

Graphical Languages for Preference Representation and Applications

Sylvain Bouveret
Onera Toulouse

Jérôme Lang
Université Paris Dauphine

Twenty second International Joint Conference on Artificial Intelligence
Barcelona, July 16–22, 2011



Why preferences in Artificial Intelligence?

- *planning/robotics*: an autonomous agent acts as a proxy for a user
- *individual/collective decision aid*: the system helps the user to make a decision.

Examples: recommender systems, product configuration (and more generally e-commerce).

Specifying a goal: often insufficient

I want a cheap, direct and fast train from Paris to Barcelona

We need to allow for

- flexibility
- partial satisfaction
- trade-off solutions if the goal cannot be reached



Preference structures

$\mathcal{E}$ set of **alternatives** or **outcomes**

Specifying preferences on $\mathcal{E}$: **comparing** / **ranking** alternatives.

Ordinal preferences

Preference relation on $\mathcal{E}$: **reflexive and transitive relation** $\succeq$

$x \succeq y$ x is at least as good as y

$x \succ y \iff x \succeq y$ and not $y \succeq x$
 x is preferred to y (**strict preference**)

$x \sim y \iff x \succeq y$ and $y \succeq x$
 x and y are equally preferred (**indifference**)

Cardinal preferences

Utility function $u : \mathcal{E} \rightarrow \mathbb{R}$ [more generally $u : \mathcal{E} \rightarrow V$ ordered scale]

More sophisticated models: interval orders, fuzzy preferences etc.



Preferences: models and queries



Figure borrowed from (Brafman and Domshlak, 07)



Preferences: models and queries

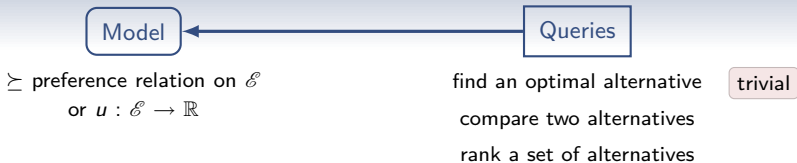


Figure borrowed from (Brafman and Domshlak, 07)



Preferences: models and queries

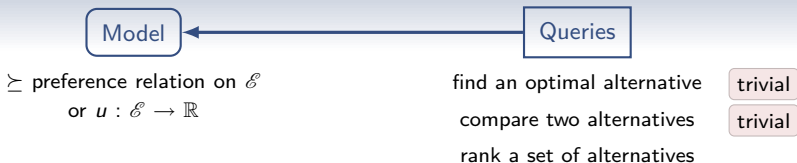


Figure borrowed from (Brafman and Domshlak, 07)



Preferences: models and queries

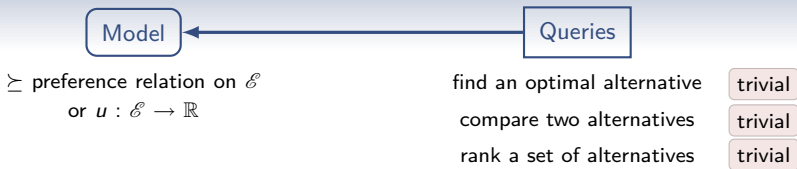


Figure borrowed from (Brafman and Domshlak, 07)



Conclusions

- + good news: everything is easy
- bad news: not much interesting to say



Conclusions

- + good news: everything is easy
- bad news: not much interesting to say

Questions?



Structure of the set of alternatives

A key question in individual and collective decision making: what is the *structure* of the set $\mathcal{E}$ of alternatives?

Sometimes $\mathcal{E}$ has a simple structure (and a small size):

- choose a transportation mode to come to the conference centre:
 $\mathcal{E} = \{\text{metro, walk, bike}\}$
- choose the organizer of the next IJCAI:
 $\mathcal{E} = \{\text{Angela Merkel, Dominique Strauss-Kahn, Jose Luis Zapatero}\}$

But sometimes, $\mathcal{E}$ has a *combinatorial* structure.



Example 1: a configuration problem

Buying a tour package on Internet

- I'd like a return ticket Ithaca – Troy, or else Ithaca – Barcelona.
- if I go to Troy I prefer a palace; in Barcelona, a youth hostel.
- if the trip back is by boat, I'd prefer it with several stopovers.
- if the trip includes a flight, I prefer *Cyclope Airways*.
- if the trip back is by boat I prefer to avoid Charybd and Scylla, except if this saves me at least 50 ϕ .
- my budget is 600 ϕ . (ϕ = Ithaca currency, not convertible).

How can we express such preferences in an efficient way?

How can we find the best k solutions?



Example 2: multiple referenda

2 binary variables:

S (build a swimming pool or not) and T (build a tennis court or not)

Voters' preferences

- **voters 1 and 2** $\rightarrow s\bar{t} \succ \bar{s}t \succ \bar{s}\bar{t} \succ st$
- **voters 3 and 4** $\rightarrow \bar{s}t \succ s\bar{t} \succ \bar{s}\bar{t} \succ st$
- **voter 5** $\rightarrow st \succ s\bar{t} \succ \bar{s}t \succ \bar{s}\bar{t}$

How will voters 1-4 express their preferences on $\{s, \bar{s}\}$ and on $\{t, \bar{t}\}$?



Example 2: multiple referenda

2 binary variables:

S (build a swimming pool or not) and T (build a tennis court or not)

Voters' preferences

- **voters 1 and 2** $\rightarrow s\bar{t} \succ \bar{s}t \succ \bar{s}\bar{t} \succ st$
- **voters 3 and 4** $\rightarrow \bar{s}t \succ s\bar{t} \succ \bar{s}\bar{t} \succ st$
- **voter 5** $\rightarrow st \succ s\bar{t} \succ \bar{s}t \succ \bar{s}\bar{t}$

How will voters 1-4 express their preferences on $\{s, \bar{s}\}$ and on $\{t, \bar{t}\}$?

Suppose they do it in an “optimistic” way :

- voters 1, 2 and 5: s ; voters 3 and 4: $\bar{s} \Rightarrow \text{decision} = s$;
- voters 3, 4 and 5: t ; voters 1 and 2: $\bar{t} \Rightarrow \text{decision} = t$.



Example 2: multiple referenda

2 binary variables:

S (build a swimming pool or not) and T (build a tennis court or not)

Voters' preferences

- **voters 1 and 2** $\rightarrow s\bar{t} \succ \bar{s}t \succ \bar{s}\bar{t} \succ st$
- **voters 3 and 4** $\rightarrow \bar{s}t \succ s\bar{t} \succ \bar{s}\bar{t} \succ st$
- **voter 5** $\rightarrow st \succ s\bar{t} \succ \bar{s}t \succ \bar{s}\bar{t}$

How will voters 1-4 express their preferences on $\{s, \bar{s}\}$ and on $\{t, \bar{t}\}$?

Suppose they do it in an “optimistic” way :

- voters 1, 2 and 5: s ; voters 3 and 4: $\bar{s} \Rightarrow \text{decision} = s$;
- voters 3, 4 and 5: t ; voters 1 and 2: $\bar{t} \Rightarrow \text{decision} = t$.

st is chosen, although it is the worst alternative for all voters but one.

How should such a voting process be handled?



Example 3: combinatorial auctions

$\mathcal{O} = \{o_1, \dots, o_p\}$ set of objects

for each agent i , $V_i : 2^{\mathcal{O}} \rightarrow \mathbb{N}$

$V_i(X)$ = maximum price that i is ready to pay for the set of objects X .

if V_i additive for all i : then sell each object to its highest bidder

but V_i is generally *non-additive* :

- {left shoe}: 10 €; {right shoe}: 10 €; {left shoe, right shoe}: 50 €
- {horchata}: 2 €; {beer}: 3 €; {horchata, beer}: 4 €

optimal allocation π^* : maximizes the seller's revenue $\sum_{i=1}^n V_i(\pi(i))$
where $\pi(i)$ is the set of objects allocated to agent i

How can bidders express their functions V_i ?

How can the seller determine π^* ?



Preferential dependencies

Examples 1 to 3 : the set of alternatives $\mathcal{E}$ has a *combinatorial structure*

$$\mathcal{E} = D_1 \times \dots \times D_n$$

D_i finite domain of values for variable X_i

Existence of *preferential dependencies* between variables:

- if I go to Troy I prefer a palace; in Barcelona, a youth hostel
- if the swimming pool is built then I'd prefer the tennis court not to be built
- if I receive the left shoe then I'm ready to pay more for the right shoe

⇒ makes it hard to decompose a problem into independent subproblems.



Combinatorial spaces...

The combinatorial trap...

Two variables...

$o_1 \succ o_2 \succ o_1 o_2 \succ \emptyset \rightarrow 4$ subsets to compare



Combinatorial spaces...

The combinatorial trap...

Four variables...

$o_1 o_2 \succ o_2 o_3 o_4 \succ o_1 \succ \emptyset \succ o_2 \succ o_1 o_2 o_3 o_4 \succ o_1 o_3 \succ o_2 o_4 \succ o_3 o_4 \succ o_1 o_4 \succ$
 $o_1 o_3 o_4 \succ o_2 o_3 \succ o_4 \succ o_3 \succ o_1 o_2 o_4 \succ o_1 o_2 o_3 \rightarrow 15 \text{ subsets}$



Combinatorial spaces...

The combinatorial trap...

Twenty variables...

$0805 \succ 050309 \succ 08 \succ \emptyset \succ 05 \succ 08050309 \succ 0803 \succ 0509 \succ 0309 \succ 0809 \succ$
 $080309 \succ 0503 \succ 09 \succ 03 \succ 080509 \succ 0805030102050809 \succ 010506 \succ$
 $07 \succ 02030405060708 \succ 0102030405 \succ 0103 \succ 02 \succ 01030709 \succ 0105 \succ$
 $01070809 \succ 02 \succ 04 \succ 06 \succ 0107 \succ 010203 \succ 0102 \succ 020504 \succ 01 \succ$
 $02 \succ 01020504 \succ 0105 \succ 0204 \succ 0504 \succ 0104 \succ 010504 \succ 0205 \succ 04 \succ$
 $05 \succ 010204 \succ 010205 \succ 0105 \succ 050309 \succ 01 \succ \emptyset \succ 05 \succ 01050309 \succ$
 $0103 \succ 0509 \succ 0309 \succ 0109 \succ 010309 \succ 0503 \succ 09 \succ 03 \succ 010509 \succ$
 $0105030906050109 \succ 090506 \succ 07 \succ 06030405060701 \succ 0906030405 \succ$
 $0903 \succ 06 \succ 09030709 \succ 0905 \succ 09070109 \succ 06 \succ 04 \succ 06 \succ 0907 \succ$
 $090603 \succ 0906 \succ 060504 \succ 09 \succ 06 \succ 09060504 \succ 0905 \succ 0604 \succ 0504 \succ$
 $0904 \succ$

→ 1048575 subsets → the expression takes more than 12 days.



The dilemma

- The expression of **preferential dependencies** is often necessary.
- **but** ... Representing and eliciting $\succeq$ or u *in extenso* is unfeasible in practice.



The dilemma

- The expression of **preferential dependencies** is often necessary.
- **but** ... Representing and eliciting $\succeq$ or u *in extenso* is unfeasible in practice.

$\Rightarrow$ *languages for compact preference representation*



Plan

- 1 **Languages for compact preference representation**
- 2 **Logical and bidding languages**
 - Propositional logic
 - Bidding languages
- 3 **Graphical languages for ordinal preferences**
 - An introduction to graphical languages
 - Preferential independence
 - CP-nets
 - Outcome optimization with CP-nets
 - CP-nets: extensions and variants
- 4 **Graphical languages for cardinal preferences**
 - Generalized additivity
- 5 **Collective decision making**
 - Voting
 - MultiAgent Resource Allocation
- 6 **Conclusion**



Outline

- 1 **Languages for compact preference representation**
- 2 **Logical and bidding languages**
 - Propositional logic
 - Bidding languages
- 3 **Graphical languages for ordinal preferences**
 - An introduction to graphical languages
 - Preferential independence
 - CP-nets
 - Outcome optimization with CP-nets
 - CP-nets: extensions and variants
- 4 **Graphical languages for cardinal preferences**
 - Generalized additivity
- 5 **Collective decision making**
 - Voting
 - MultiAgent Resource Allocation
- 6 **Conclusion**



Preferences, languages

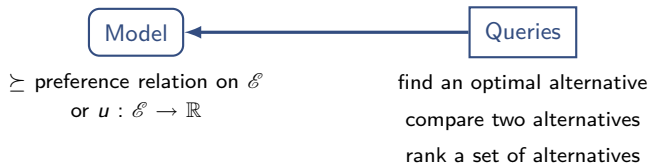


Figure borrowed from (Brafman and Domshlak, 07)



Preferences, languages

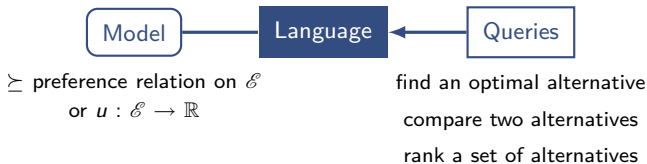


Figure borrowed from (Brafman and Domshlak, 07)



Preferences, languages

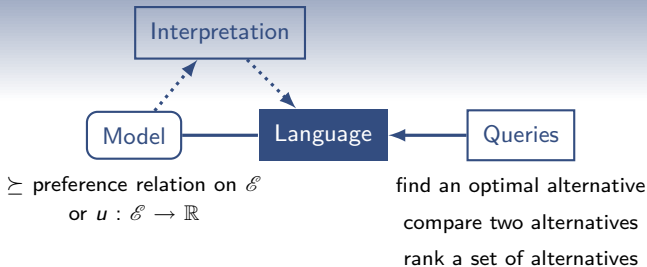


Figure borrowed from (Brafman and Domshlak, 07)



Representation languages

Representation language : $\langle L, I_L \rangle$, where

- L language
- $I_L : \Phi \in L \mapsto$ preference relation $\succeq_\Phi$ or utility function u_Φ induced by Φ

A prototypical language: plain propositional logic

- L : set of all propositional formulas built on a finite set of propositional symbols PS .
- $I_L : \varphi \mapsto u_\varphi$ defined by

$$\text{for all } \omega \in 2^{PS}, u_\varphi(\omega) = \left\{ \begin{array}{ll} +1 & \text{if } \omega \models \varphi \\ 0 & \text{if } \omega \models \neg\varphi \end{array} \right\}$$

or equivalently

- $I_L : \varphi \mapsto \succeq_\varphi$ defined by

$$\text{for all } \omega, \omega' \in 2^{PS}, \omega \succeq_\varphi \omega' \text{ iff } \omega \models \varphi \text{ or } \omega' \models \neg\varphi$$



Representation languages

Representation language : $\langle L, I_L \rangle$, where

- L language
- $I_L : \Phi \in L \mapsto$ preference relation $\succeq_\Phi$ or utility function u_Φ induced by Φ

Another prototypical language: the “explicit” representation.

- L : set of all weak orders on $D = D_1 \times \dots \times D_n$
- I_L : identity

(and similarly for utility functions)



Representation languages

On which criteria can we evaluate the different languages?

- **Expressive power:** what is the set of all preference structures expressible in the language?

Example: expressivity of plain propositional logic = set of all *dichotomous* preference relations



Representation languages

On which criteria can we evaluate the different languages?

- **Expressive power:** what is the set of all preference structures expressible in the language?

Example: expressivity of plain propositional logic = set of all *dichotomous* preference relations

- **Succinctness:** (informally) language L_1 is at least as succinct as language L_2 is any preference structure expressible in L_2 can be expressed in L_1 *without any exponential growth of size*.



Representation languages

On which criteria can we evaluate the different languages?

- **Expressive power:** what is the set of all preference structures expressible in the language?

Example: expressivity of plain propositional logic = set of all *dichotomous* preference relations

- **Succinctness:** (informally) language L_1 is at least as succinct as language L_2 is any preference structure expressible in L_2 can be expressed in L_1 *without any exponential growth of size*.
- **Computational complexity:** how hard is it to compare two alternatives or to find an optimal alternative when the preference structure is represented in L ?



Representation languages

On which criteria can we evaluate the different languages?

- **Expressive power:** what is the set of all preference structures expressible in the language?

Example: expressivity of plain propositional logic = set of all *dichotomous* preference relations

- **Succinctness:** (informally) language L_1 is at least as succinct as language L_2 is any preference structure expressible in L_2 can be expressed in L_1 *without any exponential growth of size*.
- **Computational complexity:** how hard is it to compare two alternatives or to find an optimal alternative when the preference structure is represented in L ?
- **Easiness of elicitation**

Preference elicitation = interaction with a user, so as to acquire her preferences, encoded in a language L .

Is it easy to construct protocols for eliciting the agent's preferences in L ?



More on expressive power

Representation language: $\langle L, I_L \rangle$

Expressive power of a language = set of all preference structures that can be expressed in the language $= I_L(L)$.

$\langle L, I_L \rangle$ at least as expressive as $\langle L', I_{L'} \rangle$ iff $I_L(L) \supseteq I_{L'}(L')$.

Examples :

- expressive power of plain propositional logic: all dichotomous utility functions (or all dichotomous preference relations)
- expressive power of explicit representation: all preference relations / utility functions

The explicit representation is more expressive than plain propositional logic.



More on succinctness

Relative notion:

L_1 is at least as succinct as L_2 if there exists $F : L_2 \rightarrow L_1$ and a polynomial function p such that for all $\Phi \in L_2$:

- $I_{L_2}(\Phi) = I_{L_1}(F(\Phi))$: Φ and $F(\Phi)$ induce the same preferences
- $|F(\Phi)| \leq p(|\Phi|)$: the translation is succinct

Example:

- L_1 = plain propositional logic
- L_2 = explicit representation restricted to dichotomous preferences:
 $L_2 = \{X \mid X \subseteq 2^{P^S}\}$
- L_3 = explicit representation language (non restricted to dichotomous preferences)

L_1 is strictly more succinct than L_2 ; but L_1 and L_3 are incomparable.



More on computational complexity

What is the computational complexity of the following problems *when the preferences on D are represented in the language L* :

Given an input Φ in the language L , ...

- DOMINANCE: and $\vec{x}, \vec{y} \in D$, do we have $\vec{x} \succeq_{\Phi} \vec{y}$?
- OPTIMISATION: find the preferred option (or one of the preferred options) $\vec{x}$;
- CONSTRAINED OPTIMISATION: and a subset C , possibly defined succinctly as well succinctly, find the preferred option (or one of the preferred options) $\vec{x} \in C$.

Examples:

- plain propositional logic: DOMINANCE in P , OPTIMISATION (constrained or not) NP-hard
- explicit representation: all problems in P (provided that C is also represented explicitly)



More on elicitation

Preference elicitation = interaction with a user, so as to acquire her preferences, encoded in a language L (or more generally, so as to acquire enough information about her preferences for suggesting her an optimal decision).

Construction of elicitation protocols for some families of languages:

- exploiting preferential independencies so as to reduce the amount of information to elicit and the cognitive effort spent in communication;
- trade-off expressivity vs. elicitation complexity.



Back to preferential dependencies

Expressing preferential dependencies: classes of languages

First dichotomy:

- **Graphical languages:**

- **graphical component** describing preferential dependencies between variables
- collection of **local preferences** on single variables or small subsets of variables, compatible with the dependence structure.

- **Logical languages:**

use logic to express preferential dependencies

Second dichotomy:

- languages for expressing **ordinal preferences**
- languages for expressing **cardinal preferences**



Back to preferential dependencies

Expressing preferential dependencies: classes of languages

First dichotomy:

- **Logical languages:**
use logic to express preferential dependencies
- **Graphical languages:**
 - **graphical component** describing preferential dependencies between variables
 - collection of **local preferences** on single variables or small subsets of variables, compatible with the dependence structure.
- **Bidding languages:**
a special kind of languages designed for auctions

Second dichotomy:

- languages for expressing **ordinal preferences**
- languages for expressing **cardinal preferences**

In this talk:

- a quick glimpse on logical and bidding languages
- a special focus on graphical languages: CP-nets, GAI-nets and extensions



Outline

- 1 **Languages for compact preference representation**
- 2 **Logical and bidding languages**
 - Propositional logic
 - Bidding languages
- 3 **Graphical languages for ordinal preferences**
 - An introduction to graphical languages
 - Preferential independence
 - CP-nets
 - Outcome optimization with CP-nets
 - CP-nets: extensions and variants
- 4 **Graphical languages for cardinal preferences**
 - Generalized additivity
- 5 **Collective decision making**
 - Voting
 - MultiAgent Resource Allocation
- 6 **Conclusion**



Of logic and goals

Logic-based languages suit well when we have to deal with **binary variables** (e.g. resource allocation problems).

- A propositional syntax $L_{\mathcal{O}} \dots$
 - set of propositional symbols $\mathcal{O}$,
 - usual connectives $\neg, \wedge, \vee$



Of logic and goals

Logic-based languages suit well when we have to deal with **binary variables** (e.g. resource allocation problems).

- A propositional syntax $L_{\mathcal{O}} \dots$
 - set of propositional symbols $\mathcal{O}$,
 - usual connectives $\neg, \wedge, \vee$

Example

- $\mathcal{O} = \{ \text{phone}, \text{notebook}, \text{camera}, \text{monitor}, \text{printer}, \text{DVD} \}$.
- Set of requests for one agent:
 - $\text{DVD} \wedge ((\text{printer} \wedge \text{monitor}) \vee \text{notebook})$,
 - $\text{camera} \wedge \text{printer}$.



Dichotomous preferences...

What to do with all these goals ?



Dichotomous preferences...

What to do with all these goals ?

A first (simplistic) example

Example

Variables $\mathcal{O} = \{o_1, o_2, o_3\}$

- $o_2 \wedge \neg o_1$
 $\{o_2\}, \{o_2, o_3\} \succ \emptyset, \{o_1\}, \{o_3\}, \{o_1, o_2\}, \{o_1, o_3\}, \{o_1, o_2, o_3\}$
- $(o_1 \wedge o_2 \wedge \neg o_3) \vee (\neg o_1 \wedge o_2 \wedge o_3)$
 $\{o_1, o_2\}, \{o_2, o_3\} \succ \emptyset, \{o_1\}, \{o_2\}, \{o_3\}, \{o_1, o_3\}, \{o_1, o_2, o_3\}$



A bit more interesting...

- Goal bases (e.g. $\{\varphi_1, \dots, \varphi_m\}$):
 - cardinality
 - inclusion
 - distances
- **Prioritized** goal bases (e.g. $\{\langle \varphi_1, k_1 \rangle, \dots, \langle \varphi_m, k_m \rangle\}$):
 - discrimin
 - leximin
 - best-out
- **Weighted** goal bases



A language based on weighted goals

Preference representation:

- A propositional language $L_{\mathcal{O}}$...
 - a set of propositional symbols $\mathcal{O}$,
 - the usual connectives $\neg, \wedge, \vee$
- ...and some weights $w \in \mathcal{V}$.



A language based on weighted goals

Preference representation:

- A propositional language $L_{\mathcal{O}}$...
 - a set of propositional symbols $\mathcal{O}$,
 - the usual connectives $\neg, \wedge, \vee$
- ...and some weights $w \in \mathcal{V}$.

Example

- $\mathcal{O} = \{ \text{phone}, \text{laptop}, \text{camera}, \text{monitor}, \text{printer}, \text{DVD} \}$.
- Agent 1's requests:
 - $\left\langle \text{DVD} \wedge \left((\text{printer} \wedge \text{monitor}) \vee \text{laptop} \right), 110 \right\rangle,$
 - $\left\langle \text{DVD}, -10 \right\rangle,$
 - $\left\langle \text{camera} \wedge \text{printer}, 50 \right\rangle.$



Individual utility

Expresses the satisfaction of an agent regarding an allocation. Depends on:

- her share (assumption of non exogenous preferences),
- her weighted requests,

and is obtained by **aggregating** the weights of the satisfied formulas, using an operator $\oplus$.

Individual utility

Given an agent i , her requests Δ_i , an allocation $\vec{\pi}$, her individual utility is:

$$u_i(\pi_i) = \bigoplus \{w \mid \langle \varphi, w \rangle \in \Delta_i \text{ and } x_i \models \varphi\}.$$

Two reasonable choices for $\oplus$: $+$ or $\max$.



Individual utility

Example

- $\theta = \{ \text{phone}, \text{laptop}, \text{camera}, \text{monitor}, \text{printer}, \text{DVD} \}.$

- Agent 1's requests:

- $\left\langle \text{DVD} \wedge \left((\text{printer} \wedge \text{monitor}) \vee \text{laptop} \right), 110 \right\rangle,$

- $\left\langle \text{DVD}, -10 \right\rangle,$

- $\left\langle \text{camera} \wedge \text{printer}, 50 \right\rangle.$

Computation of individual utility ($\oplus = +$) :

$$\pi_1 = \{ \text{DVD}, \text{printer}, \text{laptop}, \text{printer} \}$$



Individual utility

Example

- $\theta = \{ \text{phone}, \text{laptop}, \text{camera}, \text{monitor}, \text{printer}, \text{DVD} \}.$

- Agent 1's requests:

- $\left\langle \text{DVD} \wedge ((\text{printer} \wedge \text{monitor}) \vee \text{laptop}), 110 \right\rangle,$

- $\left\langle \text{DVD}, -10 \right\rangle,$

- $\left\langle \text{camera} \wedge \text{printer}, 50 \right\rangle.$

Computation of individual utility ($\oplus = +$) :

$$\pi_1 = \{ \text{DVD}, \text{printer}, \text{laptop}, \text{printer} \} \Rightarrow u_1(\pi_1) = \text{DVD} \wedge ((\text{printer} \wedge \text{monitor}) \vee \text{laptop})$$

110



Individual utility

Example

- $\theta = \{ \text{phone}, \text{camera}, \text{monitor}, \text{printer}, \text{DVD} \}$.

- Agent 1's requests:

- $\langle \text{DVD} \wedge ((\text{camera} \wedge \text{monitor}) \vee \text{printer}), 110 \rangle,$

- $\langle \text{DVD}, -10 \rangle,$

- $\langle \text{camera} \wedge \text{printer}, 50 \rangle.$

Computation of individual utility ($\oplus = +$) :

$$\pi_1 = \{ \text{DVD}, \text{camera}, \text{monitor}, \text{printer} \} \Rightarrow u_1(\pi_1) = 110 \overset{\text{DVD}}{-10}$$



Individual utility

Example

- $\theta = \{ \text{phone}, \text{laptop}, \text{camera}, \text{monitor}, \text{printer}, \text{DVD} \}.$

- Agent 1's requests:

- $\left\langle \text{DVD} \wedge ((\text{printer} \wedge \text{monitor}) \vee \text{laptop}), 110 \right\rangle,$

- $\left\langle \text{DVD}, -10 \right\rangle,$

- $\left\langle \text{camera} \wedge \text{printer}, 50 \right\rangle.$

Computation of individual utility ($\oplus = +$) :

$$\pi_1 = \{ \text{DVD}, \text{printer}, \text{laptop}, \text{printer} \} \Rightarrow u_1(\pi_1) = 110 - 10 + \cancel{0}$$



Individual utility

Example

• $\theta = \{ \text{phone}, \text{laptop}, \text{camera}, \text{monitor}, \text{printer}, \text{DVD} \}.$

• Agent 1's requests:

• $\left\langle \text{DVD} \wedge \left((\text{printer} \wedge \text{monitor}) \vee \text{laptop} \right), 110 \right\rangle,$

• $\left\langle \text{DVD}, -10 \right\rangle,$

• $\left\langle \text{camera} \wedge \text{printer}, 50 \right\rangle.$

Computation of individual utility ($\oplus = +$) :

$$\pi_1 = \{ \text{DVD}, \text{printer}, \text{laptop} \} \Rightarrow u_1(\pi_1) = 110 - 10 + 0 = 100$$



Weighted goals: expressive power

- **Unrestricted language:** fully expressive.
- **Positive cubes only:** fully expressive (see Möbius transform).
- **Positive cubes and positive weights only:** only monotonic functions.
- **Literals only:** additive functions

See [Uckelman, 2008] for more details



Uckelman, J. (2008).

More Than the Sum of Its Parts. Compact Preference Representation of Combinatorial Domains.
PhD thesis, Universiteit van Amsterdam.



Weighted logic: succinctness

- **Unrestricted language:** fully expressive.
- **Positive cubes only:** fully expressive.



Weighted logic: succinctness

- **Unrestricted language:** fully expressive.
- **Positive cubes only:** fully expressive.

But... Unrestricted language strictly more succinct than the restriction to positive cubes only.

Example

u defined as $\langle o_1 \vee o_2 \vee \dots \vee o_n, 1 \rangle$.



Dominance and optimization

- [DOMINANCE] can be solved in **polynomial time**.
- [OPTIMIZATION] is **NP-complete** in the general case.
- [OPTIMIZATION] is **polynomial** for the monotonic fragment of weighted logic (no negation, positive weights).



Auctions

- Numerical preferences: **money**.
- **Usual criterion**: maximize the revenue of the auctioneer.

Classical auctions:

- Additive preferences.
- Each object is sold to the bidder proposing the highest price.



From classical to combinatorial auctions

Example from [Uckelman, 2008].

An auction with two shoes: {left, right}

- The first bidder has the following utility function: $u(lr) = €40$, $u(\pi) = 0$ otherwise.
- The second bidder has an additive utility function : $u = 10l + 10r$.

How should the bidders bid in a traditional (sequential) auction ?



Uckelman, J. (2008).

More Than the Sum of Its Parts. Compact Preference Representation of Combinatorial Domains.
PhD thesis, Universiteit van Amsterdam.



Bidding languages

Bidding languages are preference representation languages developed specifically for combinatorial auctions, to allow bidders to transmit their valuations to the auctioneer.

- Bids are combinations of atomic bids $b = \langle \pi, w \rangle$: bundle/price. The valuation of a bundle π' is w if $\pi \subseteq \pi'$, and 0 otherwise.
- In the XOR-language, atomic bids by the same bidder are assumed to be mutually exclusive:

$$(b_1 \text{ XOR } b_2)(\pi) = \max\{b_1(\pi), b_2(\pi)\}$$

- In the OR-language, the valuation is taken to be the maximal value that can be obtained by accepting disjoint bids

$$(b_1 \text{ OR } b_2)(\pi) = \max_{\pi' \subseteq \pi} \{b_1(\pi'), b_2(\pi \setminus \pi')\}$$



Examples

XOR language: $(\pi_1, w_1) \text{ XOR } \dots \text{ XOR } (\pi_n, w_n)$

$(\{\text{DVD}, \text{hard drive}, \text{printer}\}, 400\text{€}) \text{ XOR } (\{\text{camera}, \text{printer}\}, 700\text{€}) \text{ XOR } (\{\text{phone}\}, 100\text{€})$

$\{\text{DVD}, \text{hard drive}, \text{printer}, \text{camera}, \text{phone}\} \Rightarrow u =$



Examples

XOR language: $(\pi_1, w_1) \text{ XOR } \dots \text{ XOR } (\pi_n, w_n)$

$(\{\text{DVD}, \text{hard drive}, \text{printer}\}, 400\text{€}) \text{ XOR } (\{\text{camera}, \text{printer}\}, 700\text{€}) \text{ XOR } (\{\text{phone}\}, 100\text{€})$

$\{\text{DVD}, \text{hard drive}, \text{printer}, \text{camera}, \text{phone}\} \Rightarrow u = 700\text{€}$



Examples

OR language: $(\pi_1, w_1) \text{ OR } \dots \text{ OR } (\pi_n, w_n)$

$(\{\text{DVD}, \text{hard drive}, \text{printer}\}, 400\text{€}) \text{ OR } (\{\text{camera}, \text{printer}\}, 700\text{€}) \text{ OR } (\{\text{phone}\}, 100\text{€})$

$\{\text{DVD}, \text{hard drive}, \text{printer}, \text{camera}, \text{phone}\} \Rightarrow u =$



Examples

OR language: $(\pi_1, w_1) \text{ OR } \dots \text{ OR } (\pi_n, w_n)$

$(\{\text{DVD}, \text{hard drive}, \text{printer}\}, 400\text{€}) \text{ OR } (\{\text{camera}, \text{printer}\}, 700\text{€}) \text{ OR } (\{\text{phone}\}, 100\text{€})$

$\{\text{DVD}, \text{hard drive}, \text{printer}, \text{camera}, \text{phone}\} \Rightarrow u = 800\text{€}$



Examples

Other languages

- Combinations of OR and XOR (OR-of-XOR / XOR-of-OR...)
- The **OR^{*}-language** is like the OR-language, but dummy items can be used to express exclusiveness constraints. Example:

$\langle \{a, dummy\}, 3 \rangle$ OR $\langle \{b, dummy\}, 3 \rangle$ OR $\langle \{a, b, dummy\}, 5 \rangle$.

- Weighted logic.



The WDP

Winner Determination Problem

- **Input** : A set of bids (XOR, OR, or other...)
- **Solution** : An allocation that maximizes the sum of utilities (*i.e* the auctioneer's payment).

Example

- **Agent 1** : ($\{\text{DVD}, \text{Laptop}, \text{Printer}\}, 400\text{€}$)
- **Agent 2** : ($\{\text{Camera}, \text{Printer}\}, 700\text{€}$) *OR* ($\{\text{Phone}\}, 100\text{€}$)



The WDP

Winner Determination Problem

- **Input** : A set of bids (XOR, OR, or other...)
- **Solution** : An allocation that maximizes the sum of utilities (*i.e* the auctioneer's payment).

Example

- **Agent 1** : $(\{\text{DVD}, \text{Laptop}, \text{Printer}\}, 400\text{€})$
- **Agent 2** : $(\{\text{Camera}, \text{Printer}\}, 700\text{€})$ OR $(\{\text{Phone}\}, 100\text{€})$

Solution :

- **Agent 1** : Nothing
- **Agent 2** : $\{\text{Camera}, \text{Printer}, \text{Phone}\} \rightarrow u = 800 \text{ €}.$



Bidding language: features

- **Expressive power:**

- XOR can represent all (monotonic) valuations.
- OR can only represent superadditive valuations (why ?).

- **Succinctness:**

- OR is strictly more succinct than XOR.



Bidding language: features

- **Expressive power:**

- XOR can represent all (monotonic) valuations.
- OR can only represent superadditive valuations (why ?).

- **Succinctness:**

- OR is strictly more succinct than XOR.

Proof sketch:

- Every superadditive function has the same representation in XOR and OR.
- Function $u(\pi) = |\pi|$ needs an exponential XOR formula (whereas it only needs a linear size OR formula).



Bidding language: Complexity

- Compute the utility of a given XOR bundle: **polynomial**.
- Compute the utility of a given OR bundle: **NP-complete** !
- The WDP is also NP-complete for most languages.



Bidding language: Complexity

- Compute the utility of a given XOR bundle: **polynomial**.
- Compute the utility of a given OR bundle: **NP-complete** !
- The WDP is also NP-complete for most languages.

Proof sketch

- Membership to NP is easy.
- Hardness by reduction from [SET PACKING]

[Set Packing]

- **Input:** Collection $\mathcal{C}$ of finite sets and $K \in \mathbb{N}$.
- **Question:** Is there a collection of disjoint sets $\mathcal{C}' \subseteq \mathcal{C}$ s.t. $|\mathcal{C}'| > K$?



Outline

1 Languages for compact preference representation

2 Logical and bidding languages

Propositional logic

Bidding languages

3 Graphical languages for ordinal preferences

An introduction to graphical languages

Preferential independence

CP-nets

Outcome optimization with CP-nets

CP-nets: extensions and variants

4 Graphical languages for cardinal preferences

Generalized additivity

5 Collective decision making

Voting

MultiAgent Resource Allocation

6 Conclusion



A short introduction

[Probabilistic] graphical models are a marriage between probability theory and graph theory. They provide a natural tool for dealing with two problems that occur throughout applied mathematics and engineering – uncertainty and complexity.

The Economist, 03/22/2001



A short introduction

[Probabilistic] graphical models are a marriage between probability theory and graph theory. They provide a natural tool for dealing with two problems that occur throughout applied mathematics and engineering – uncertainty and complexity.

The Economist, 03/22/2001

More generally, a natural tool for dealing with complexity and:

- **uncertainty**: Bayesian networks, Markov random fields [Pearl, 1988]
- **(un)satisfaction** → constraint networks [Montanari, 1974]
- **costs** → value constraint networks (cost function networks) [Schiex et al., 1995]
- **ordinal preferences** → CP-nets... [Boutilier et al., 2004]
- **utilities** → Multi-attribute utility [Keeney and Raiffa, 1976], k -additivity [Grabisch, 1997], GAI-nets [Bacchus and Grove, 1995]...



A short introduction

Graphical models / languages are based upon:

- a **graphical component** describing (un)directed dependencies between variables
- collection of **local ordinal or numerical statements** on single variables or small subsets of variables, compatible with the dependence structure

Why graphs ?

- A very natural and intuitive way of representing interactions between variables
- Efficient algorithms



Preferential independence

Conditional preferential independence (Keeney & Raiffa, 76):

$\mathcal{V}$ set of variables; $\{\mathcal{X}, \mathcal{Y}, \mathcal{Z}\}$ partition of $\mathcal{V}$.

$D_{\mathcal{X}} = \times_{X_i \in \mathcal{X}} D_i$ etc.

$\mathcal{X}$ is preferentially independent from $\mathcal{Y}$ (given $\mathcal{Z}$) iff

$$\begin{aligned} &\text{for all } x, x' \in D_{\mathcal{X}}, y, y' \in D_{\mathcal{Y}}, z \in D_{\mathcal{Z}}, \\ &(x, y, z) \succeq (x', y, z) \text{ iff } (x, y', z) \succeq (x', y', z) \end{aligned}$$

given any fixed value z of $\mathcal{Z}$, the preferences over the possible values of $\mathcal{X}$ are independent from the value of $\mathcal{Y}$

Example

Two binary variables A, B with domains $\{a, \bar{a}\}, \{b, \bar{b}\}$

Preference relation: $ab \succ a\bar{b} \succ \bar{a}\bar{b} \succ \bar{a}b$



Preferential independence

Conditional preferential independence (Keeney & Raiffa, 76):

$\mathcal{V}$ set of variables; $\{\mathcal{X}, \mathcal{Y}, \mathcal{Z}\}$ partition of $\mathcal{V}$.

$D_{\mathcal{X}} = \times_{x_i \in \mathcal{X}} D_i$ etc.

$\mathcal{X}$ is preferentially independent from $\mathcal{Y}$ (given $\mathcal{Z}$) iff

$$\begin{aligned} &\text{for all } x, x' \in D_{\mathcal{X}}, y, y' \in D_{\mathcal{Y}}, z \in D_{\mathcal{Z}}, \\ &(x, y, z) \succeq (x', y, z) \text{ iff } (x, y', z) \succeq (x', y', z) \end{aligned}$$

given any fixed value z of $\mathcal{Z}$, the preferences over the possible values of $\mathcal{X}$ are independent from the value of $\mathcal{Y}$

Example

Two binary variables A, B with domains $\{a, \bar{a}\}, \{b, \bar{b}\}$

Preference relation: $ab \succ a\bar{b} \succ \bar{a}\bar{b} \succ \bar{a}b$

A preferentially independent from B



Preferential independence

Conditional preferential independence (Keeney & Raiffa, 76):

$\mathcal{V}$ set of variables; $\{\mathcal{X}, \mathcal{Y}, \mathcal{Z}\}$ partition of $\mathcal{V}$.

$D_{\mathcal{X}} = \times_{X_i \in \mathcal{X}} D_i$ etc.

$\mathcal{X}$ is preferentially independent from $\mathcal{Y}$ (given $\mathcal{Z}$) iff

$$\begin{aligned} &\text{for all } x, x' \in D_{\mathcal{X}}, y, y' \in D_{\mathcal{Y}}, z \in D_{\mathcal{Z}}, \\ &(x, y, z) \succeq (x', y, z) \text{ iff } (x, y', z) \succeq (x', y', z) \end{aligned}$$

given any fixed value z of $\mathcal{Z}$, the preferences over the possible values of $\mathcal{X}$ are independent from the value of $\mathcal{Y}$

Example

Two binary variables A, B with domains $\{a, \bar{a}\}, \{b, \bar{b}\}$

Preference relation: $ab \succ a\bar{b} \succ \bar{a}\bar{b} \succ \bar{a}b$

A preferentially independent from B

B preferentially dependent on A



Plan

- 1 **Languages for compact preference representation**
- 2 **Logical and bidding languages**
 - Propositional logic
 - Bidding languages
- 3 **Graphical languages for ordinal preferences**
 - An introduction to graphical languages
 - Preferential independence
 - CP-nets
 - Outcome optimization with CP-nets
 - CP-nets: extensions and variants
- 4 **Graphical languages for cardinal preferences**
 - Generalized additivity
- 5 **Collective decision making**
 - Voting
 - MultiAgent Resource Allocation
- 6 **Conclusion**



CP-nets: definition

- first reference: (Boutilier, Brafman, Hoos and Poole, 99)
- one of the most popular preference representation languages on combinatorial domains.
- based on conditional preferential independence.

A CP-net is composed of

- a *directed graph* representing the preferential dependencies between variables
- for each variable, a *conditional preference table* expressing the local preference on the values of its domain given the possible combination of values of its parents.



CP-nets: an example

Variables:

- T (time of the flight), with possible values d (day) and n (night);
- S (stopover), with possible values s (yes) and $\bar{s}$ (no);
- C (company), with two possible values c_1 and c_2 .

The agent's preferences:

- she prefers a day flight to a night flight, unconditionally;

$$d \succ n$$

- for a day flight she prefers to make a stopover; for a night flight she prefers not;

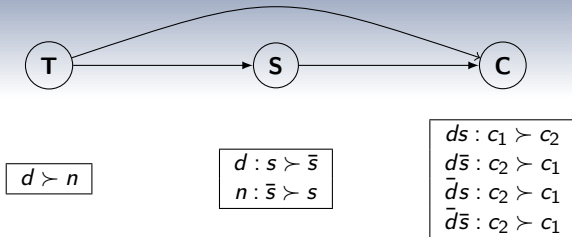
$$d : s \succ \bar{s}; \quad n : \bar{s} \succ s$$

- for a day flight with a stopover she prefers company c_1 because it implies spending a few hours in an airport she likes; in all other cases she prefers c_2 .

$$ds : c_1 \succ c_2; \quad d\bar{s} : c_2 \succ c_1; \quad \bar{d}s : c_2 \succ c_1; \quad \bar{d}\bar{s} : c_2 \succ c_1$$



CP-nets: an example



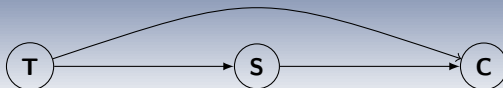
T independent of **S** and **C**; **S** independent of **C**

$d \succ n$ | $T = d$ preferred to $T = n$
everything else (**S**, **C**) being equal (*ceteris paribus*)

$$dsc_1 \succ nsc_1; \quad dsc_2 \succ nsc_2; \quad d\bar{s}c_1 \succ n\bar{s}c_1; \quad d\bar{s}c_2 \succ n\bar{s}c_2$$



CP-nets: an example



$$d \succ n$$

$$\begin{array}{l} d : s \succ \bar{s} \\ n : \bar{s} \succ s \end{array}$$

$$\begin{array}{l} ds : c_1 \succ c_2 \\ d\bar{s} : c_2 \succ c_1 \\ \bar{d}s : c_2 \succ c_1 \\ \bar{d}\bar{s} : c_2 \succ c_1 \end{array}$$

T independent of **S** and **C**; **S** independent of **C**

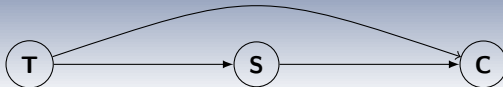
$d : s \succ \bar{s}$ | if $T = d$
 then $S = s$ preferred to $S = \bar{s}$
 everything else (C) being equal (*ceteris paribus*)

$$dsc_1 \succ d\bar{s}c_1; \quad dsc_2 \succ d\bar{s}c_2$$

and likewise: $n\bar{s}c_1 \succ nsc_1; \quad n\bar{s}c_2 \succ nsc_2$



CP-nets: an example



$\mathcal{N}$:

$$d \succ n$$

$$\begin{array}{l} d : s \succ \bar{s} \\ n : \bar{s} \succ s \end{array}$$

$$\begin{array}{l} ds : c_1 \succ c_2 \\ d\bar{s} : c_2 \succ c_1 \\ \bar{d}s : c_2 \succ c_1 \\ \bar{d}\bar{s} : c_2 \succ c_1 \end{array}$$

$$\succ_T : \{dsc_1 \succ nsc_1, dsc_2 \succ nsc_2, d\bar{s}c_1 \succ n\bar{s}c_1, d\bar{s}c_2 \succ n\bar{s}c_2\};$$

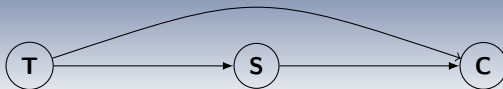
$$\succ_S : \{dsc_1 \succ d\bar{s}c_1, dsc_2 \succ d\bar{s}c_2, n\bar{s}c_1 \succ nsc_1, n\bar{s}c_2 \succ nsc_2\};$$

$$\succ_C : \{dsc_1 \succ dsc_2, d\bar{s}c_2 \succ d\bar{s}c_1, nsc_2 \succ nsc_1, n\bar{s}c_2 \succ n\bar{s}c_1\}$$

$\succ_{\mathcal{N}}$ = preference relation induced by the CP-net $\mathcal{N}$
 = transitive closure of $\succ_T \cup \succ_S \cup \succ_C$



CP-nets: an example



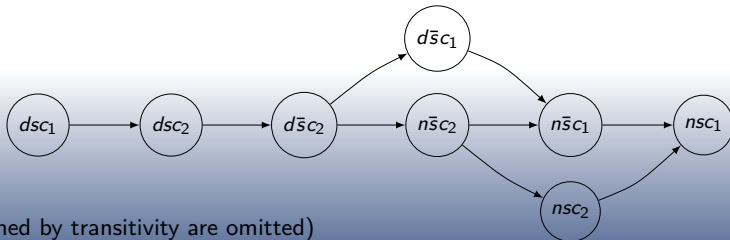
$\mathcal{N}$:

$$d \succ n$$

$$\begin{array}{l} d : s \succ \bar{s} \\ n : \bar{s} \succ s \end{array}$$

$$\begin{array}{l} ds : c_1 \succ c_2 \\ d\bar{s} : c_2 \succ c_1 \\ \bar{d}s : c_2 \succ c_1 \\ \bar{d}\bar{s} : c_2 \succ c_1 \end{array}$$

$\succ_{\mathcal{N}}$:

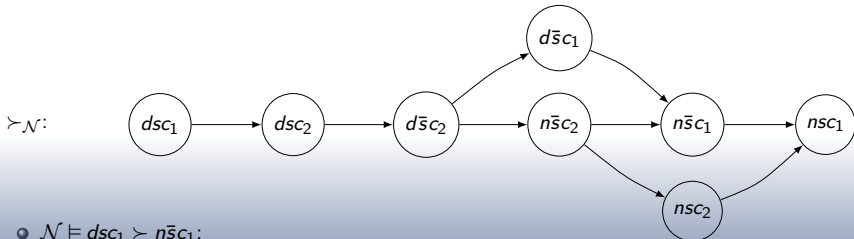


(edges obtained by transitivity are omitted)



CP-nets: semantics

- $\succ_N$ generally not complete
- the complete preference relations extending $\succ_N$ can be viewed as *possible models* of the user's true preference relation
- *consequence* in CP-nets: $\mathcal{N} \models o \succ o'$ if $o \succ o'$ holds in all complete preference relations extending $\succ_N$

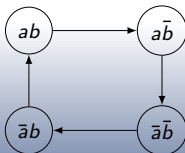
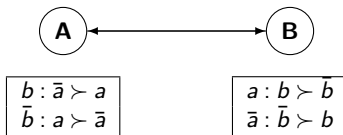


- $\mathcal{N} \models dsc_1 \succ n\bar{s}c_1$;
- $\mathcal{N} \not\models d\bar{s}c_1 \succ n\bar{s}c_2$ and $\mathcal{N} \not\models n\bar{s}c_2 \succ d\bar{s}c_1$



CP-nets: consistency

- a CP-net $\mathcal{N}$ is consistent if its associated preference relation $\succ_N$ is acyclic
- if the dependency graph G of $\mathcal{N}$ is acyclic then $\mathcal{N}$ is consistent
- this is not always true if G is cyclic:





Plan

1 Languages for compact preference representation

2 Logical and bidding languages

Propositional logic

Bidding languages

3 Graphical languages for ordinal preferences

An introduction to graphical languages

Preferential independence

CP-nets

Outcome optimization with CP-nets

CP-nets: extensions and variants

4 Graphical languages for cardinal preferences

Generalized additivity

5 Collective decision making

Voting

MultiAgent Resource Allocation

6 Conclusion



Dominance and optimization

Main purposes of a preference representation language: answer queries such as

- *comparison*: given two outcomes, determine if one of them dominates the other;
- *optimization*: determine some undominated outcome.
- *outcome optimality*: given an outcome, determine if it is undominated.
- *consistency*: determine if the CP-net is consistent.



Outcome optimization

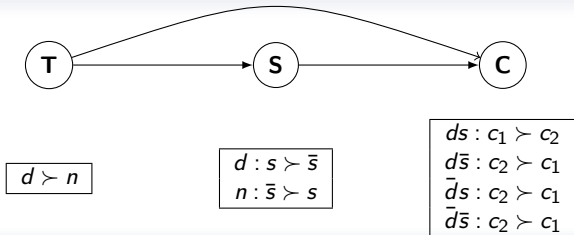
When G is acyclic:

- the CP-net $\mathcal{N}$ is consistent;
- there is a unique nondominated (and dominating) outcome
- this outcome can be found in polynomial time, using the *forward sweep algorithm*



Outcome optimization

Forward sweep: consider variables in sequence in some order compatible with G , and choose the best value given the values of parents variables (already assigned).



- step 1: preferred value of T : $d \Rightarrow T := d$
- step 2: preferred value of S given $T = d$: $s \Rightarrow S := s$
- step 3: preferred value of C given $T = d, S = s$: $c_1 \Rightarrow C := c_1$

optimal outcome: dsc_1



Outcome optimization

Consequence of forward sweep: when G is acyclic, optimization is in P.

More generally:

	G hypertree	G acyclic	any G
optimization	P	in P	NP-hard
comparison	P	NP-hard (in NP?)	PSPACE-complete
outcome optimality	P	P	P
consistency	trivial	trivial	PSPACE-complete



Outcome optimization

General case: translation into a *propositional satisfiability problem*



$$\begin{array}{l} b : a \succ \bar{a} \\ \bar{b} : \bar{a} \succ a \end{array}$$

$$\begin{array}{l} a : b \succ \bar{b} \\ \bar{a} : \bar{b} \succ b \end{array}$$

every entry $u : x \succ \bar{x}$ (resp. $u : \bar{x} \succ x$) of every table
is translated into $u \rightarrow x$ (resp. $u \rightarrow \neg x$)

$$\Phi_{\mathcal{N}} = (b \rightarrow a) \wedge (\neg b \rightarrow \neg a) \wedge (a \rightarrow b) \wedge (\neg a \rightarrow \neg b) \equiv (a \leftrightarrow b)$$

models of $\Phi_{\mathcal{N}} = \{ab, \bar{a}\bar{b}\} =$ undominated outcomes of $\mathcal{N}$

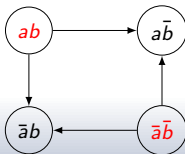


Outcome optimization



$$\begin{array}{l} b : a \succ \bar{a} \\ \bar{b} : \bar{a} \succ a \end{array}$$

$$\begin{array}{l} a : b \succ \bar{b} \\ \bar{a} : \bar{b} \succ b \end{array}$$

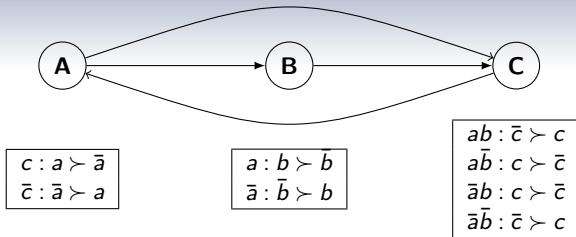


undominated outcomes of $\mathcal{N}$: $\{ab, \bar{a}\bar{b}\}$



Outcome optimization

$\Phi_{\mathcal{N}}$ maybe consistent even when $\mathcal{N}$ is inconsistent



$\succ_{\mathcal{N}}$ has a cycle: $abc \succ \bar{a}bc \succ \bar{a}\bar{b}\bar{c} \succ ab\bar{c} \succ abc$

$$\Phi_{\mathcal{N}} \equiv (a \leftrightarrow c) \wedge (b \leftrightarrow a) \wedge (c \leftrightarrow (a \leftrightarrow b)) \equiv \neg a \wedge \neg b \wedge c$$

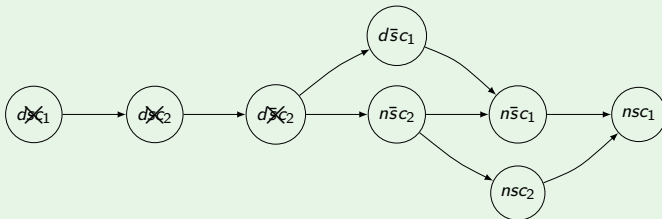
models of $\Phi_{\mathcal{N}} = \{\bar{a}\bar{b}c\}$: $\bar{a}\bar{b}c$ is non-dominated.



Constrained optimization

- sometimes not all outcomes are feasible.
- *constrained CP-net*: CP-net $\mathcal{N}$ + set of constraints Γ ;
- find an outcome α both feasible and undominated, *i.e.*, such that there is no feasible outcome β such that $\beta \succ_{\mathcal{N}} \alpha$ (Boutilier *et al.*, 04).

Example



Constraint: $\Gamma = \{T = d \Rightarrow S = \bar{s}, C = c_2 \Rightarrow T = n\}$

Undominated feasible outcomes: $d\bar{s}c_1$ and $n\bar{s}c_2$.

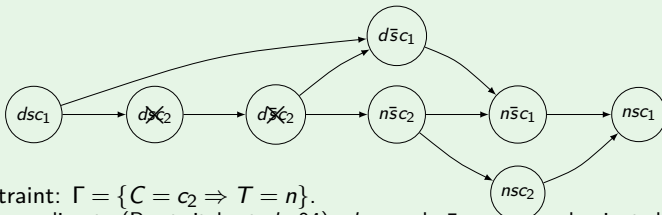


Constrained optimization

A different way of defining optimal solutions for constrained CSPs (Prestwich *et al.*, 04):

- α dominates β if there is a flipping sequence from α to β that goes through feasible outcomes only;
- again we look for non-dominated feasible outcomes.

Example



Constraint: $\Gamma = \{C = c_2 \Rightarrow T = n\}$.

- according to (Prestwich *et al.*, 04): dsc_1 and $n\bar{s}c_2$ are non-dominated;
- according to (Boutilier *et al.*, 04): only dsc_1 is non-dominated.



Plan

1 Languages for compact preference representation

2 Logical and bidding languages

Propositional logic

Bidding languages

3 Graphical languages for ordinal preferences

An introduction to graphical languages

Preferential independence

CP-nets

Outcome optimization with CP-nets

CP-nets: extensions and variants

4 Graphical languages for cardinal preferences

Generalized additivity

5 Collective decision making

Voting

MultiAgent Resource Allocation

6 Conclusion



TCP-nets

- (Brafman, Domshlak & Shimony, 06)
- enrich CP-nets by allowing importance statements between variables

A TCP-net contains

- conditional preference statements just like in CP-nets
- unconditional importance statements such as $A \triangleright B$ (A is more important than B)
- conditional importance statements such as $A = a : B \triangleright C$ (if $A = a$ then B is more important than C)

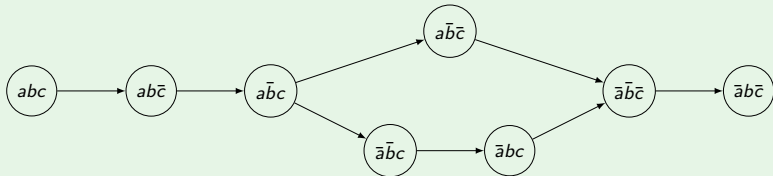


TCP-nets

Example

Importance statements: $a : B \triangleright C$
 $\bar{a} : C \triangleright B$

Preference statements: $a \succ \bar{a}$ $a : b \succ \bar{b}$ $c \succ \bar{c}$
 $\bar{a} : \bar{b} \succ b$





CP-theories

- (Wilson, 06)
- enrich TCP-nets (and *a fortiori*, than CP-nets) by allowing to specify the set of variables that must be fixed.

Example:

if a then $b \succ \bar{b}$ whatever the value of C and D ,
ceteris paribus (the value of D , etc. being fixed)

CP-theories are more general than TCP-nets (and *a fortiori*, than CP-nets);



CI-nets

- (Bouveret, Endriss and Lang, 09)
- allow to express importance statements between sets of variables

Example:

if a and $\bar{b}$ then C, D, E together are more important than F, G together,
ceteris paribus

CI-networks are useful for expressing preferences over sets of goods (compact preference language for fair division)



CI-nets: the language

Conditional importance statement

$S^+, S^- : S_1 \triangleright S_2$ (with S^+, S^-, S_1 and S_2 pairwise-disjoint).

Example: $\overline{ad} : b \triangleright ce$ implies for example $ab \succ ace$, $abfg \succ acefg$, ...

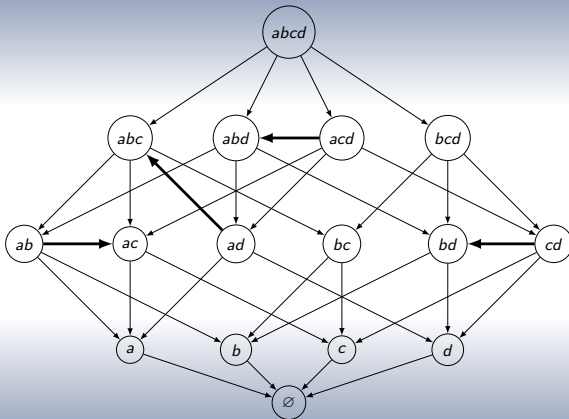
CI-net

A CI-net on $\mathcal{V}$ is a set $\mathcal{N}$ of conditional importance statements on $\mathcal{V}$.



Semantics

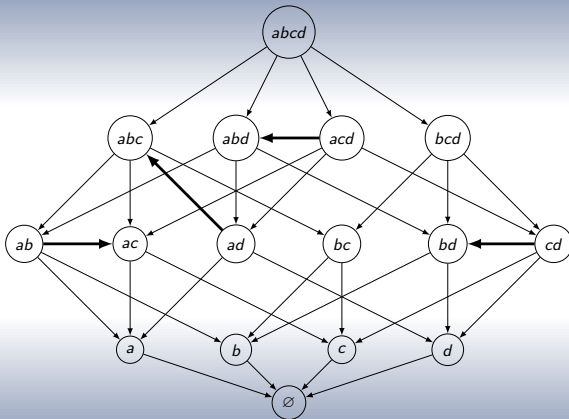
A CI-net of 4 objects $\{a, b, c, d\}$: $\{a : d \triangleright bc, \overline{ad} : b \triangleright c, d : c \triangleright b\}$





Semantics

A CI-net of 4 objects $\{a, b, c, d\}$: $\{a : d \triangleright bc, \overline{a}d : b \triangleright c, d : c \triangleright b\}$



Induced preference relation $\succ_{\mathcal{N}}$: the smallest preference **monotonic** relation compatible with **all CI-statements**.



Worsening flips

Worsening flip

$\mathcal{V}_1 \rightsquigarrow \mathcal{V}_2$ is called a **worsening flip** wrt. $\mathcal{N}$ if:

- either $\mathcal{V}_1 \subseteq \mathcal{V}_2$ (monotonicity flip);
- or they match a CI-statement in $\mathcal{N}$ (CI-flip).

Proposition (dominance)

We have $A \succ_{\mathcal{N}} B$ if and only if there exists a sequence of worsening flips from A to B wrt. $\mathcal{N}$.

Proposition (satisfiability)

A CI-net $\mathcal{N}$ is satisfiable if and only if it does not possess any cycle of worsening flips.



Expressivity

Proposition

CI-nets can express all strict monotonic preference relations on 2^V .

Proposition

Full expressivity is lost as soon as:

- (i) we do not allow positive preconditions;
- (ii) we do not allow negative preconditions;
- (iii) the cardinality of compared sets is bounded by a fixed integer.



Optimization

[non-dominated]

Input: A satisfiable CI-net $\mathcal{N}$, a bundle X .

Question: Is X non dominated for $\succ_{\mathcal{N}}$?



Optimization

[non-dominated]

Input: A satisfiable CI-net $\mathcal{N}$, a bundle X .

Question: Is X non dominated for $\succ_{\mathcal{N}}$?

Irrelevant: the entire set is the only one non dominated set!



Optimization

[non-dominated]

Input: A satisfiable CI-net $\mathcal{N}$, a bundle X .

Question: Is X non dominated for $\succ_{\mathcal{N}}$?

Irrelevant: the entire set is the only one non dominated set!

More interesting:

- **Constrained optimization**
- **Resource allocation**



Dominance

[dominance]

Input: A (satisfiable) CI-net $\mathcal{N}$, two bundles X and Y .

Question: $X \succ_{\mathcal{N}} Y$?



Dominance

[dominance]

Input: A (satisfiable) CI-net $\mathcal{N}$, two bundles X and Y .

Question: $X \succ_{\mathcal{N}} Y$?

Some bad news. . .

Proposition

[DOMINANCE] in satisfiable CI-nets is **PSPACE**-complete, even under any of these restrictions:

- ❶ every CI-statement bears on singletons and has no negative preconditions;
- ❷ every CI-statement bears on singletons and has no positive preconditions;
- ❸ every CI-statement is precondition-free.



Dominance

[dominance]

Input: A (satisfiable) CI-net $\mathcal{N}$, two bundles X and Y .

Question: $X \succ_{\mathcal{N}} Y$?

Some good news. . .

SCI-nets: precondition-free, singleton-comparing CI-statements.

Example: $\{a \triangleright c, b \triangleright c, e \triangleright d\}$.

Proposition

[DOMINANCE] in satisfiable SCI-nets is in P.



Satisfiability

[satisfiability]

Input: A CI-net $\mathcal{N}$.

Question: Is $\mathcal{N}$ satisfiable ?



Satisfiability

[satisfiability]

Input: A CI-net $\mathcal{N}$.

Question: Is $\mathcal{N}$ satisfiable ?

Some bad news. . .

Proposition

[SATISFIABILITY] for CI-nets is **PSPACE**-complete.



Satisfiability

[satisfiability]

Input: A CI-net $\mathcal{N}$.

Question: Is $\mathcal{N}$ satisfiable ?

Some good news. . .

- [SATISFIABILITY] for SCI-nets is in P.
- Two **sufficient** conditions for satisfiability: based on **acyclicity**.



Outline

- 1 **Languages for compact preference representation**
- 2 **Logical and bidding languages**
 - Propositional logic
 - Bidding languages
- 3 **Graphical languages for ordinal preferences**
 - An introduction to graphical languages
 - Preferential independence
 - CP-nets
 - Outcome optimization with CP-nets
 - CP-nets: extensions and variants
- 4 **Graphical languages for cardinal preferences**
 - Generalized additivity
- 5 **Collective decision making**
 - Voting
 - MultiAgent Resource Allocation
- 6 **Conclusion**



Plan

1 Languages for compact preference representation

2 Logical and bidding languages

Propositional logic

Bidding languages

3 Graphical languages for ordinal preferences

An introduction to graphical languages

Preferential independence

CP-nets

Outcome optimization with CP-nets

CP-nets: extensions and variants

4 Graphical languages for cardinal preferences

Generalized additivity

5 Collective decision making

Voting

MultiAgent Resource Allocation

6 Conclusion



Extend additivity...

Additive utility functions are probably the simplest and the most intuitive ones. If $\mathcal{O}$ is of size p , an additive utility function is only defined by p weights.

Example: $u = 3o_1 + 7o_2 - 2o_3$ is an additive function. $u(\{o_1, o_2\}) = 10$.



Extend additivity...

Additive utility functions are probably the simplest and the most intuitive ones. If $\mathcal{O}$ is of size p , an additive utility function is only defined by p weights.

Example: $u = 3o_1 + 7o_2 - 2o_3$ is an additive function. $u(\{o_1, o_2\}) = 10$.

But... Additive utility functions are unable to represent dependencies (synergies) between object.

Is there a way to represent these dependencies while keeping the representation not too big ?



The k -additive form

- A utility function is called **k -additive** iff the utility assigned to a bundle π can be represented as the sum of basic utilities assigned to subsets of π with cardinality $\leq k$ (limited synergies).
- The **k -additive form** of representing utility functions:

$$u(\pi) = \sum_{\pi' \subseteq \pi} \alpha^{\pi'} \text{ with } \alpha^{\pi'} = 0 \text{ whenever } |\pi'| > k$$

Example: $u = 3o_1 + 7o_2 - 2o_2o_3$ is a 2-additive function

- That is, specifying a utility function in this language means specifying the **coefficients** α^π for bundles $\pi \subseteq \mathcal{O}$.
- The value α^π can be seen as the additional benefit incurred from owning the items in π **together**, i.e. beyond the benefit of owning all proper subsets.



The k -additive form – expressivity

The k -additive form is **fully expressive**, if we choose k large enough:

Proposition

Any utility function is representable in the k -additive form for some $k \leq |\mathcal{O}|$.

Proof: For any utility function u , we can define coefficients α^π :

- $\alpha^\emptyset = u(\emptyset)$;
- $\alpha^\pi = u(\pi) - \sum_{\pi' \subsetneq \pi} \alpha^{\pi'}$ for all $\pi \subseteq \mathcal{O}$.

The function $u \mapsto (\pi \mapsto \alpha^\pi)$ is called the **Möbius transform** of u [Grabisch, 1997].



Grabisch, M. (1997).

k -order additive discrete fuzzy measure and their representation.
Fuzzy Sets and Systems, 92:167–189.



Example

$$\mathcal{O} = \{o_1, o_2, o_3, o_4\}$$

$$u = 2o_1 + 3o_2 + 2o_2o_3 + 2o_3o_4$$

	$\emptyset$	o_1	o_2	o_3	o_4	o_1o_2	o_1o_3	o_1o_4	o_2o_3	o_2o_4
u	0	2	3	0	0	5	2	2	5	3

	o_3o_4	$o_1o_2o_3$	$o_1o_2o_4$	$o_2o_3o_4$	$o_1o_2o_3o_4$
u	2	7	5	7	9



Explicit vs. k -additive Form

Among the k -additive form and the explicit one, which one is the **most succinct** ?



Explicit vs. k -additive Form

Among the k -additive form and the explicit one, which one is the **most succinct** ?

Proposition

The explicit and the k -additive form of representing utility functions are incomparable with respect to succinctness.

Proof sketch: The following two functions can be used to prove the mutual lack of a polysize reduction:

- $u_1(\pi) = |\pi|$: representing u_1 requires $|\mathcal{O}|$ non-zero coefficients in the k -additive form (linear); but $2^{|\mathcal{O}|} - 1$ non-zero values in the explicit form (exponential).
- $u_2(\pi) = 1$ for $|\pi| = 1$ and $u_2(\pi) = 0$ otherwise: requires $|\mathcal{O}|$ non-zero values in the explicit form (linear); but $2^{|\mathcal{O}|} - 1$ non-zero coefficients in the k -additive form (exponential), namely $\alpha^\pi = 1$ for $|\pi| = 1$, $\alpha^\pi = -2$ for $|\pi| = 2$, $\alpha^\pi = 3$ for $|\pi| = 3$, ...



k -additive form: complexity

- [DOMINANCE] is in P.
- [OPTIMIZATION] is NP-complete, but in P if all the weights are positive.
- [EEF EXISTENCE] with preemption constraints only is Σ_2^P -complete ($\Sigma_2^P = \mathbf{NP}^{\mathbf{NP}}$) even for additive preferences.
- [MAX-CUF] $\leadsto$ see later.



Generalized Additive Independence

$\mathcal{X}$ set of variables (binary or not), v instantiation.

Definition

Let $\mathcal{X}_1, \dots, \mathcal{X}_k$ be a family of subsets of $\mathcal{X}$ such that $\bigcup_i \mathcal{X}_i = \mathcal{X}$.
 u is GAI-decomposable with respect to $\mathcal{X}_1, \dots, \mathcal{X}_k$ iff $\exists u_1 \dots u_k$ utility functions $\mathcal{X}_i \rightarrow \mathbb{R}$ such that $u(v) = \sum_{i=1}^k u_i(v \downarrow \mathcal{X}_i)$

Remarks:

- For binary variables, additivity = GAI-decomposability, with $|\mathcal{X}_i| = 1, \forall i$.
- For binary variables, k -Additivity = GAI-decomposability, with $|\mathcal{X}_i| \leq k, \forall i$.



GAI-nets: example

{**house**, **tractor**, **lawn mower**, **fields**}

<i>f</i>	0	1
<i>u</i> ₁	0	10

<i>h</i>	0	1
<i>u</i> ₂	0	15

<i>tr</i>	0	1
<i>u</i> ₃	0	2

<i>lm</i>	0	1
<i>u</i> ₄	0	1

<i>f</i>	0	0	1	1
<i>tr</i>	0	1	0	1
<i>u</i> ₅	0	0	0	8

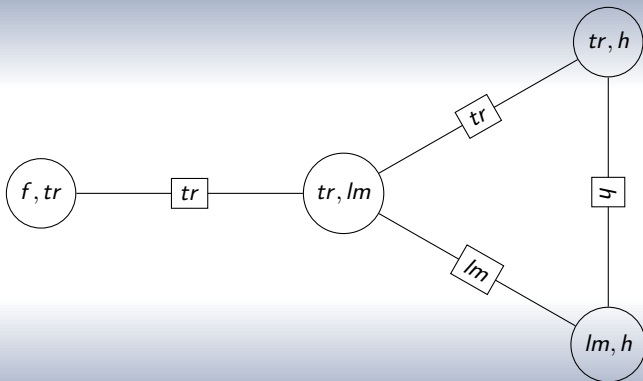
<i>tr</i>	0	0	1	1
<i>lm</i>	0	1	0	1
<i>u</i> ₆	0	0	0	−8

<i>h</i>	0	0	1	1
<i>tr</i>	0	1	0	1
<i>u</i> ₇	0	0	0	7

<i>h</i>	0	0	1	1
<i>lm</i>	0	1	0	1
<i>u</i> ₈	0	0	0	10



GAI-nets





Outline

- 1 **Languages for compact preference representation**
- 2 **Logical and bidding languages**
 - Propositional logic
 - Bidding languages
- 3 **Graphical languages for ordinal preferences**
 - An introduction to graphical languages
 - Preferential independence
 - CP-nets
 - Outcome optimization with CP-nets
 - CP-nets: extensions and variants
- 4 **Graphical languages for cardinal preferences**
 - Generalized additivity
- 5 **Collective decision making**
 - Voting
 - MultiAgent Resource Allocation
- 6 **Conclusion**



Plan

1 Languages for compact preference representation

2 Logical and bidding languages

Propositional logic

Bidding languages

3 Graphical languages for ordinal preferences

An introduction to graphical languages

Preferential independence

CP-nets

Outcome optimization with CP-nets

CP-nets: extensions and variants

4 Graphical languages for cardinal preferences

Generalized additivity

5 Collective decision making

Voting

MultiAgent Resource Allocation

6 Conclusion



Voting

- $\mathcal{A} = \{1, \dots, n\}$ finite set of **voters** ;
- $\mathcal{X}$ finite set of **candidates** (= alternatives);
- **profile**:

$$P = (\succ_1, \dots, \succ_n)$$

$\succ_i$ = linear order on $\mathcal{X}$ (vote) expressed by voter i .

A **voting rule** maps every profile to a candidate.



Voting

An important problem in social choice: **voting in a combinatorial domain**.

Examples:

Example 1 choosing a common menu:

$$\begin{aligned}\mathcal{X} = & \quad \{\text{soup, salad}\} \\ & \times \quad \{\text{fish, meat}\} \\ & \times \quad \{\text{apple pie, ice cream}\}\end{aligned}$$

Example 2 multiple referendum: a local community has to decide on several interrelated issues (should we build a swimming pool or not? should we build a tennis court or not?)

Example 3 recruiting committee (3 positions, 6 applicants):

$$\mathcal{X} = \{A \mid A \subseteq \{a, b, c, d, e, f\}, |A| \leq 3\}.$$

Combinatorial domains: $\mathcal{V} = \{X_1, \dots, X_p\}$ set of **variables**, or **issues**;

$\mathcal{X} = D_1 \times \dots \times D_p$ (where D_i is a finite value domain for variable X_i)

Voting in a combinatorial domain: choose collectively a value for each variable.



Voting in combinatorial domains

A naive solution: **don't bother and vote separately on each variable.**

⇒ **multiple election paradoxes**

Example

2 binary variables S (build a new swimming pool), T (build a new tennis court)

voters 1 and 2 $S\bar{T} \succ \bar{S}T \succ \bar{S}\bar{T} \succ ST$

voters 3 and 4 $\bar{S}T \succ S\bar{T} \succ \bar{S}\bar{T} \succ ST$

voter 5 $ST \succ S\bar{T} \succ \bar{S}T \succ \bar{S}\bar{T}$

Problem 1: voters 1-4 feel ill at ease reporting a preference on $\{S, \bar{S}\}$ and $\{T, \bar{T}\}$



Voting in combinatorial domains

A naive solution: **don't bother and vote separately on each variable.**
⇒ **multiple election paradoxes**

Example

2 binary variables S (build a new swimming pool), T (build a new tennis court)

voters 1 and 2 $S\bar{T} \succ \bar{S}T \succ \bar{S}\bar{T} \succ ST$

voters 3 and 4 $\bar{S}T \succ S\bar{T} \succ \bar{S}\bar{T} \succ ST$

voter 5 $ST \succ S\bar{T} \succ \bar{S}T \succ \bar{S}\bar{T}$

Problem 1: voters 1-4 feel ill at ease reporting a preference on $\{S, \bar{S}\}$ and $\{T, \bar{T}\}$

Problem 2: suppose they do so by an “optimistic” projection

- voters 1, 2 and 5: S ; voters 3 and 4: $\bar{S} \Rightarrow$ decision = S ;
- voters 3,4 and 5: T ; voters 1 and 2: $\bar{T} \Rightarrow$ decision = T .

Alternative ST is chosen although it is the worst alternative for all but one voter.



Voting and graphical languages

The input of a voting problem is a collection of preference relations

⇒ CP-nets graphical languages for ordinal preferences are relevant for voting in combinatorial domains

Two ways of using graphical languages in this context.

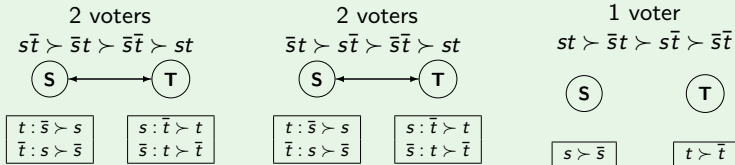
- ① **elicit preferences globally, and then aggregate them:**
- ② **proceed sequentially, interleaving elicitation and aggregation:**



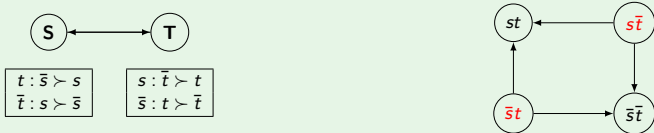
Voting and CP-nets: aggregating CP-nets

Example

Swimming pool and tennis: 5 voters, 2 binary variables S , T



Aggregate locally (by majority) for each pair of adjacent outcomes:

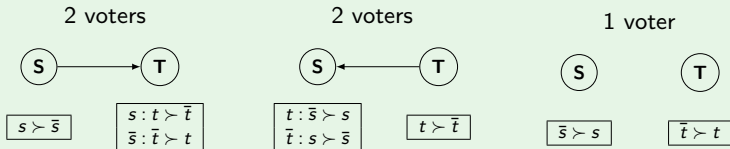




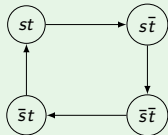
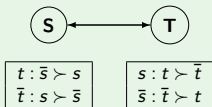
Voting and CP-nets: aggregating CP-nets

Example

Another example: again 5 voters and 2 binary variables S , T



Aggregate locally (by majority) for each pair of adjacent outcomes:





Voting and CP-nets: aggregating CP-nets

- + always applicable, because any preference relation is compatible with some CP-net (possibly with cyclic dependencies).
- elicitation cost: in the worst case, exponential number of queries to each voter
- computation cost: dominance in CP-nets with cyclic dependencies is PSPACE-complete
- there might be no winner; there might be several winners

(Xia, Conitzer and Lang, 08/11)



Voting and CP-nets: sequential voting

Assumption: there exists an order on variables, say $X_1 > \dots > X_p$, such that for every voter and for every i , X_i is preferentially independent of $X_{i+1}, \dots, X_p$ given $X_1, \dots, X_{i-1}$.

Equivalently: if the dependencies graphs of the agents are $G_1, \dots, G_n$ then $G_1 \cup \dots \cup G_n$ is acyclic.

Sequential voting: apply local voting rules, one variable after the other, in an order compatible with G .

At every step:

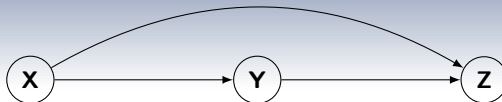
- we elicit the voters' preferences about a single variable;
- a local rule is used to compute the value chosen for this variable;
- this value is communicated to the voters.

We don't need to know the whole preference relations of the voters but only a part of their CP-nets.

(Lang & Xia, 09)



Voting and CP-nets: sequential voting



- ❶ elicit voters' preferences on **X** (possible because their preferences on **X** are unconditional);
- ❷ apply local voting rule r_X and determine the "local" winner x^* ;
- ❸ elicit voters' preferences on **Y** given $\mathbf{X} = x^*$ (possible because their preferences on **Y** depend only on **X**);
- ❹ apply local voting rule r_Y and determine y^* ;
- ❺ elicit voters' preferences on **Z** given $\mathbf{X} = x^*$ and $\mathbf{Y} = y^*$.
- ❻ apply local voting rule r_Z and determine z^* .
- ❼ winner: (x^*, y^*, z^*)



Voting and CP-nets: sequential voting

Example: $r_X = r_Y = \text{majority rule}$

3 voters
$\bar{x} \succ x$
$x : \bar{y} \succ y$
$\bar{x} : y \succ \bar{y}$

2 voters
$x \succ \bar{x}$
$x : y \succ \bar{y}$
$\bar{x} : \bar{y} \succ y$

2 voters
$x \succ \bar{x}$
$\bar{y} \succ y$

order: $X > Y$.

4 voters out of 7 prefer x to $\bar{x} \Rightarrow x^* = r_X(\succ_1, \dots, \succ_7) = x$



Voting and CP-nets: sequential voting

Example: $r_X = r_Y = \text{majority rule}$

3 voters	2 voters	2 voters
$\bar{x} \succ x$	$x \succ \bar{x}$	$x \succ \bar{x}$
$x : \bar{y} \succ y$	$x : y \succ \bar{y}$	$\bar{y} \succ y$
$\bar{x} : y \succ \bar{y}$	$\bar{x} : \bar{y} \succ y$	

order: $X > Y$.

4 voters out of 7 prefer x to $\bar{x} \Rightarrow x^* = r_X(\succ_1, \dots, \succ_7) = x$

given $X = x$, 5 voters out of 7 prefer $\bar{y}$ to $y \Rightarrow y^* = \bar{y}$

$$\text{Seq}(r_X, r_Y)(P) = x\bar{y}$$



Plan

- 1 **Languages for compact preference representation**
- 2 **Logical and bidding languages**
 - Propositional logic
 - Bidding languages
- 3 **Graphical languages for ordinal preferences**
 - An introduction to graphical languages
 - Preferential independence
 - CP-nets
 - Outcome optimization with CP-nets
 - CP-nets: extensions and variants
- 4 **Graphical languages for cardinal preferences**
 - Generalized additivity
- 5 **Collective decision making**
 - Voting
 - MultiAgent Resource Allocation
- 6 **Conclusion**

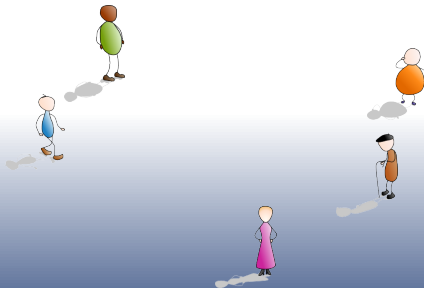


The resource allocation problem...



The resource allocation problem...

- A finite set $\mathcal{N}$ of **agents** .





The resource allocation problem...

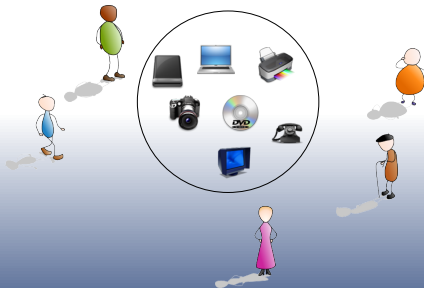
- A finite set $\mathcal{N}$ of **agents** .
- A limited common **resource**.





The resource allocation problem...

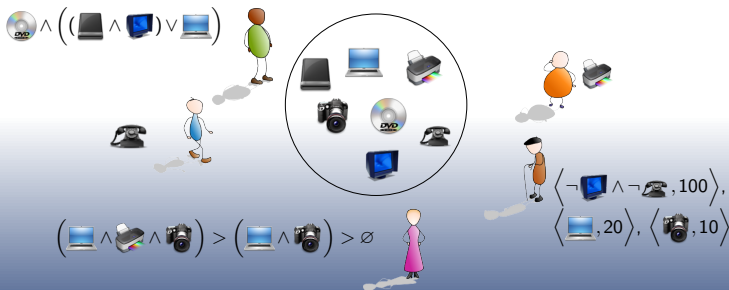
- A finite set $\mathcal{N}$ of **agents** .
- A limited common **resource**.





The resource allocation problem...

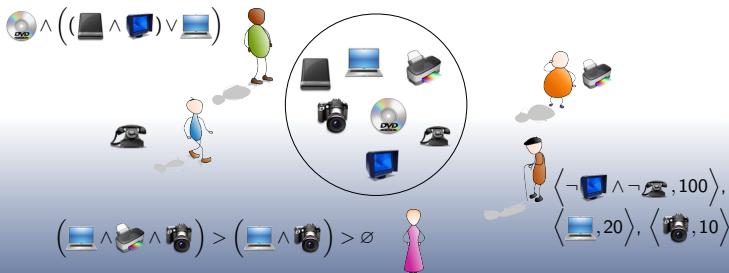
- A finite set $\mathcal{N}$ of **agents** having some **requests** and **preferences** on the resources.
- A limited common **resource**.





The resource allocation problem...

- A finite set $\mathcal{N}$ of **agents** having some **requests** and **preferences** on the resources.
- A limited common **resource**.
- A set of **constraints** (physical, legal, moral, ...).

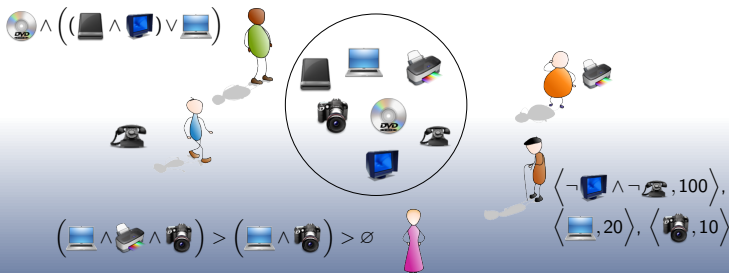


A bundle cannot exceed the transport capacity of an agent.



The resource allocation problem...

- A finite set $\mathcal{N}$ of **agents** having some **requests** and **preferences** on the resources.
- A limited common **resource**.
- A set of **constraints** (physical, legal, moral, ...).
- An **optimization** or **decision** criterion.



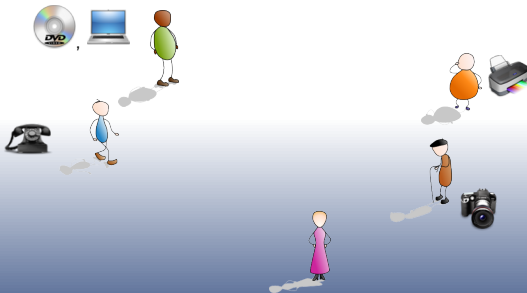
A bundle cannot exceed the transport capacity of an agent.



The resource allocation problem...

- A finite set $\mathcal{N}$ of **agents** having some **requests** and **preferences** on the resources.
- A limited common **resource**.
- A set of **constraints** (physical, legal, moral, ...).
- An **optimization** or **decision** criterion.

How to allocate a part of or the whole resource to each agent such that no constraint is violated, and the criterion is optimized or verified.

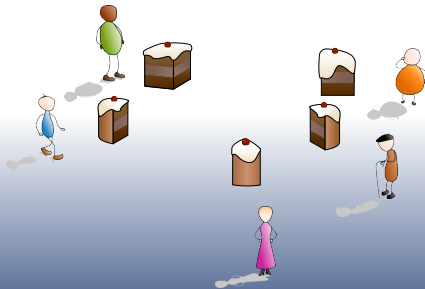




The resource allocation problem...

- A finite set $\mathcal{N}$ of **agents** having some **requests** and **preferences** on the resources.
- A limited common **resource**.
- A set of **constraints** (physical, legal, moral, ...).
- An **optimization** or **decision** criterion.

How to allocate a part of or the whole resource to each agent such that no constraint is violated, and the criterion is optimized or verified.





Real-world applications

An ubiquitous problem. . .

- Fair share of Earth Observation Satellites.
- Tasks or subjects allocation.
- Combinatorial auctions problems [Cramton et al., 2006].
- Allocation of waste water treatment rights [Murillo Espinar, 2010].
- Computer network sharing, rostering problems, allocation of take-off and landing slots in airports [Faltings, 2005],. . . .



Cramton, P., Shoham, Y., and Steinberg, R., editors (2006).

Combinatorial Auctions.

MIT Press.



Faltings, B. (2005).

A budget-balanced, incentive-compatible scheme for social choice.

In Faratin, P. and Rodriguez-Aguilar, J. A., editors, *Agent-Mediated Electronic Commerce VI*, volume 3435 of *LNAI*, pages 30–43. Springer.



Murillo Espinar, J. (2010).

Egalitarian Behaviour in Multiunit Combinatorial Auctions.

PhD thesis, Universitat de Girona.



A resource to share

- Resource may be **continuous** (e.g. energy) or **discrete** (e.g. laptop)
- It can be **sharable** (e.g. picture) or **non sharable** (e.g. laptop)
- Other dichotomies: **static** vs. **time-dependent**, **single-unit** vs. **multi-unit**, **tasks** vs. **goods**



A resource to share

- Resource may be **continuous** (e.g. energy) or **discrete** (e.g. laptop)
- It can be **sharable** (e.g. picture) or **non sharable** (e.g. laptop)
- Other dichotomies: **static** vs. **time-dependent**, **single-unit** vs. **multi-unit**, **tasks** vs. **goods**

In the following, we focus on **discrete**, **static**, **single-unit** resource.

- Elements of $\mathcal{G}$ are called (indivisible) **items**, **goods** or **objects**.
- A **bundle** is an element $\pi \in 2^{\mathcal{G}}$.
- an **allocation** is a vector $\vec{\pi} = \langle \pi_1, \dots, \pi_n \rangle \in 2^{\mathcal{G}^n}$
- $\pi_i \cap \pi_j = \emptyset$ if the goods are non shareable.



Preferences for MARA

- **Assumption of non-exogenous preferences:** the set of alternatives is $2^{\mathcal{O}}$.
- **Assumption of monotonicity:** $\pi \subseteq \pi' \Rightarrow \pi \preceq \pi'$



Auctions

A special case of multiagent resource allocation problem

- Numerical preferences: **money**.
- **Usual criterion**: maximize the revenue of the auctioneer.
- **Classical auctions**:
 - Additive preferences.
 - Each object is sold to the bidder proposing the highest price.
- **Combinatorial auctions** [Cramton et al., 2006]
 - preferential dependencies (→ bidding languages)
 - *allocate the objects* → **Winner Determination Problem**



The WDP

Winner Determination Problem

- **Input** : A set of bids (XOR, OR, or other...)
- **Solution** : An allocation that maximizes the sum of utilities (*i.e* the auctioneer's payment).

Example

- **Agent 1** : ($\{\text{DVD}, \text{Laptop}, \text{Printer}\}, 400\text{€}$)
- **Agent 2** : ($\{\text{Camera}, \text{Printer}\}, 700\text{€}$) *OR* ($\{\text{Phone}\}, 100\text{€}$)



The WDP

Winner Determination Problem

- **Input** : A set of bids (XOR, OR, or other...)
- **Solution** : An allocation that maximizes the sum of utilities (*i.e* the auctioneer's payment).

Example

- **Agent 1** : ($\{\text{DVD}, \text{Laptop}, \text{Printer}\}, 400\text{€}$)
- **Agent 2** : ($\{\text{Camera}, \text{Printer}\}, 700\text{€}$) OR ($\{\text{Phone}\}, 100\text{€}$)

Solution :

- **Agent 1** : Nothing
- **Agent 2** : $\{\text{Camera}, \text{Printer}, \text{Phone}\} \rightarrow u = 800 \text{ €}$.



The WDP: solving

- NP-complete problem (for most languages)
- Efficient solving:

- linear formulation: x_i 0-1 variables, bids (π_i, w_i)

$$\begin{aligned} & \text{maximize } \sum_i w_i \times x_i \\ & \forall i \neq j \text{ t.q. } \pi_i \cap \pi_j \neq \emptyset, x_i + x_j \leq 1 \end{aligned}$$

- branching algorithms (objects / bids),
 - approximated algorithms...



Auctions and fairness

Example

- Agent 1 : ($\{\text{DVD}, \text{Laptop}, \text{Printer}\}, 400\text{€}\)$)
- Agent 2 : ($\{\text{Camera}, \text{Printer}\}, 700\text{€}\)$ OR ($\{\text{Phone}\}, 100\text{€}\)$)

Solution :

- Agent 1 : Nothing
- Agent 2 : $\{\text{Camera}, \text{Printer}, \text{Phone}\} \rightarrow u = 800 \text{ €}.$



Auctions and fairness

Example

- Agent 1 : ($\{\text{DVD}, \text{hard drive}, \text{printer}\}, 400\text{€}$)
- Agent 2 : ($\{\text{camera}, \text{printer}\}, 700\text{€}$) OR ($\{\text{phone}\}, 100\text{€}$)

Solution (fairer) :

- Agent 1 : $\{\text{DVD}, \text{hard drive}, \text{printer}\} \rightarrow u = 400 \text{ €}.$
- Agent 2 : $\{\text{phone}\} \rightarrow u = 100 \text{ €}.$



Usual Social Welfare Orderings

- Classical utilitarian order.
- Egalitarian order.
- Leximin egalitarian order.
- Compromises between classical utilitarianism and egalitarianism: Nash ($\times$), families OWA and sum of powers,...



Usual Social Welfare Orderings

- **Classical utilitarian order.**
- Egalitarian order.
- Leximin egalitarian order.
- Compromises between classical utilitarianism and egalitarianism: Nash ($\times$), families OWA and sum of powers,...

Classical utilitarianism [Harsanyi]

$$\vec{u} \preceq \vec{v} \Leftrightarrow \sum_{i=1}^n u_i \leq \sum_{i=1}^n v_i.$$

Features

Conveys the sum-fitness principle (resource goes to who makes the best use of it).

Indifferent to inequalities (Pigou-Dalton) $\leadsto$ can lead to huge inequalities between the agents.



Usual Social Welfare Orderings

- Classical utilitarian order.
- **Egalitarian order.**
- Leximin egalitarian order.
- Compromises between classical utilitarianism and egalitarianism: Nash ($\times$), families OWA and sum of powers,...

Egalitarianism [Rawls]

$$\vec{u} \preceq \vec{v} \Leftrightarrow \min_{i=1}^n u_i \leq \min_{i=1}^n v_i.$$

Features

Conveys the compensation principle: the least well-off must be made as well-off as possible (justice according to needs) $\leadsto$ tends to equalize the utility profile.



Usual Social Welfare Orderings

- Classical utilitarian order.
- Egalitarian order.
- Leximin egalitarian order.
- Compromises between classical utilitarianism and egalitarianism: Nash ($\times$), families OWA and sum of powers,...

Egalitarianism [Rawls]

$$\vec{u} \preceq \vec{v} \Leftrightarrow \min_{i=1}^n u_i \leq \min_{i=1}^n v_i.$$

Features

Conveys the compensation principle: the least well-off must be made as well-off as possible (justice according to needs) $\leadsto$ tends to equalize the utility profile.

However, it can lead to non Pareto-efficient decisions (drowning effect).



Usual Social Welfare Orderings

- Classical utilitarian order.
- Egalitarian order.
- Leximin egalitarian order.
- Compromises between classical utilitarianism and egalitarianism: Nash ($\times$), families OWA and sum of powers,...

Egalitarianism [Rawls]

$$\vec{u} \preceq \vec{v} \Leftrightarrow \min_{i=1}^n u_i \leq \min_{i=1}^n v_i.$$

Egalitarian SWO and Pareto-efficiency

$\langle 1, 1, 1, 1 \rangle \sim \langle 1000, 1, 1000, 1000 \rangle$, whereas $\langle 1, 1, 1, 1 \rangle$ and
 $\langle 1000, 1, 1000, 1000 \rangle$ are very different!



Usual Social Welfare Orderings

- Classical utilitarian order.
- Egalitarian order.
- Leximin egalitarian order.
- Compromises between classical utilitarianism and egalitarianism: Nash ($\times$), families OWA and sum of powers,...

Leximin egalitarianism [Sen, 1970; Kolm, 1972]

Let $\vec{x}$ be a vector. We write $\vec{x}^\uparrow$ the sorted version of $\vec{x}$.

$\vec{u} \succ_{\text{leximin}} \vec{v} \Leftrightarrow \exists k$ such that $\forall i \leq k, u_i^\uparrow = v_i^\uparrow$ and $u_{k+1}^\uparrow > v_{k+1}^\uparrow$.

This is a lexicographical comparison over sorted vectors.

Perform a leximin comparison...

Two vectors to compare: $\vec{u} = \langle 4, 10, 3, 5 \rangle$ and $\vec{v} = \langle 4, 3, 6, 6 \rangle$.

- We sort the two vectors:
$$\begin{cases} \vec{u}^\uparrow = \langle 3, 4, 5, 10 \rangle \\ \vec{v}^\uparrow = \langle 3, 4, 6, 6 \rangle \end{cases}$$
- We lexicographically sort the ordered vectors : $\vec{u}^\uparrow \prec_{\text{lexico}} \vec{v}^\uparrow$



Usual Social Welfare Orderings

- Classical utilitarian order.
- Egalitarian order.
- **Leximin egalitarian order.**
- Compromises between classical utilitarianism and egalitarianism: Nash ($\times$), families OWA and sum of powers,...

Leximin egalitarianism [Sen, 1970; Kolm, 1972]

Let $\vec{x}$ be a vector. We write $\vec{x}^\uparrow$ the sorted version of $\vec{x}$.
 $\vec{u} \succ_{\text{leximin}} \vec{v} \Leftrightarrow \exists k$ such that $\forall i \leq k, u_i^\uparrow = v_i^\uparrow$ and $u_{k+1}^\uparrow > v_{k+1}^\uparrow$.

This is a lexicographical comparison over sorted vectors.

Features

This SWO both refines the egalitarian SWO and the Pareto relation $\sim$: it inherits of the fairness features of egalitarianism, while overcoming drowning effect.



Usual Social Welfare Orderings

- Classical utilitarian order.
- Egalitarian order.
- Leximin egalitarian order.
- Compromises between classical utilitarianism and egalitarianism: Nash ($\times$), families OWA and sum of powers,...

Leximin egalitarianism [Sen, 1970; Kolm, 1972]

Let $\vec{x}$ be a vector. We write $\vec{x}^\uparrow$ the sorted version of $\vec{x}$.
 $\vec{u} \succ_{\text{leximin}} \vec{v} \Leftrightarrow \exists k$ such that $\forall i \leq k, u_i^\uparrow = v_i^\uparrow$ and $u_{k+1}^\uparrow > v_{k+1}^\uparrow$.

This is a lexicographical comparison over sorted vectors.

Leximin SWO leximin and Pareto-efficiency

$\langle 1, 1, 1, 1 \rangle \prec \langle 1000, 1, 1000, 1000 \rangle$ (the second value of the two vectors is discriminating).



Usual Social Welfare Orderings

- Classical utilitarian order.
- Egalitarian order.
- Leximin egalitarian order.
- Compromises between classical utilitarianism and egalitarianism: Nash ($\times$), families OWA and sum of powers,...



Allocation and numerical preferences

Two aggregation levels:

$$\left. \begin{array}{ccc} w_1^1, \dots, w_{p_1}^1 & \xrightarrow{\oplus} & u_1 \\ \vdots & & \vdots \\ w_1^n, \dots, w_{p_n}^n & \xrightarrow{\oplus} & u_n \end{array} \right\} \xrightarrow{g} uc.$$

With most logical based languages, generalized additivity, OR or XOR bids, ...
... **NP**-complete problem, except for very simple cases [Bouveret et al., 2005].



With ordinal preferences

No interpersonal comparison of utilities: What can we do ?



With ordinal preferences

No interpersonal comparison of utilities: What can we do ?

- *scoring* procedures + additional assumptions for comparing sets of objects [Brams and King, 2005]

o_4	$\succ$	o_2	$\succ$	o_3	$\succ$	o_1
$\downarrow$		$\downarrow$		$\downarrow$		$\downarrow$
16		8		4		2



With ordinal preferences

No interpersonal comparison of utilities: What can we do ?

- *scoring* procedures + additional assumptions for comparing sets of objects [Brams and King, 2005]
- Pareto-efficiency

Agent 1 : $o_1 \succ o_2 \succ o_1 o_2 \succ \emptyset$ / **Agent 2 :** $o_2 \succ o_2 o_1 \succ \emptyset \succ o_1$

$(1 : o_2, 2 : o_1)$ Pareto-dominates $(1 : o_1, 2 : o_2)$



With ordinal preferences

No interpersonal comparison of utilities: What can we do ?

- *scoring* procedures + additional assumptions for comparing sets of objects [Brams and King, 2005]
- Pareto-efficiency
- Envy-freeness

Agent 1 : $o_1 \succ o_2 \succ o_1 o_2 \succ \emptyset$ / **Agent 2 :** $o_2 \succ o_2 o_1 \succ \emptyset \succ o_1$

$(1 : o_1, 2 : \emptyset)$ is envy-free whereas $(1 : o_1, 2 : o_2)$ is not.



Ordinal criteria: complexity

- **Pareto-efficiency** : in general easy! (give all the objects to one agent)



Ordinal criteria: complexity

- **Pareto-efficiency** : in general easy! (give all the objects to one agent)
- **Envy-freeness** :
 - **Alone** : easy, but not interesting!



Ordinal criteria: complexity

- **Pareto-efficiency** : in general easy! (give all the objects to one agent)
- **Envy-freeness** :
 - **Alone** : easy, but not interesting!
 - **Complete allocation** : in general **NP**-complete [Lipton et al., 2004].



Ordinal criteria: complexity

- **Pareto-efficiency** : in general easy! (give all the objects to one agent)
- **Envy-freeness** :
 - **Alone** : easy, but not interesting!
 - **Complete allocation** : in general **NP**-complete [Lipton et al., 2004].
 - **Pareto-efficiency** : in general Σ_P^2 -complete: dichotomous preferences, weighted goal bases [Bouveret and Lang, 2008], (generalized) additives [de Keijzer et al., 2009], etc.



Incomplete preferences

What can we do with incomplete ordinal preferences ?

→ **necessary** or **possible** envy-freeness (resp. Pareto-efficiency)

- **Objects:** $\mathcal{O} = \{a, b, c\}$.
- **Agents:**
 - **Agent 1 :** $bc \succ ab, a \succ c$, + monotonicity (ex : $abc \succ ab$) + transitivité
 - **Agent 2 :** $c \succ a, a \succ b$ + monotonicity + transitivity.



Incomplete preferences

What can we do with incomplete ordinal preferences ?

→ **necessary** or **possible** envy-freeness (resp. Pareto-efficiency)

- **Objects:** $\mathcal{O} = \{a, b, c\}$.
- **Agents:**
 - **Agent 1 :** $bc \succ ab, a \succ c$, + monotonicity (ex : $abc \succ ab$) + transitivité
 - **Agent 2 :** $c \succ a, a \succ b$ + monotonicity + transitivity.
- $\langle 1 : a, 2 : bc \rangle$ **necessarily** leads to **envy**
- $\langle 1 : b, 2 : ac \rangle$ is **possibly envy-free** but not **necessarily**.



Resource allocation and CP-nets

CP-nets are not very well-suited for resource allocation problems. . .

→ Use CI-nets instead

However, most reasoning tasks in CI-nets are very hard. . .

→ Use a tractable fragment of CI-nets: SCI-nets [Bouveret et al., 2010]



Outline

- 1 **Languages for compact preference representation**
- 2 **Logical and bidding languages**
 - Propositional logic
 - Bidding languages
- 3 **Graphical languages for ordinal preferences**
 - An introduction to graphical languages
 - Preferential independence
 - CP-nets
 - Outcome optimization with CP-nets
 - CP-nets: extensions and variants
- 4 **Graphical languages for cardinal preferences**
 - Generalized additivity
- 5 **Collective decision making**
 - Voting
 - MultiAgent Resource Allocation
- 6 **Conclusion**



Conclusion

- Graphical languages are a powerful way of expressing preferences in combinatorial domains.
- They are equipped with (sometimes) efficient algorithms for finding an optimal outcome.
- Many applications, such as
 - configuration problems (tour packages, web page configuration etc.)
 - planning
 - information retrieval
 - group decision making
 - distributed decision making / game theory