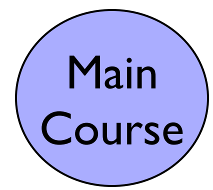
(Fish, white) → 1

(Fish, red) → 0.8

(Meat, white) → 0.3

(Meat, red) → 0.7

$c|_{mc}$



Fish → max(f(fish,white),f(fish,red))
=max(1,0.8)=1

Meat → max(f(meat,white),f(meat,red))
=max(0.3,0.7)=0.7

Projection: weighted example

$S_{WCSP} = \langle [0,+\infty], min, +, +\infty, 0 \rangle$

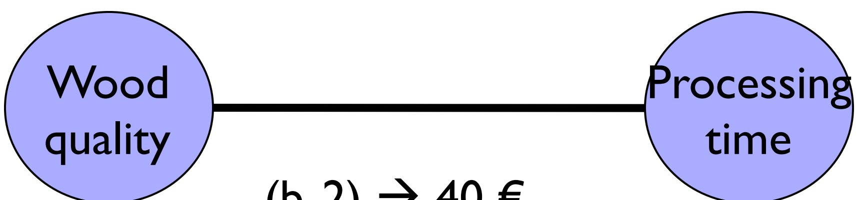
If $c = \langle f, con \rangle$, then $c|_I = \langle f', I \cap con \rangle$

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

$c = \langle f, \{wq, pt\} \rangle$

{bad,med,high}

{2,3}



(b, 2) → 40 €
(m,2) → 50 €
(m,3) → 70 €
(h,3) → 70 €

$c|_{wq}$

bad → min(f(b,2),f(b,3))=min(40,+∞)=40

med → min(f(m,2),f(m,3))=min(50,70)=50

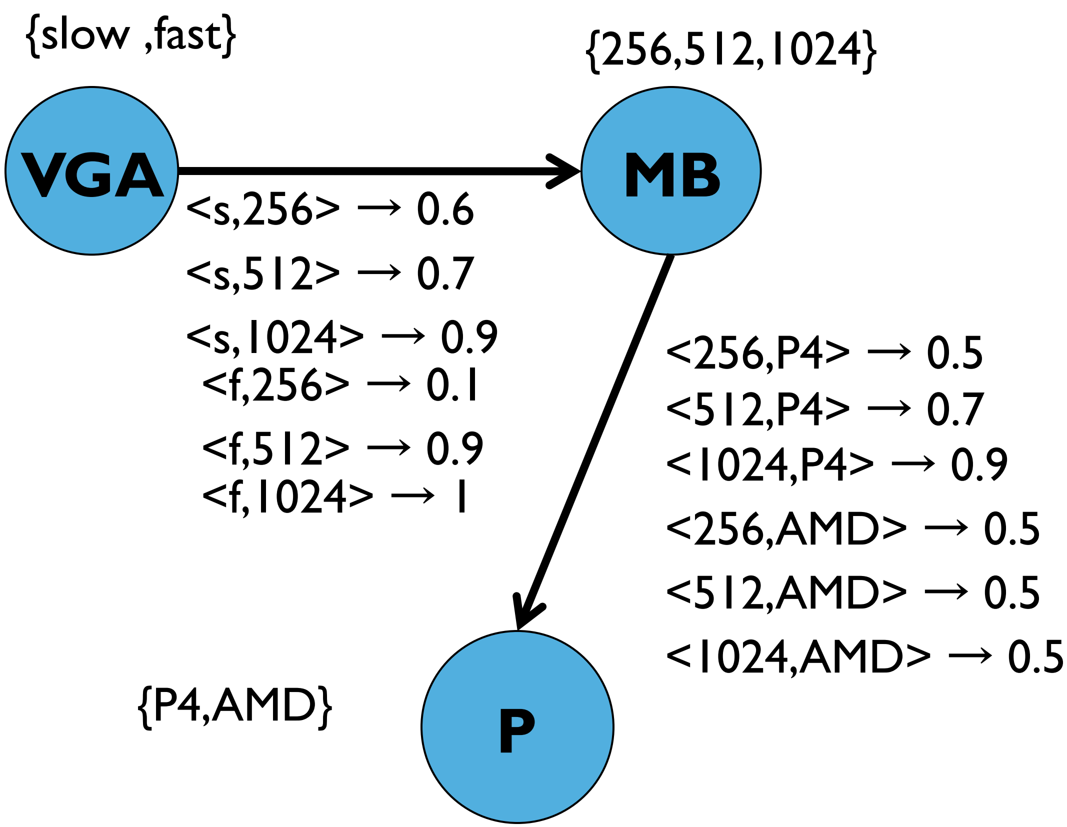
high → min(f(h,2),f(h,3))=min(+∞,70)=70

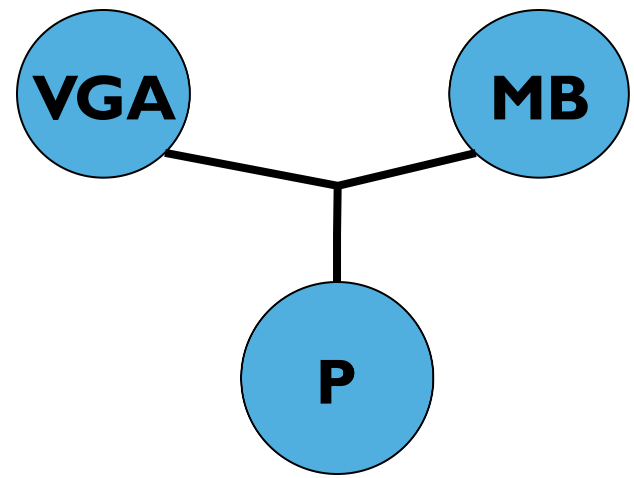
Combination: fuzzy example

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

- $f(t) = f1(t|_{con1}) \times f2(t|_{con2})$

$S_{FCSP}=\langle[0,1],max,min,0,1\rangle$





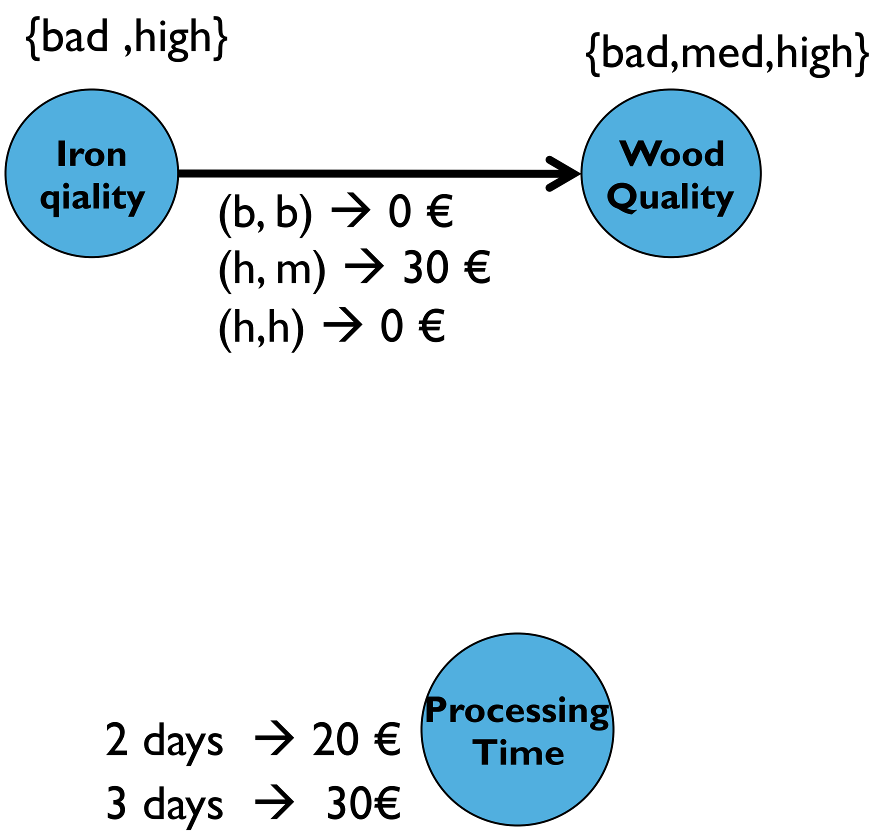
f(s,256,P4) = min(0.6,0.5) = 0.5
f(f,1024,P4)=min(0.9,0.9)=0.9
….

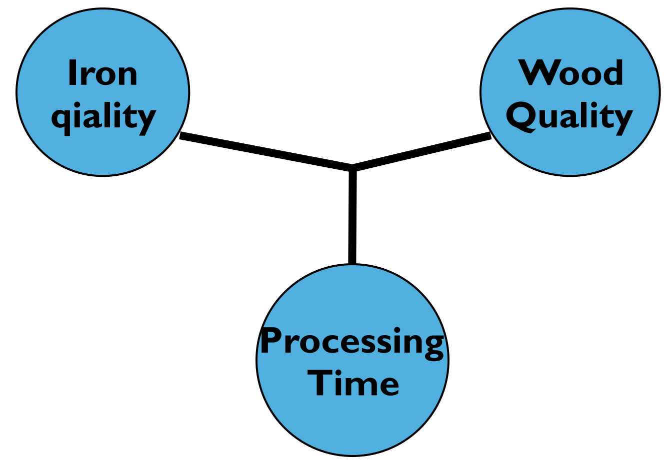
$S_{WCSP}=<[0,+\infty],min,+,+\infty,0>$

Combination: weighted example

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

- $f(t) = f1(t|_{con1}) \times f2(t|_{con2})$





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 <{0,1},or,and,0,1>
- 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

s=slow → 0.2
f=fast → 0.9

256 → 0.5
512 → 0.8
1024 → 0.7

$c_x := (c_x \times c_{xy} \times c_y)|_x$

Fuzzy semiring=
<[0,1],max,min,0,1>
→ +=max an x=min

<s,256> → 0.6 <f,256> → 0.1

<s,512> → 0.7 <f,512> → 0.9

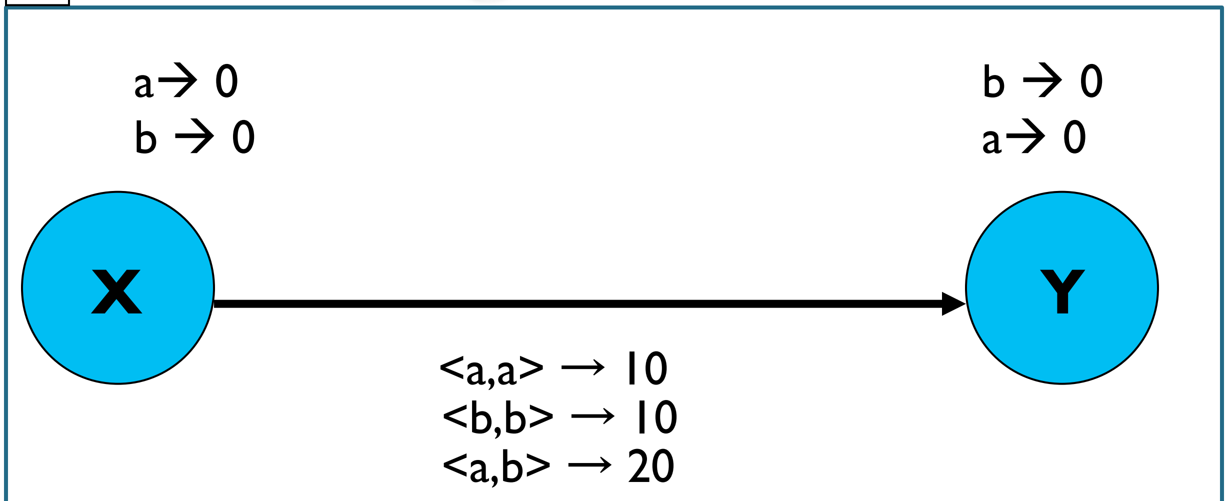
<s,1024> → 0.9 <f,1024> → 1

VGA=s → 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→ 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

weighted arc-consistency ?!

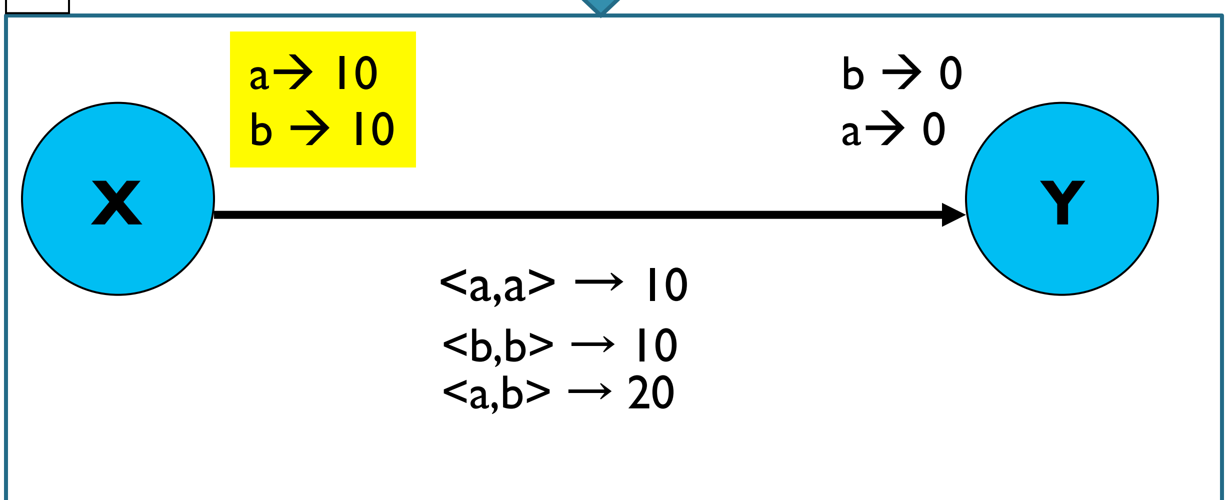
1



Weighted semiring= $<[0,+\infty],min,+,+\infty,0>$

$c_x := (c_x \text{ x } c_{xy} \text{ x } c_y)|_x$

2



Not equivalent!

pref1(a,a)=10

pref2(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 ➔ we need to compensate by subtracting appropriate quantities somewhere else ➔ we need an additional property (fairness=presence of -)
  - Equivalence
  - Termination
  - Not order-independence

[Schiex, CP 2000]

References for preferences and soft constraints

- Semiring-based Constraint Solving and Optimization, S. Bistarelli, U. Montanari and F. Rossi, Journal of ACM, vol.44, n.2, 1997
- Bucket Elimination: A Unifying Framework for Reasoning, R. Dechter, AI Journal 113, 1-2, 1999
- Preference Handling for Artificial Intelligence, J. Goldsmith, U. Junker eds, AI Magazine, 2008
- 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
- Arc Consistency for Soft Constraints, Thomas Schiex, Proc. CP 2000,Springer LNCS 1894

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

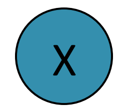
- Conditional preference statements
  - “If it is fish, I prefer white wine to red wine”
  - syntax:

    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}

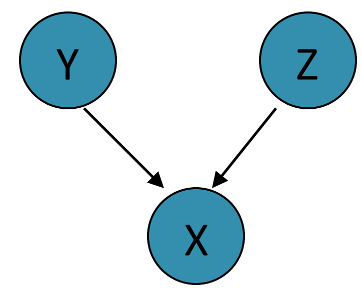
[Boutelier, Brafman, Domshlak, Hoos, Poole. JAIR 2004]
[Domshlak, Brafman KR02]

CP nets

- Variables $\{X_1, \ldots, X_n\}$ with domains
- For each variable, a total order over its values
- Indipendent variable:
  - X=v1 > X=v2 > ... > X=vk



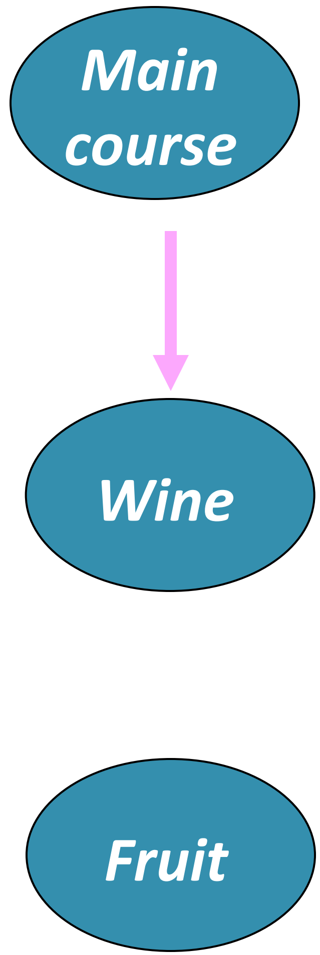
- Conditioned variable: a total order for each combination of values of some other variables (conditional preference table)
  - Y=a, Z=b: X=v1 > X=v2 > ... > X=vk
  - X depends on Y and Z (parents of X)



- Graphically: directed graph over $X_1, \ldots, X_n$
  - Possibly cyclic

CP nets: an example

Independent feature



fish>meat

Dependent feature

Conditional Preference Table

Main course	*Wine*
fish	white > red
meat	red > white

Independent feature

peaches > strawberries

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)

$\downarrow$ worsening flip

(fish, red wine, peaches)

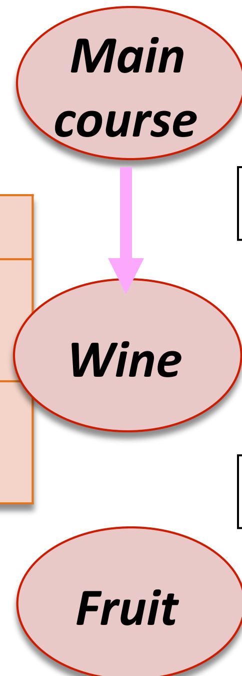
- An outcome O_1 is preferred to O_2 iff there is a sequence of worsening flips from O_1 to O_2
- Optimal outcome: if no other outcome is preferred

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

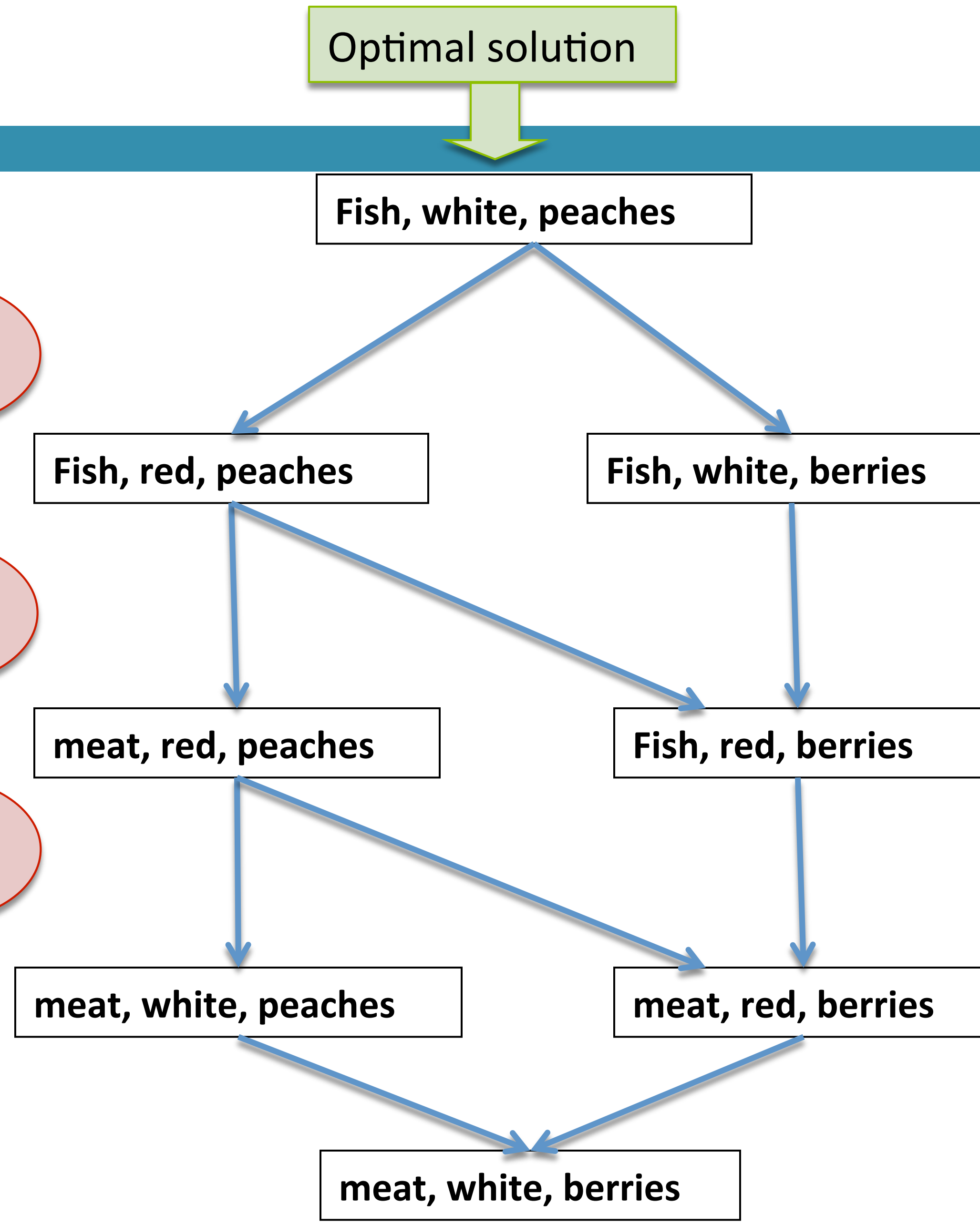
Solution ordering

fish>meat



Main course	Wine
fish	white > red
meat	red > white

peaches > strawberries

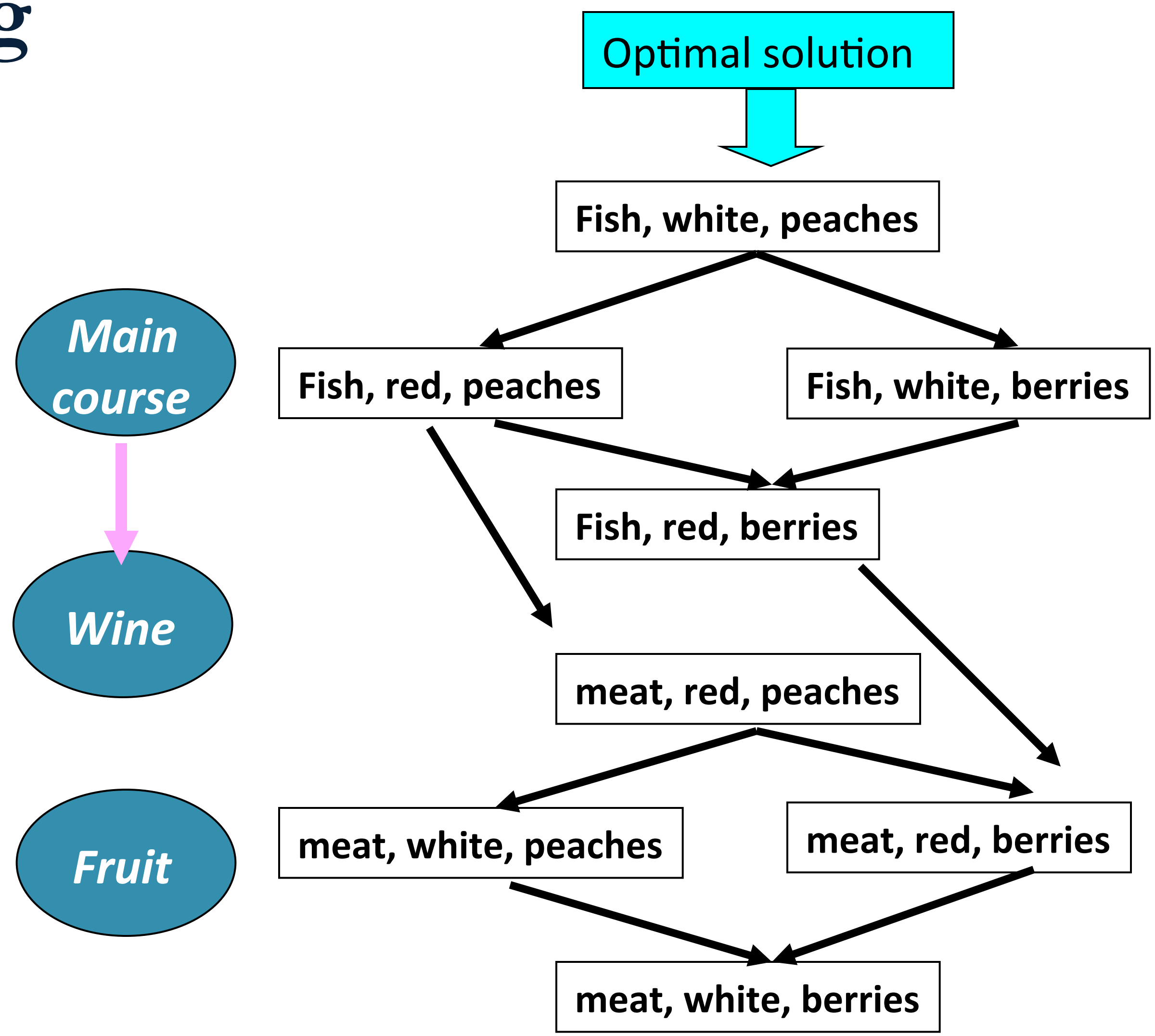


Solution ordering

fish>meat

Main course	Wine
fish	white > red
meat	red > white

peaches > strawberries



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

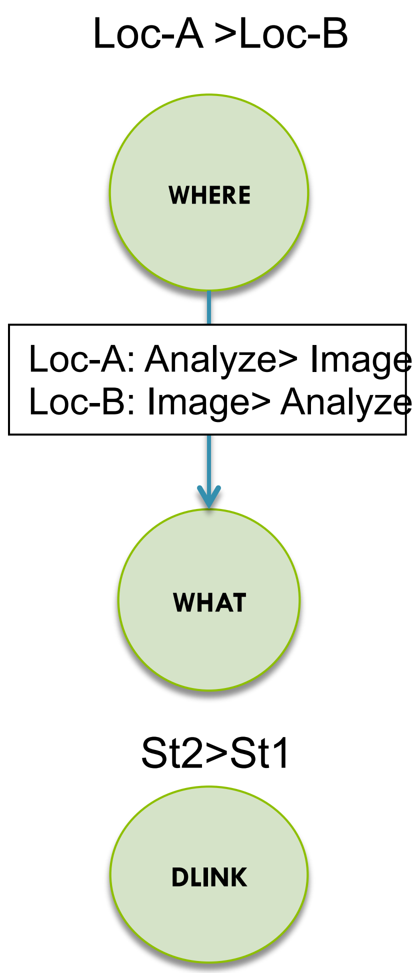
- Does O1 dominate O2?
  - Difficult even for acyclic CP nets

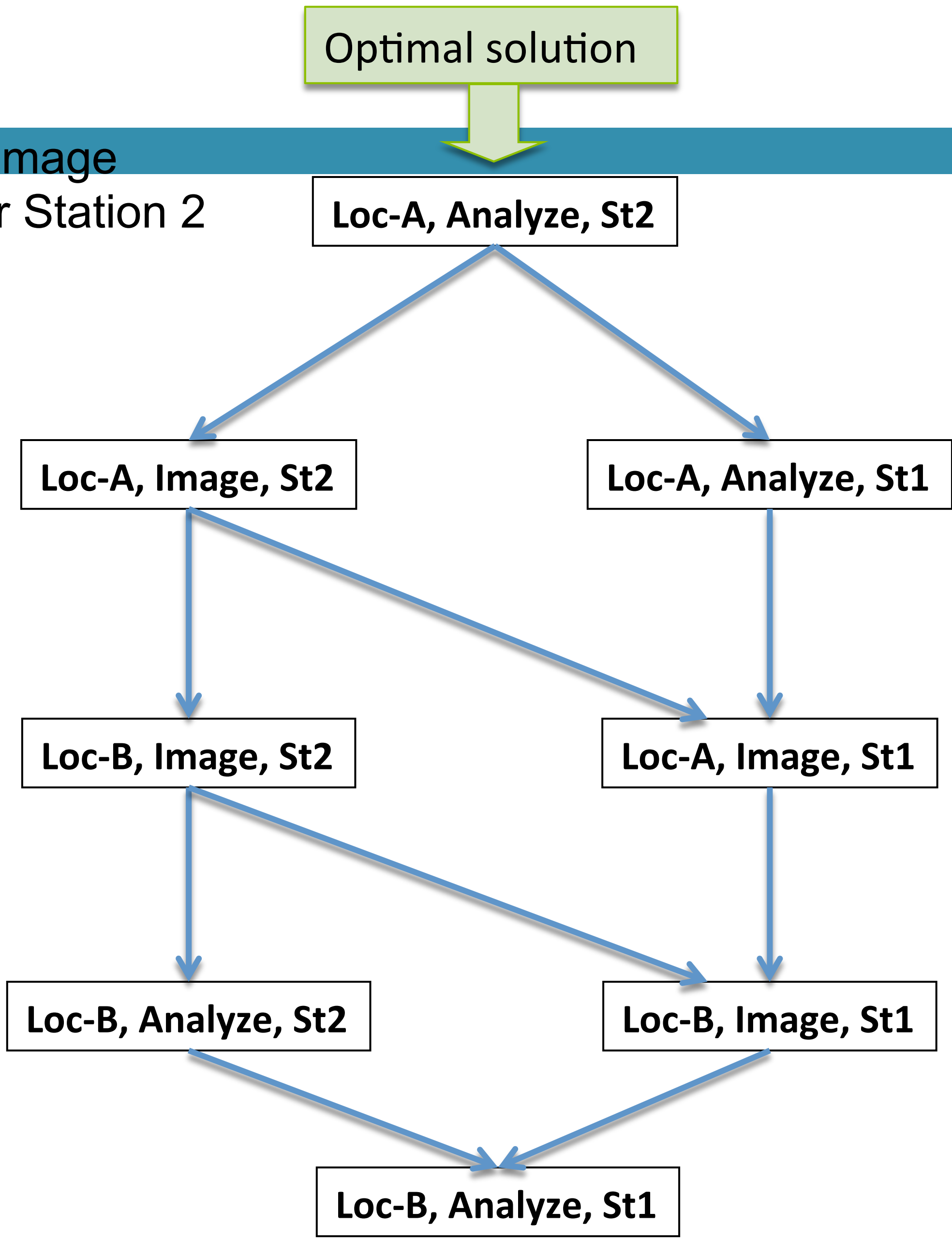
- Is O optimal?
  - Easy: test O against a CSP

Example

A rover must decide:

- Where to go: Location A or Location B
- What to do: Analyze a rock or Take and image
- Which station to downlink to: Station 1 or Station 2



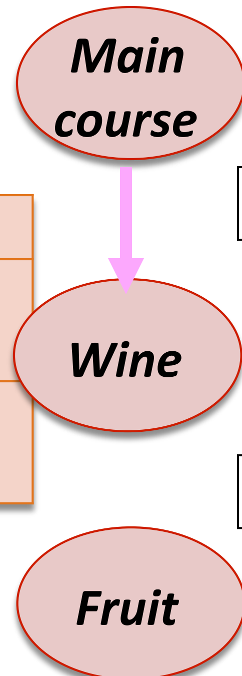


How to find optimal solutions in CP nets

- 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

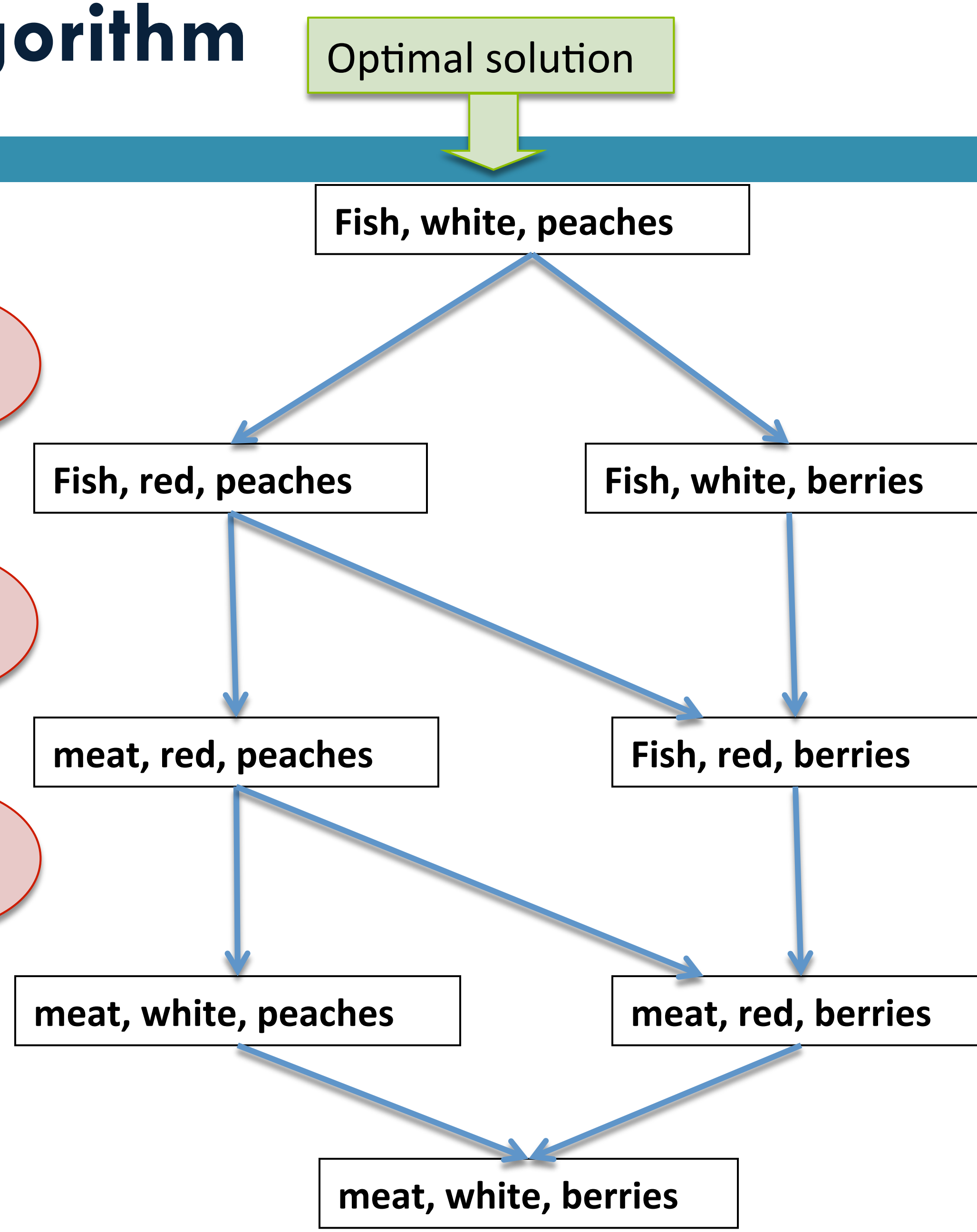
fish>meat



Main course	Wine
fish	white > red
meat	red > white

peaches > strawberries

1. F = peaches
2. M = fish
3. Since M=fish, W=white

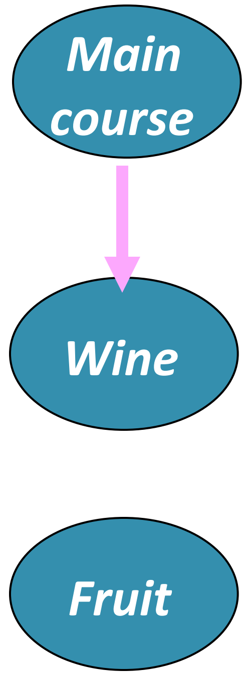


Sweep forward algorithm

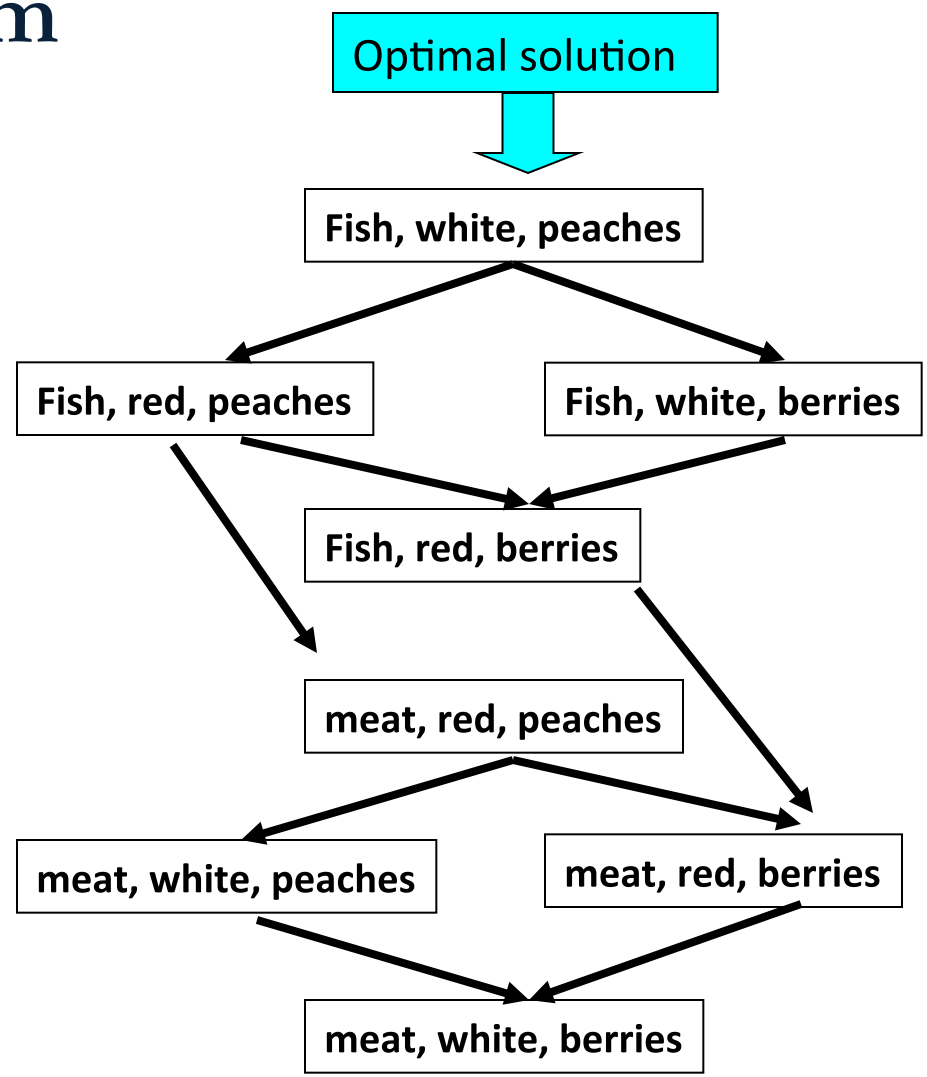
fish>meat

Main course	*Wine*
fish	white > red
meat	red > white

peaches > strawberries



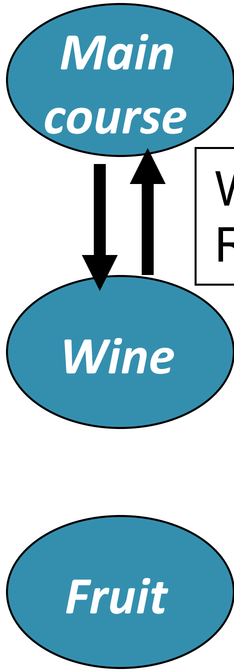
1. F = peaches
2. M = fish
3. Since M=fish, W=white



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



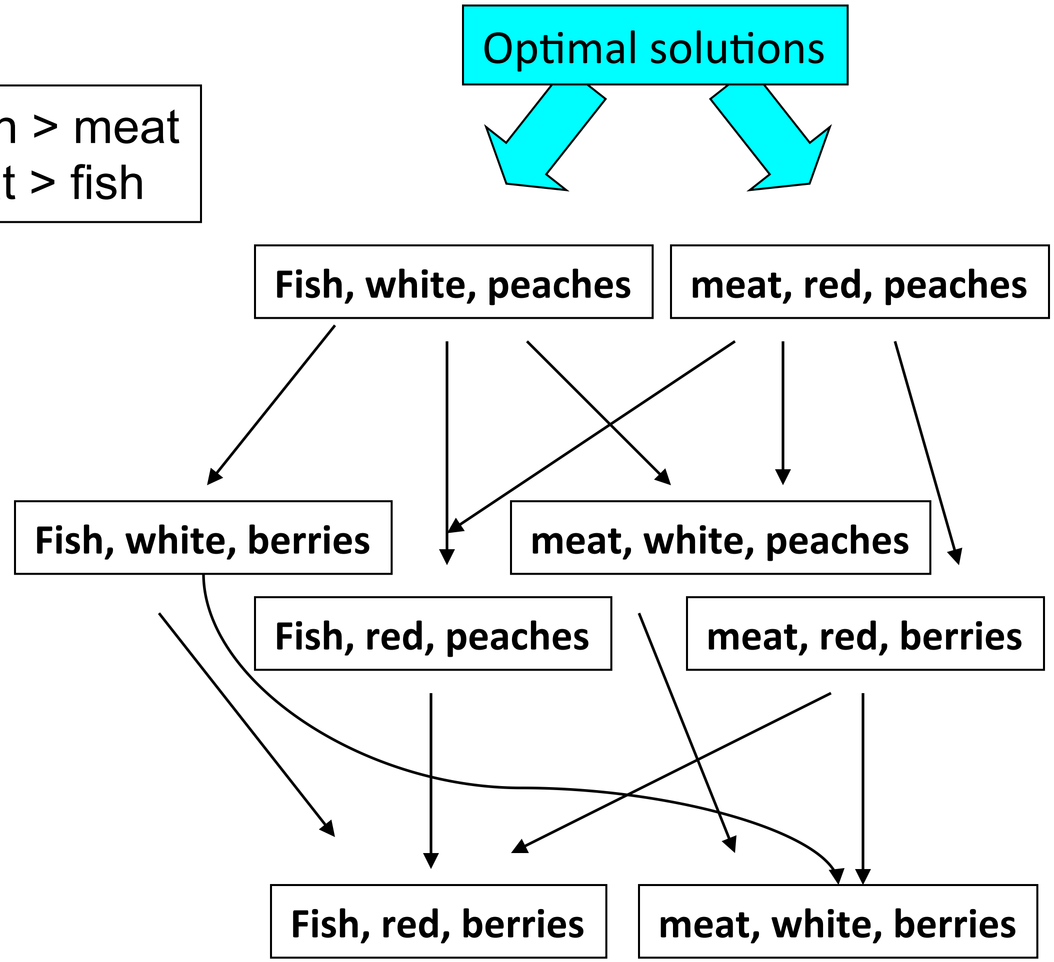
Fish: white > red
Meat: red > white

White: fish > meat
Red: meat > fish

peaches > strawberries

Constraints:

F = peaches
M = fish ➔ W=white
M = meat ➔ W = red
W = white ➔ M = fish
W = red ➔ M = meat



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,I)

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

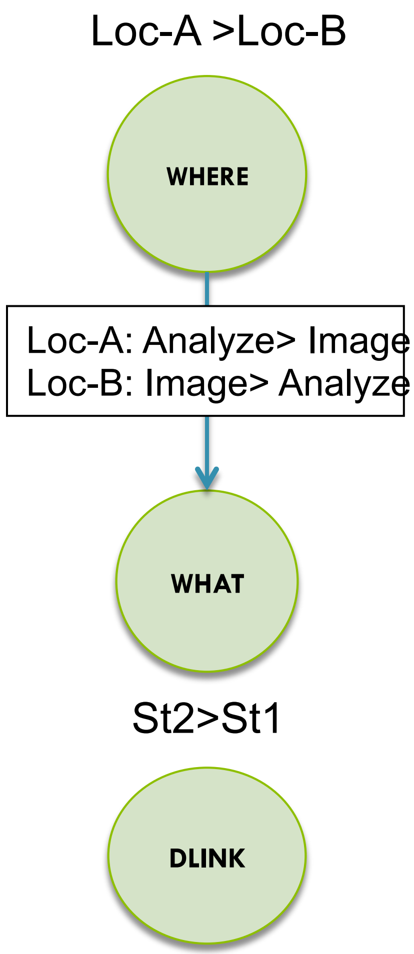
P: Preference representation

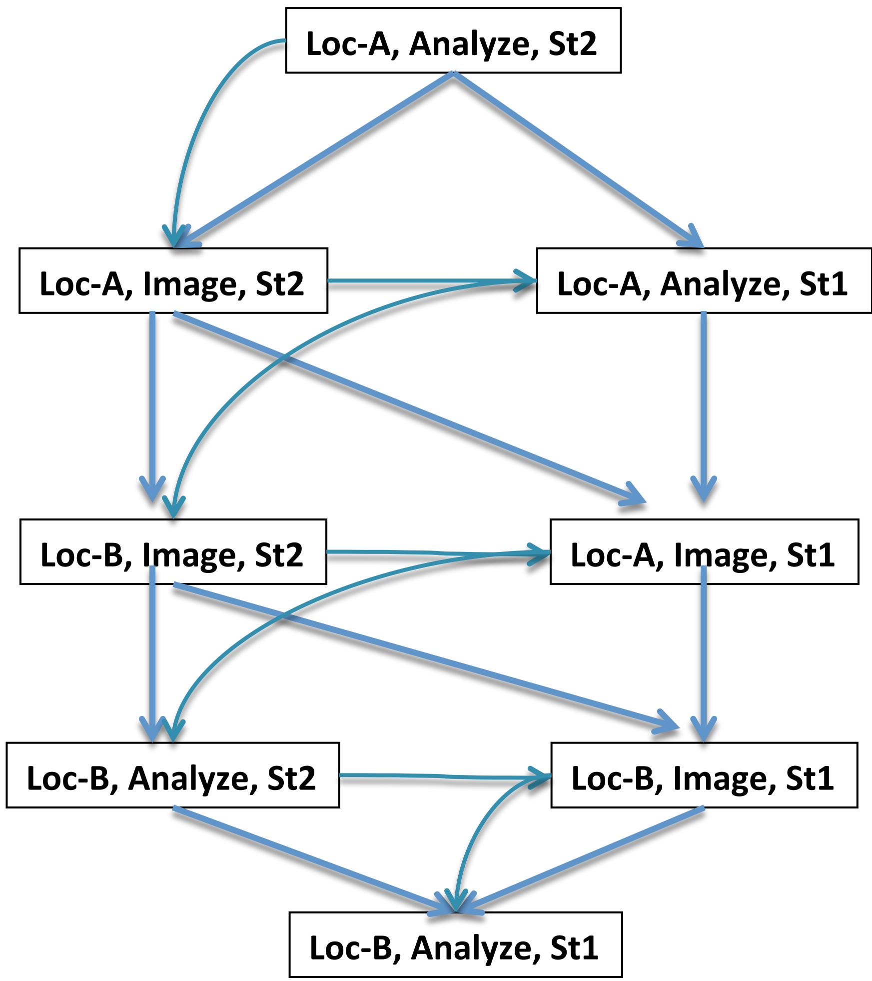
s: Solution

I: Linearization of the preference ordering

[Brafman, Salvagnin Rossi,Venable,Walsh ADT 2009, KR 2010, AAAI 2011]

Next on a CP-net: example



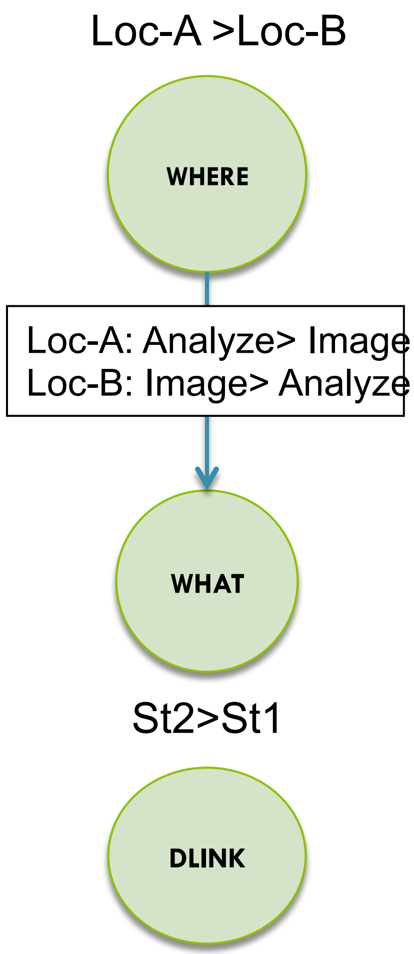


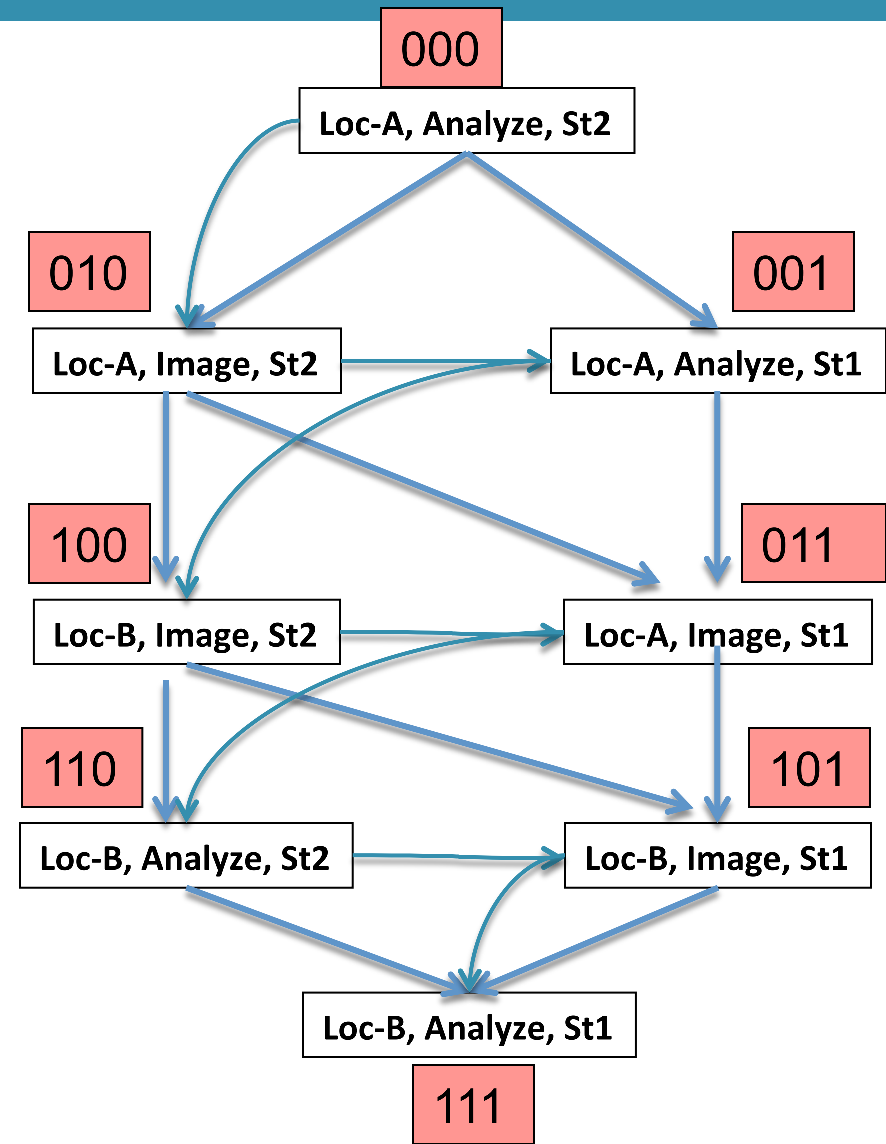
Next on acyclic CP-nets is easy for conditional lex linearization

- 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 xi: 0 if xi 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'

[Brafman, Salvagnin Rossi,Venable,Walsh, ADT 2009, KR 2010, AAAI 2011]

Solution Ordering

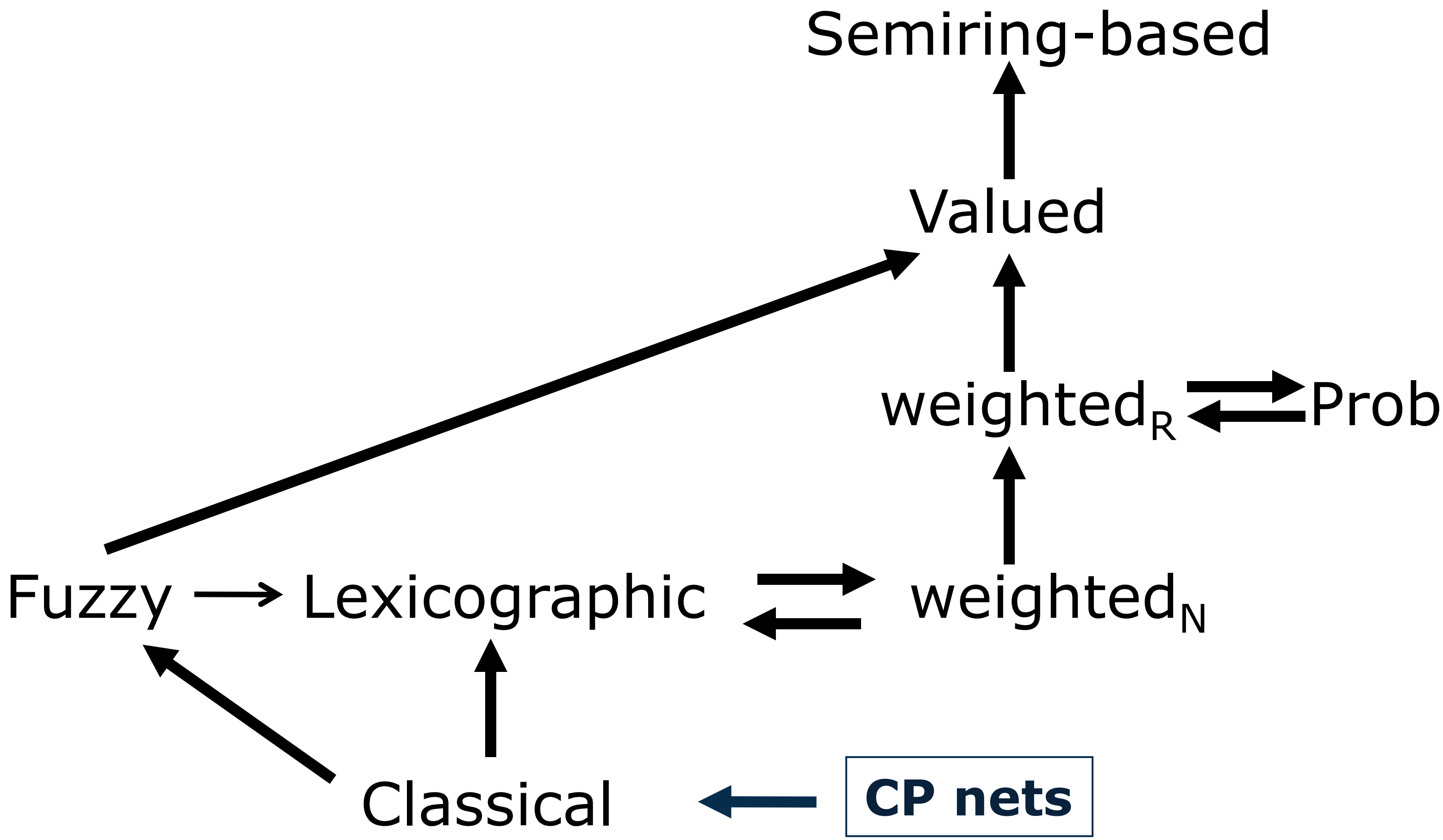




[Brafman, Salvagnin Rossi,Venable,Walsh ADT 2009, KR 2010, AAAI 2011]

Expressive power

If interested in the optimal solutions:



CP nets → classical CSPs

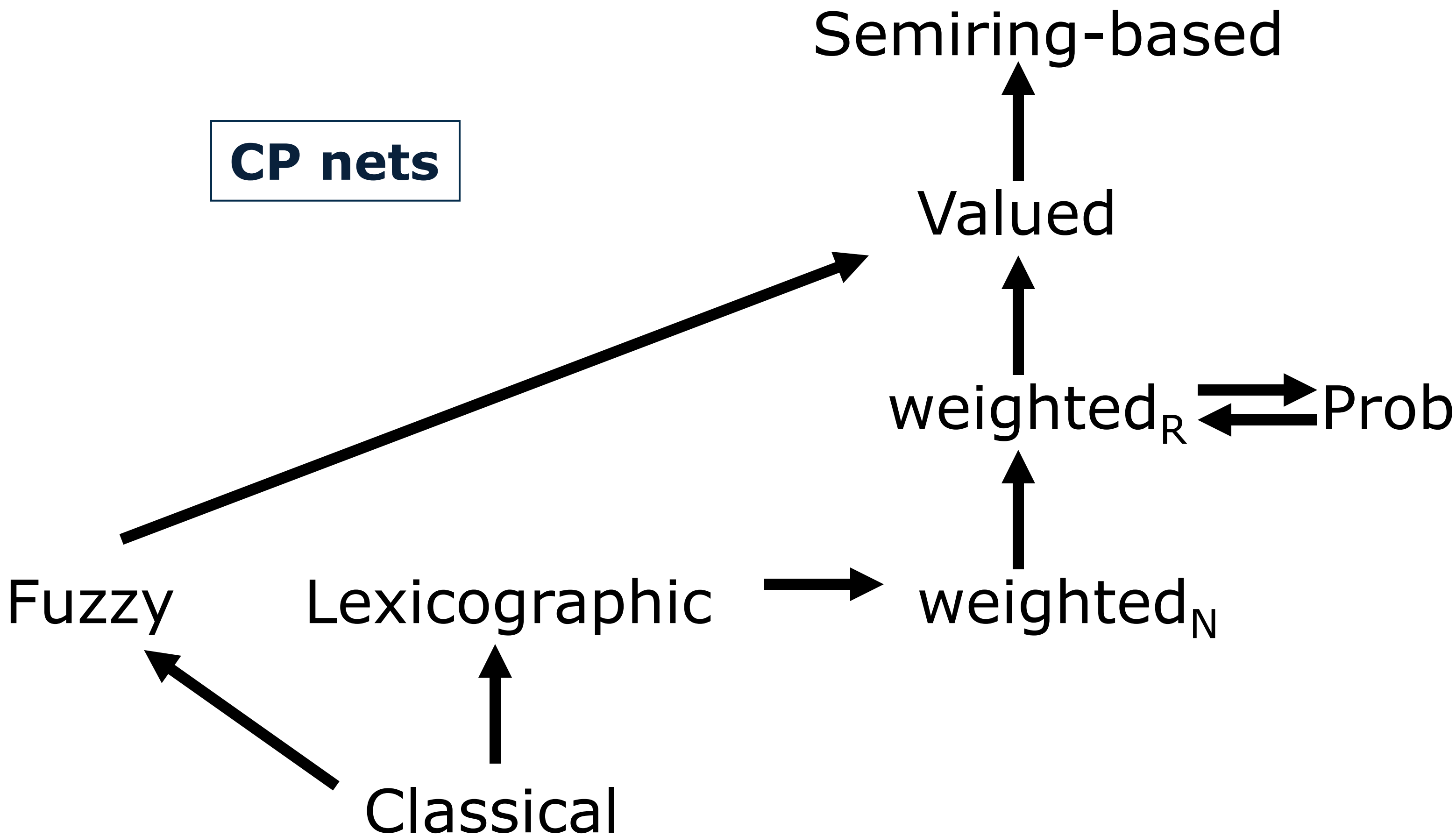
- Given a CP net, it is always possible to build in polynomial time a classical CSP with the same set of optimal solutions
  - For each Y=a, Z=b: X=v1 > X=v2 > ... > X=vk, we build the constraint Y=a, Z=b ➔ X=v1

[Brafman, Dimopoulos, CI 2004]

- 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> ➔ they must be ordered in a CP net

Expressive power

If interested in maintaining the solution ordering:



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

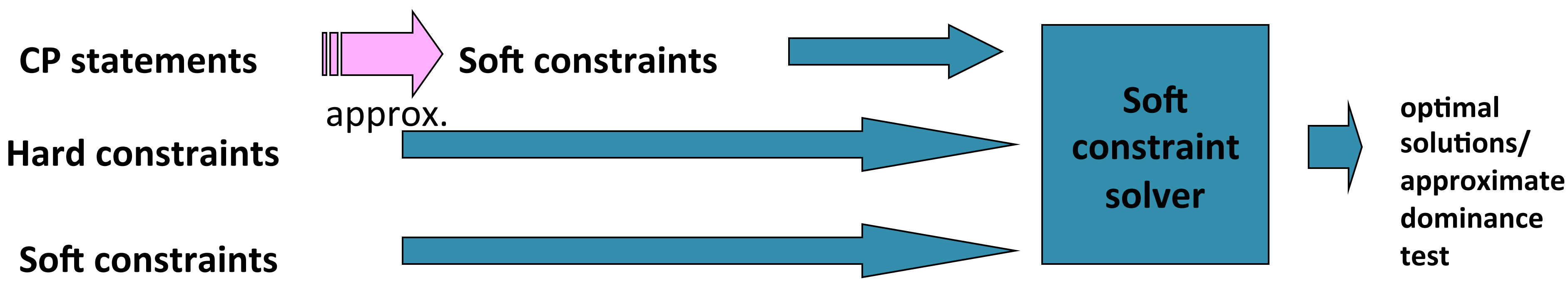
- There are soft CSPs whose orderings cannot be modelled by a CP net
  - Not all orderings can be represented by CP nets

Soft constraints vs. CP-nets

	Soft constraints	CP nets (acyclic)
Preference orderings	all	some
Find an optinmal decision	difficult	easy
Compare two decisions	easy	difficult
Find the next best decision	difficult	easy
Check if a decsion is optimal	difficult	easy

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 → more ordered
  - Easy dominance test



[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

- Pref(O1) > pref(O2) in C, or
- Pref(O1) = pref(O2) 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)

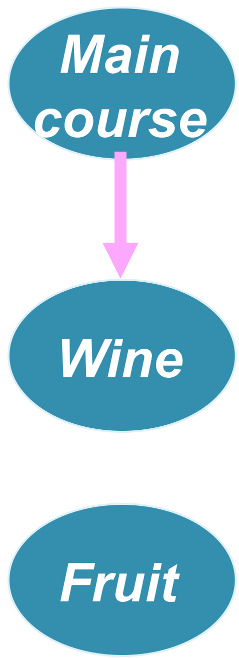
Softly Constrained CP net : example

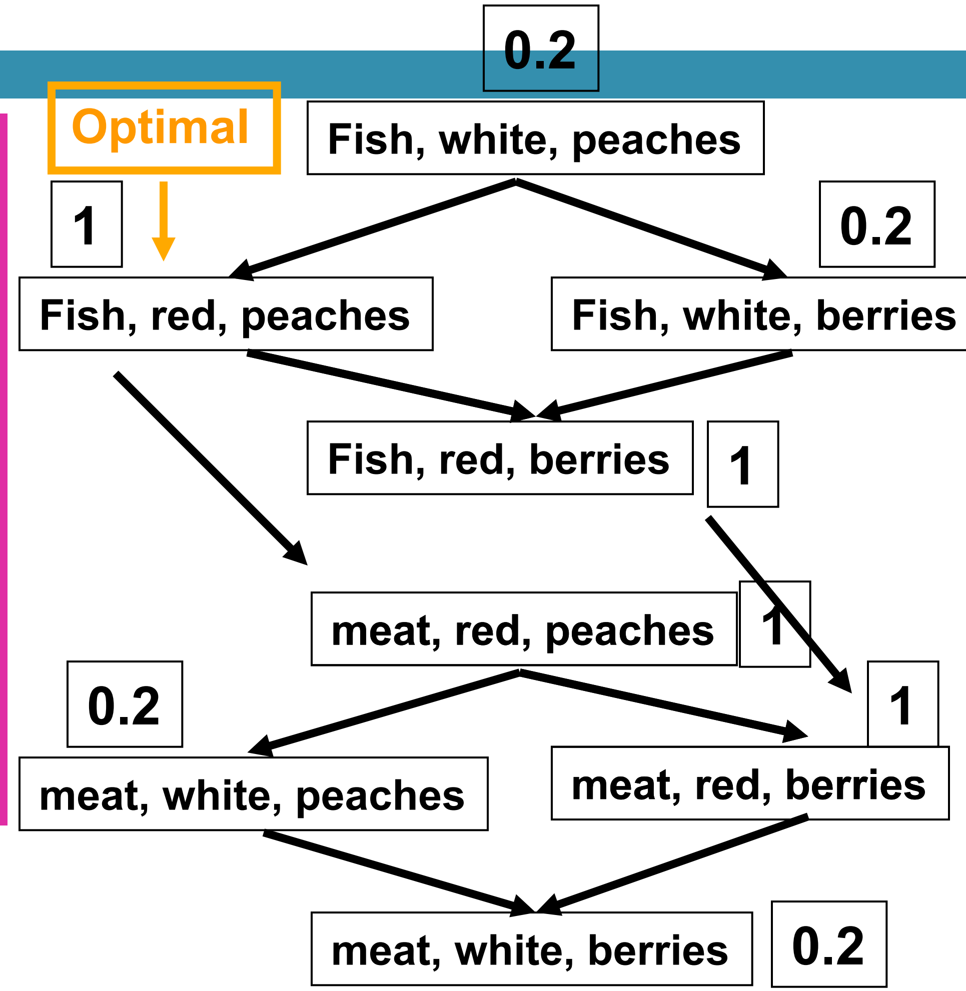
CP net

fish>meat

Main course	Wine
fish	white > red
meat	red > white

peaches > strawberries





Soft Constraint

Wine

white → 0.2
red → 1

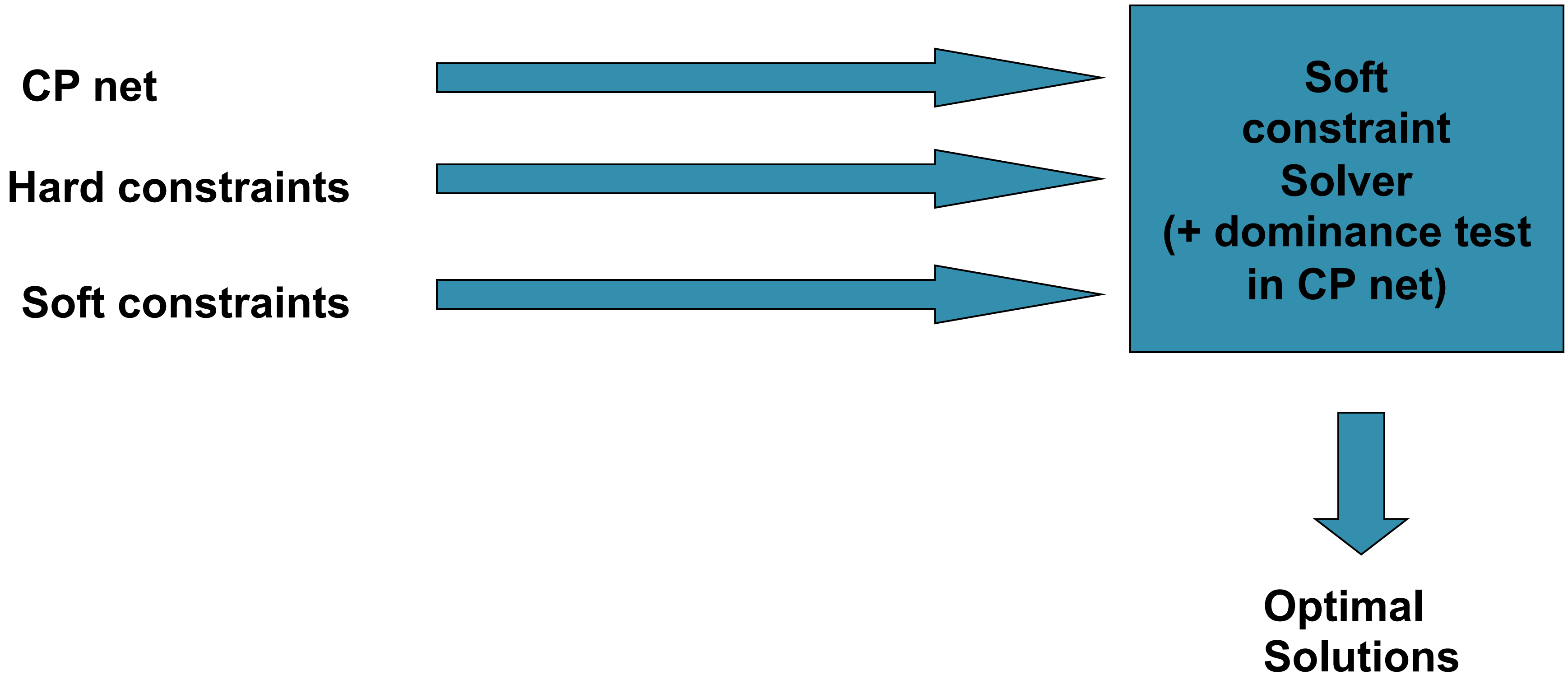
How to obtain an optimal outcome of a constrained CP net <N,C>

- From N to optimality constraints OC
- If Sol(OC ∪ C) is not empty, then they are (some of the) optimal outcomes ➔ take one of them

  ➔ only hard constraint solving
- Otherwise, dominance testing between feasible outcomes (more costly)

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

(Conditional + qualitative + quantitative) preferences + constraints



References for CP-nets

- Extended semantics and optimization algorithms for CP-networks, R. Brafman and Y. Dimopoulos, Computational Intelligence, 20(2), 2004
- CP-nets: A Tool for Representing and Reasoning with Conditional Ceteris Paribus Preference Statements, C. Boutilier, R. Brafman, C. Domshlak, H. Hoos, D. Poole JAIR, 21, 2004
- Reasoning about soft constraints and conditional preferences: complexity results and approximation techniques, C. Domshlak, F. Rossi, K. B. Venable, T. Walsh, Proc. IJCAI 2003
- CP-nets Reasoning and Consistency Testing, C. Domshlak, R. Brafman, KR 2002
- 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 ➔ $2^{10} = 1024$ 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

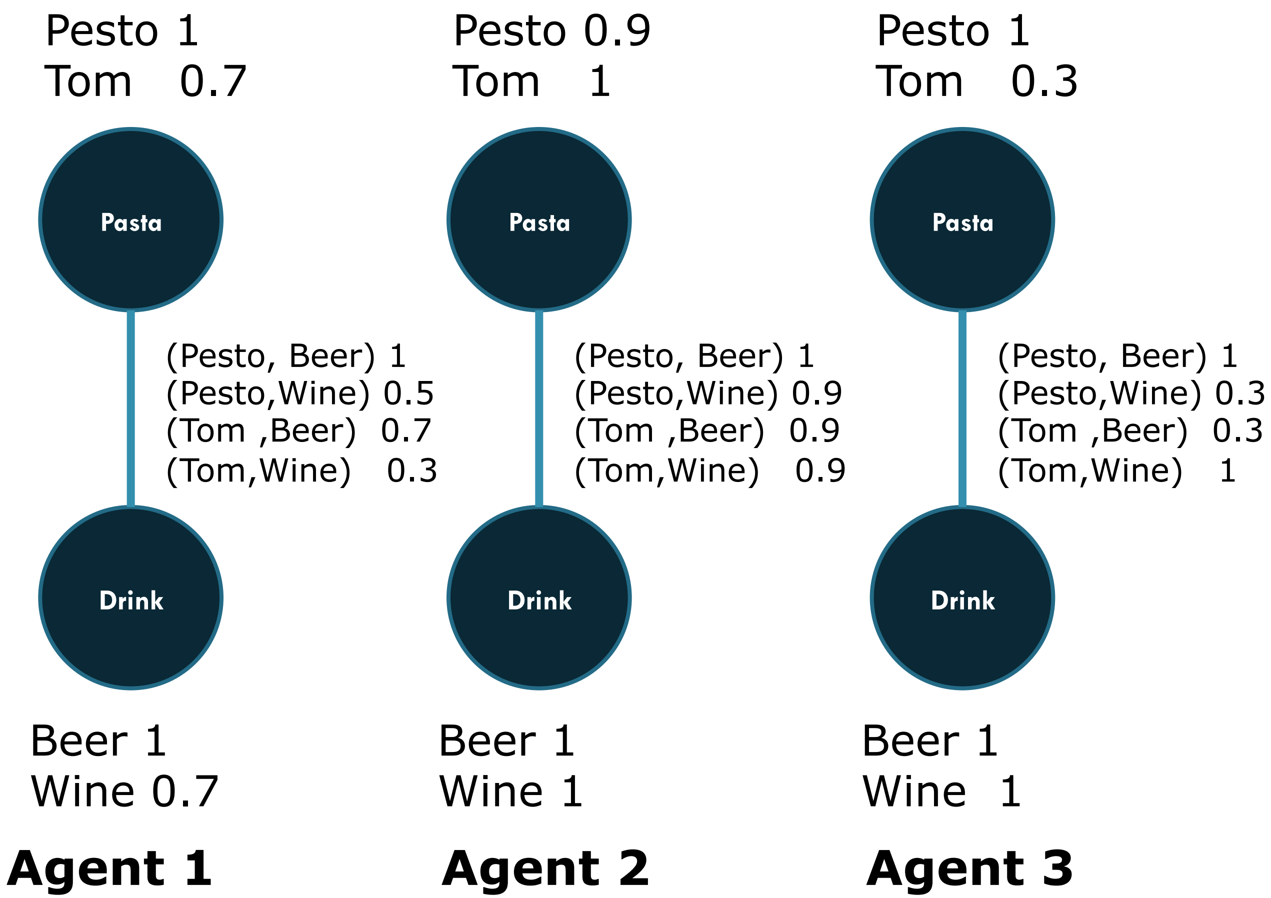
- Sequential voting avoids the problem of electing Condorcet losers
- Thm.: Sequential plurality voting over binary issues never elects a Condorcet loser
  - Proof: Consider the election for the final issue. The winning combination cannot be a CL, since it wins at least against the other combination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

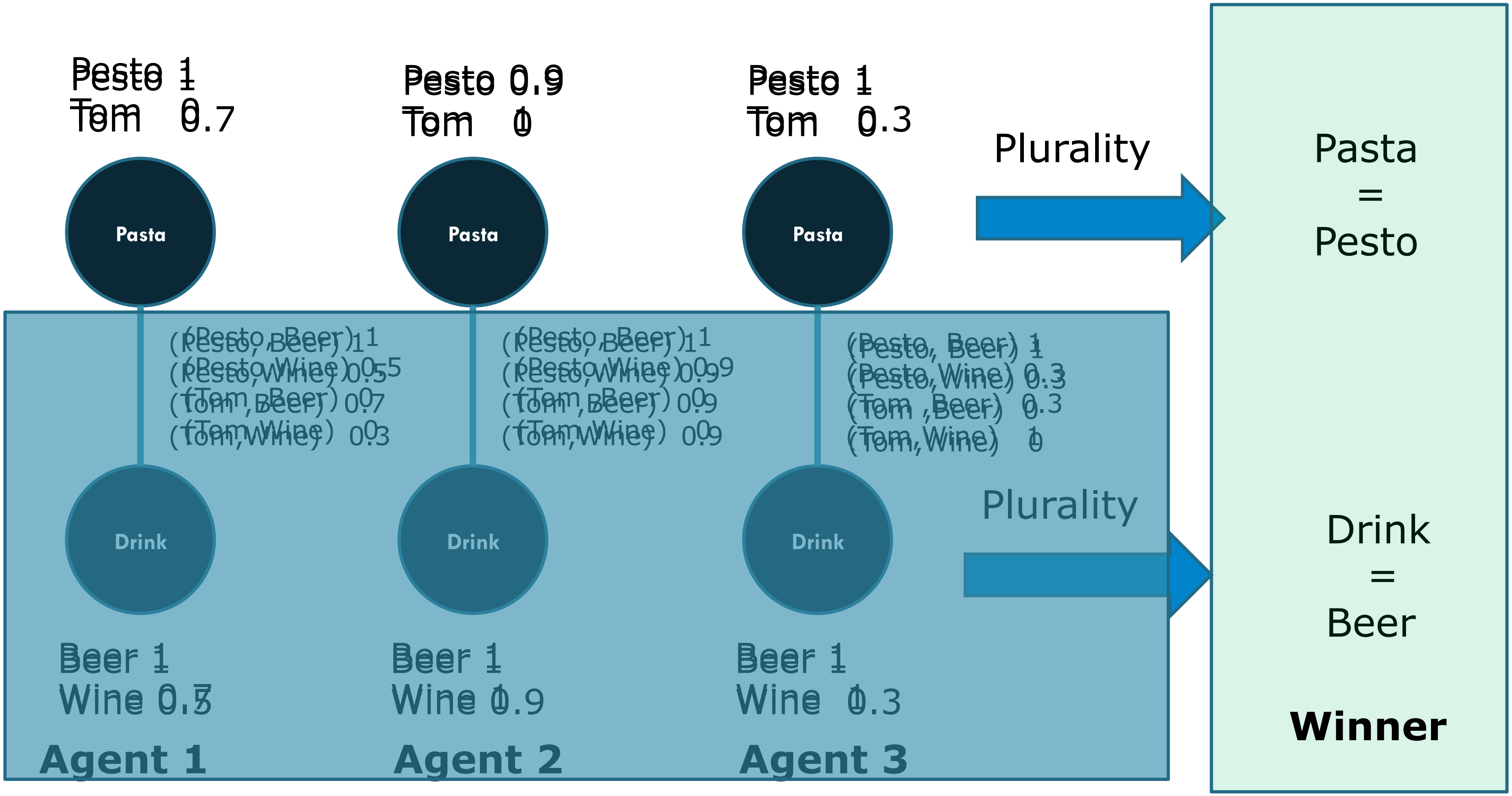
Dinner example, three agents



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



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

	Local to sequential	**Sequential to local**
Condorcet consistency	**no**	yes
Anonymity	yes	yes
Neutrality	**no**	yes
Consistency	yes	yes
Participation	**no**	yes
Efficiency	**yes if single most preferred option for all agents**	yes
Monotonicity	yes	yes
IIA	no	yes
Non-dictatorship	yes	yes
Strategy-proofness	no	yes

[Dalla Pozza, Pini, Rossi, Venable, IJCAI 2011]

Complexity of coalitional constructive manipulation

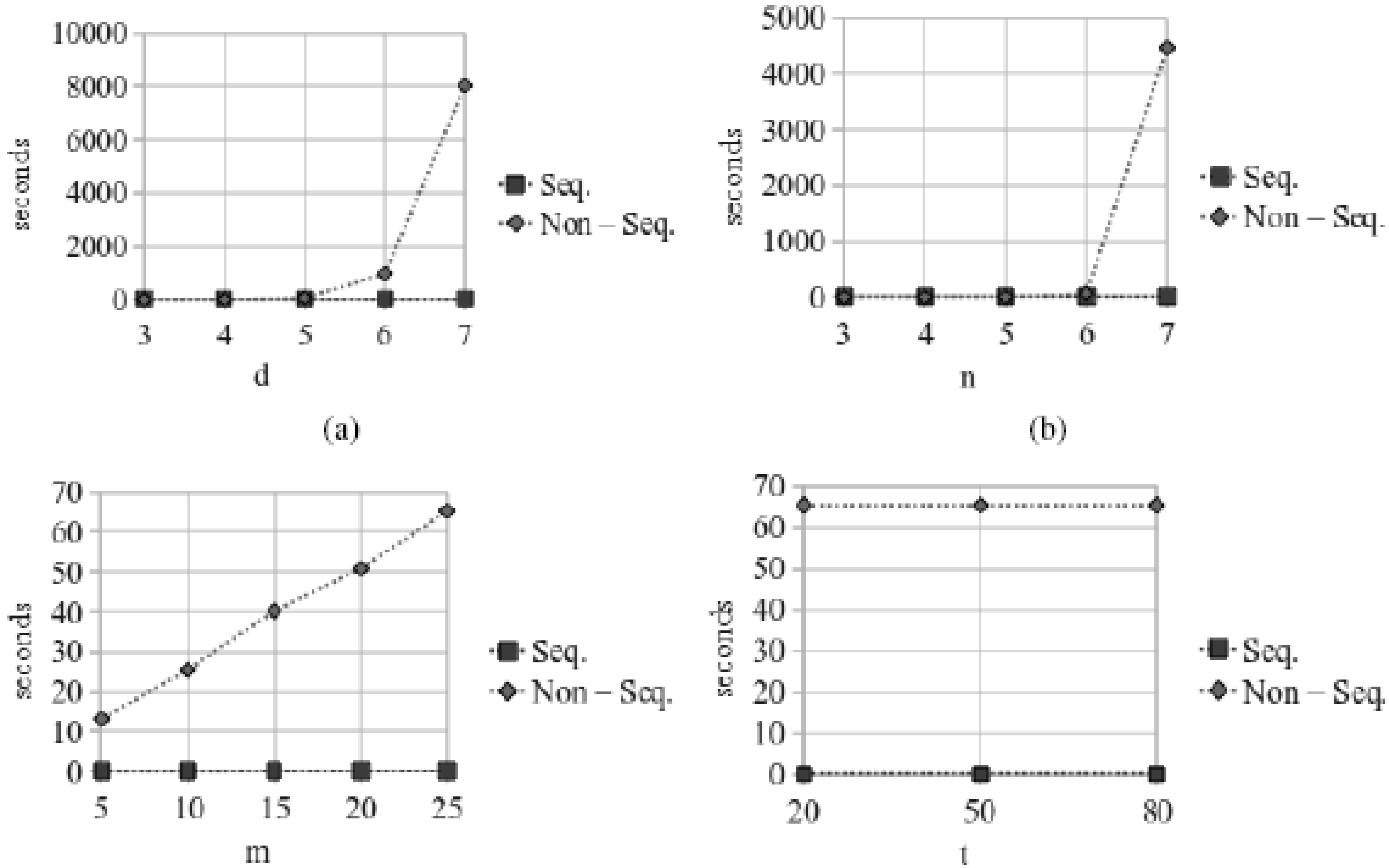
- Constructive Coalitional Manipulation CC(d,C,P,r)
  - Given voting rule r, how difficult it is for coalition of voters C to make candidate d win, knowing the other agents' preferences P?
    - Easy for Copeland with 3 candidates and for Plurality [Conitzer et al., 2007]
    - Difficult for Copeland [Faliszewski et al., 2008]
- Thms:
  - Easy for all local rules → Easy for sequential (if soft constraints are tractable)
  - Hard for one local rule → Hard for the sequential procedure

[Dalla Pozza, Pini, Rossi, Venable, 2011]

Experimental setting

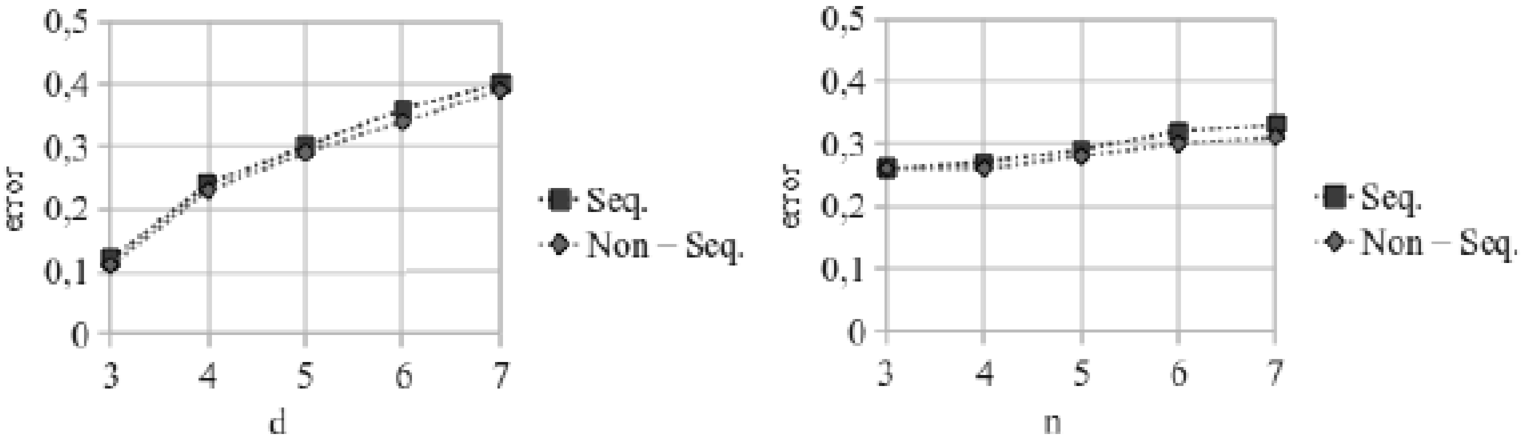
- Randomly generated tree-shaped soft implicit profiles
  - n: number of variables
  - m: number of agents
  - d: domain size
  - t: tightness
- Same rule r for all steps
- Comparison between two voting rules
  - seq(r), from the implicit profile to a solution
  - r, from the explicit profile to a solution
    - 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

Time (Borda)

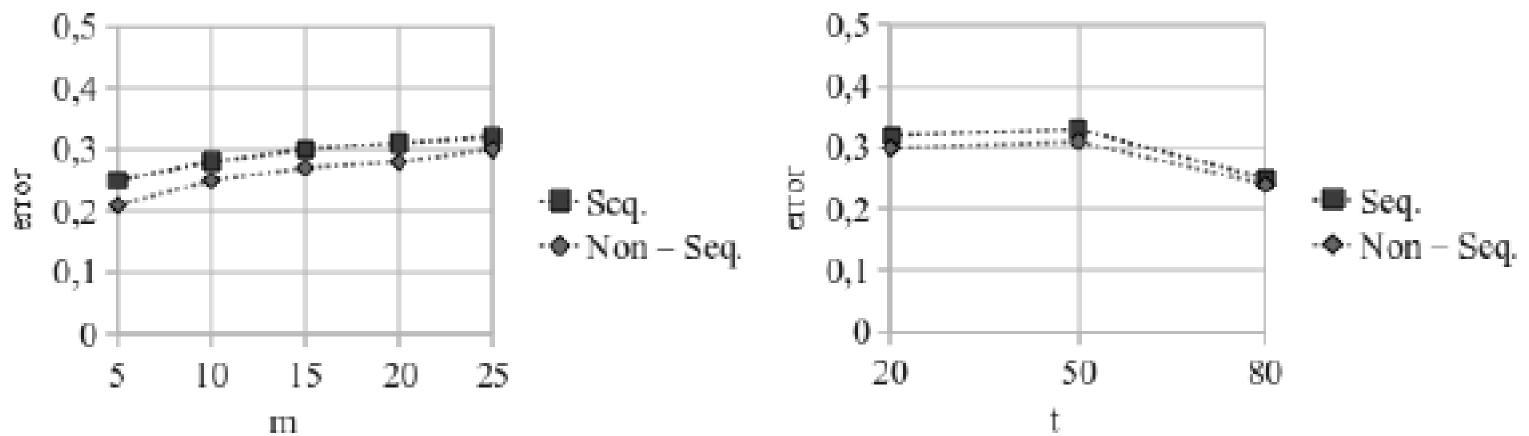


- Sequential rule much faster (no need to compute the explicit profile)

Error (Borda)



(a) (b)



- 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

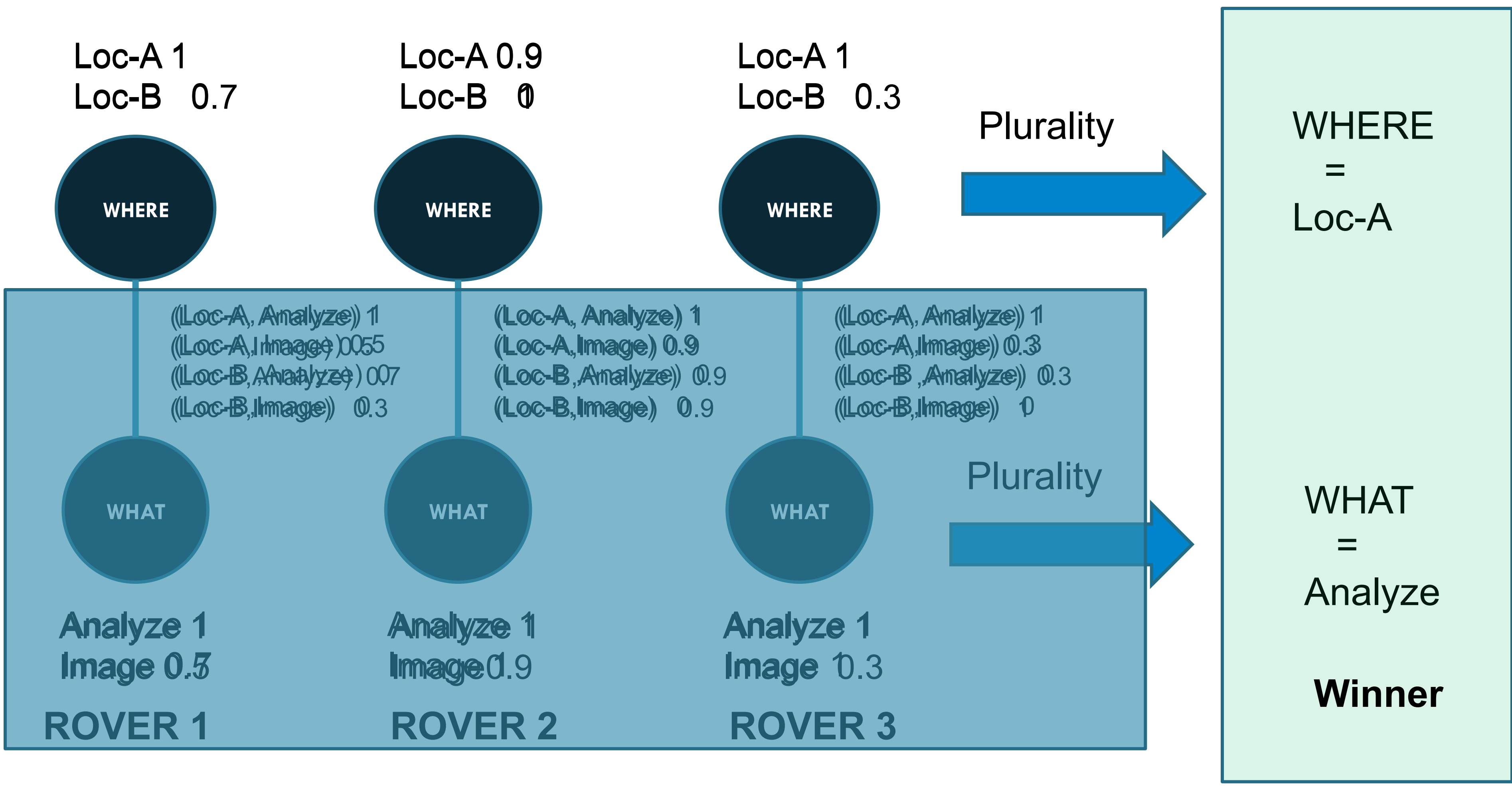
Sequential voting with soft constraints

- Assume agents **vote by giving a soft constraint problem**
- 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
- Proposed solution: **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]

[Dalla Pozza, Pini, Rossi, Venable, ICAART 2011, IJCAI 2011]

Example: 3 rovers must decide where to go and what to do



SEQUENTIAL VOTING WITH CP-NETS

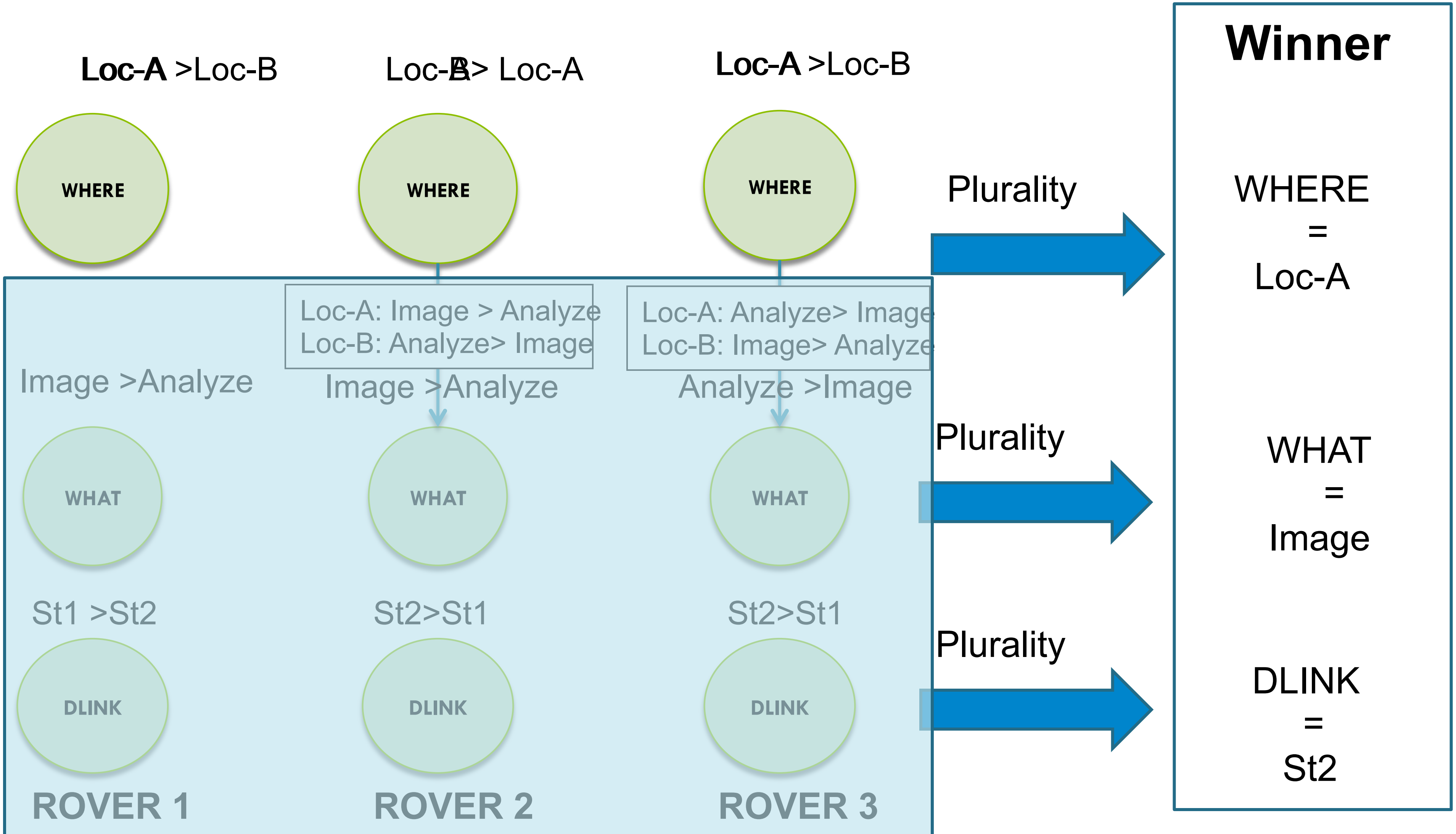
Profiles via compatible CP-nets

- 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



BRIBING CP-NETS

Bribery

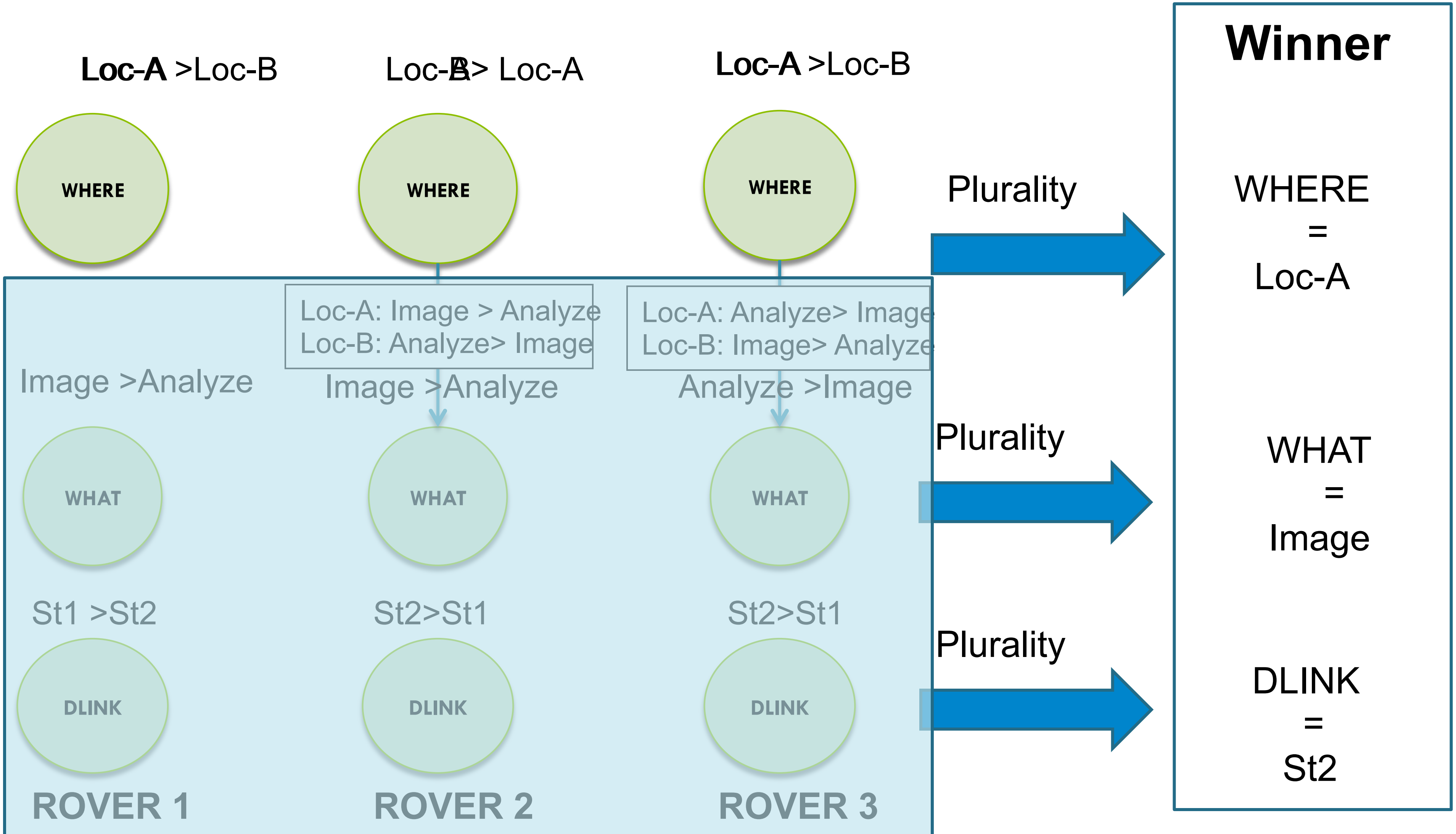
- Given:
  - a voting rule
  - m candidates
  - n voters, voting by giving a CP-net each
  - a cost scheme describing the cost of bribing each voter
  - a candidate p that the briber wants to make the winner
  - the allowed bribery requests
  - a budget B

→ Can the briber make p win by spending at most B?

Example

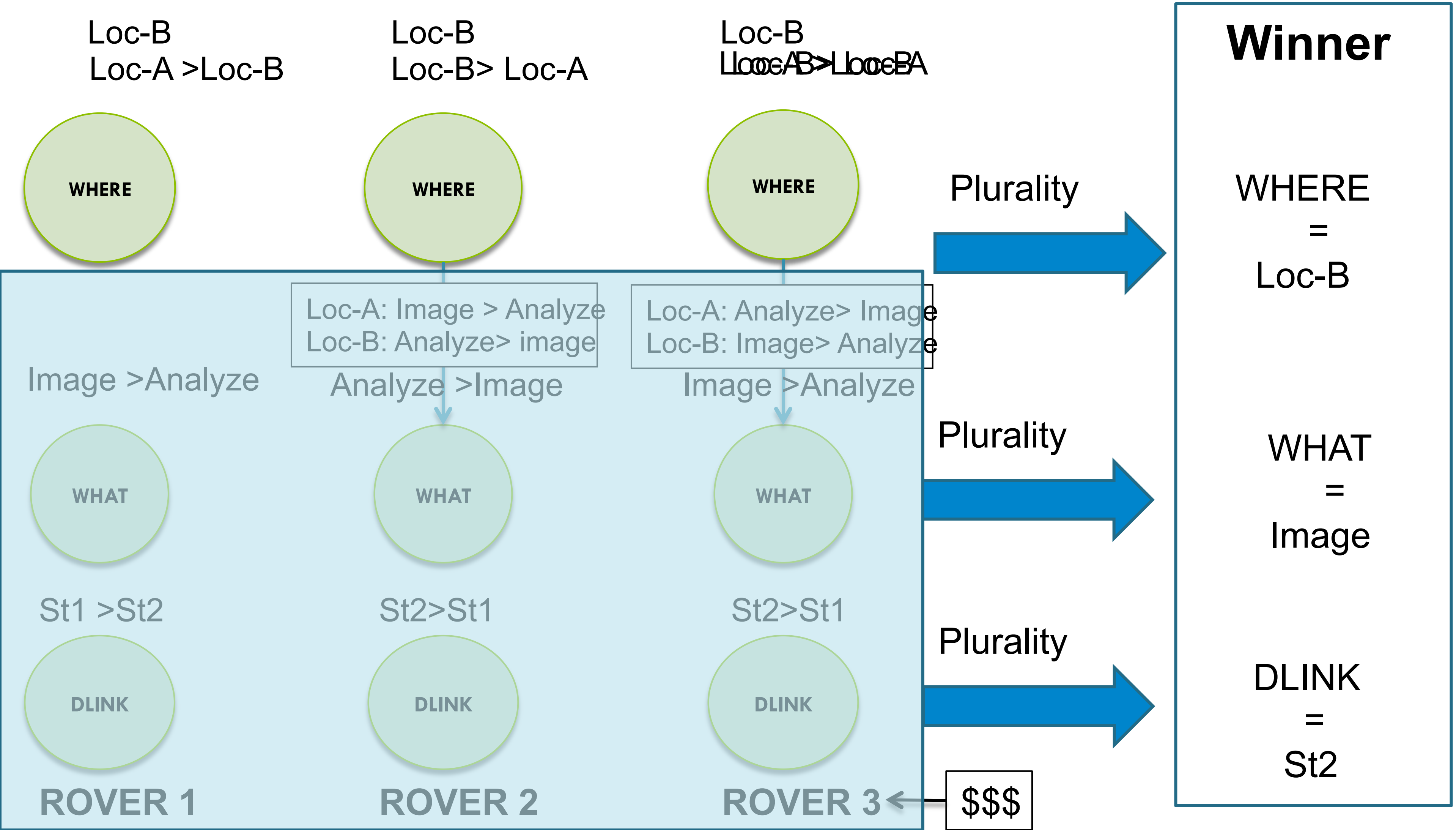
3 Rovers must decide:

- 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



Bribing example

- Voting rule: Sequential Plurality
- P: (loc-B, Image, St2)
- Candidates: all triples
- 3 voters with a CP-net each
- Bribery requests: flips in preference orderings



Bribing cost schemes

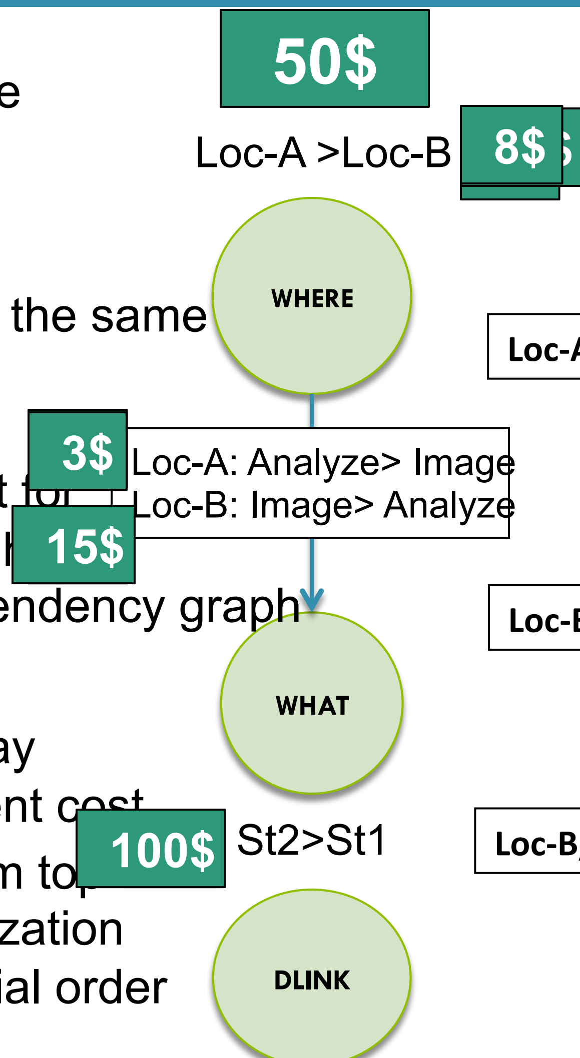
C-equal: same cost for
whatever change
in the CP-nets

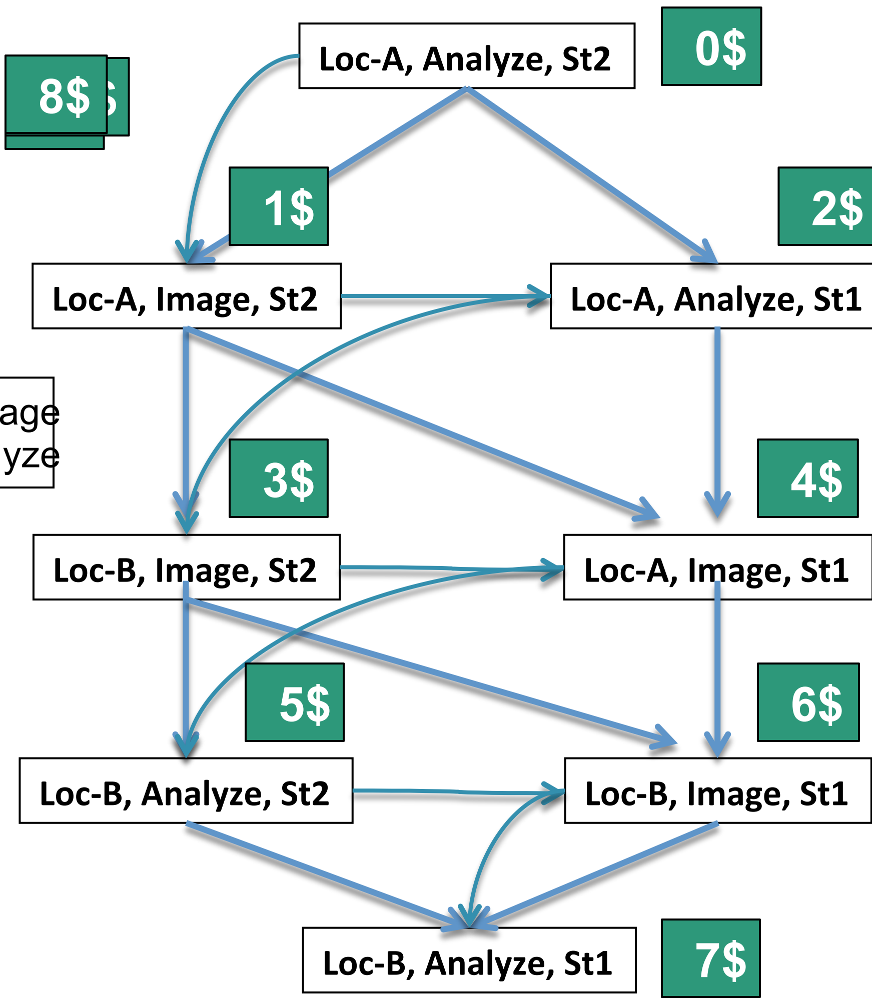
C-flip: cost = n. of flips,
all flips cost the same

C-level: cost = n. of flips,
higher cost for
flipping high
in the dependency graph

C-any: cost = n. of flips,
each flip may
have different cost

C-dist: cost = distance from top
in a linearization
of the partial order





Based on [Brafman, Pilotto, Rossi,
Salvagnin, Venable, Walsh, ADT 2009, KR 2010, AAAI 2011]

Complexity results for bribery with CP-nets

	Sequential Majority	Sequential Majority with weights	Plurality Veto K-Approval (IV)	Plurality Veto K-Approval* (DV, IV+DV)
C_EQUAL	NP-complete	NP-complete	P	P
C_FLIP	P	NP-complete	P	P
C_LEVEL	P	NP-complete	P	?
C_ANY	P	NP-complete	?	?
C_DIST	?	NP-complete	P	P

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 (A1,A2), where A1 comes from the first set and A2 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 roomates
- …

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

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

- Greg proposes to Amy, who accepts ➔ (G,A)
- Harry proposes to Bertha, who accepts ➔ (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 ➔ (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 lies

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

- 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

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

- 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

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 Amy: Harry>Greg
  - Harry: Bertha>Amy 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

- Gale, Shapley. College Admissions and the Stability of Marriage. Amer. Math. Monthly, 69:9-14, 1981
- Roth. The Economics of Matching: Stability and Incentives. Mathematics of Operational Research, 7:617-628, 1982
- Gusfield, Irving. The Stable Marriage Problem: Structure and Algorithms. MIT Press, 1989
- Gale, Sotomayor. Machiavelli and the stable matching problem. Amer. Math. Monthly, 92:261-268, 1985
- Gusfield. Three fast algorithms for four problems in stable marriage. SIAM J. of Computing, 16(1), 1987
- Teo, Sethuraman, Tan. Gale-Shapley Stable Marriage Problem Revisited: Stategic Issues and Applications. Management Science, 47(9): 1252-1267, 2001
- Pini, Rossi, Venable, Walsh. Manipulation and gender neutrality in stable marriage procedures. AAMAS-2009.

Conclusions

- Compact preference modelling
- Comparison of their expressive power and computational properties
- Ability to reason with more than one formalism in the same problem

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