Representing and Reasoning About Preferences

Combintorics Meets Decision Theory

Mirosław Truszczyński
University of Kentucky

The Third Polish Combinatorial Conference
Będlewo, September 24-30, 2010

Preferences are ubiquitous

Just consider these statements ...

- I prefer fish to beef to chicken or pork (being indifferent about the last two)
- I prefer vacation in May or June to one in July or August
- I prefer aisle seat to window seat to middle seat
- A sedan is better than an SUV
- With beef, I prefer red wine to beer to water
- But with chicken, I prefer white wine to beer to red wine to water

Preferences are important!

People are self-interested

- We like some things better than others
- And often feel strongly about it — we *really* want what we like
- And we subject our decisions to our preferences

And so, *understanding* preferences is important!

Indeed, it ...

- Can help us make better decisions
- Can help us understand decisions made by others and so, interact with them
- Is essential for building artificial agents to assist our decision making or act on our behalf

We need to be able to ...

- Acquire preferences
- Represent preferences
- Aggregate preferences coming from multiple sources
- Choose best outcomes given preferences
- *Automate* all or at least some of it

And so, *understanding* preferences is important!

Indeed, it ...

- Can help us make better decisions
- Can help us understand decisions made by others and so, interact with them
- Is essential for building artificial agents to assist our decision making or act on our behalf

We need to be able to ...

- Acquire preferences
- Represent preferences
- Aggregate preferences coming from multiple sources
- Choose best outcomes given preferences
- *Automate* all or at least some of it

Now more formally

The basic setting

- A finite set, O, of *outcomes* or *configurations*
- A binary *preference* relation $\succeq$ on O
  - $o \succeq o'$ — o is at least as good as o'
- "Derived" relations: $\succ$ and $\approx$
  - ("strict preference") $o \succ o'$ — if $o \succeq o'$ and not $o' \succeq o$
  - ("indifference") $o \approx o'$ if $o \succeq o'$ and $o' \succeq o$

Key problems

- To extract $\succeq$
- To represent $\succeq$
- To compute optimal outcomes
- To compare outcomes

Now more formally

The basic setting

- A finite set, O, of *outcomes* or *configurations*
- A binary *preference* relation $\succeq$ on O
  - $o \succeq o'$ — o is at least as good as o'
- "Derived" relations: $\succ$ and $\approx$
  - ("strict preference") $o \succ o'$ — if $o \succeq o'$ and not $o' \succeq o$
  - ("indifference") $o \approx o'$ if $o \succeq o'$ and $o' \succeq o$

Key problems

- To extract $\succeq$
- To represent $\succeq$
- To compute optimal outcomes
- To compare outcomes

Preferences and utility

Standard approach used in decision theory

- Preferences via *utility* functions
- Pick a function $u : O \to [0, 1]$ (or all reals — not important)
- Define $o \succeq o'$ iff $u(o) \geq u(o')$

The proper setting

Lotteries over O

- A *lottery* — a probability distribution over O
  - a restaurant can be described by the lottery:
    - $[0.7 : \mathit{beef}, 0.2 : \mathit{vegetarian}, 0.1 : \mathit{fish}]$
    - $[0.3 : \mathit{beef}, 0.5 : \mathit{vegetarian}, 0.2 : \mathit{fish}]$
  - a gamble is described by payoffs and their probabilities: $[0.5 : \$2,000,000, 0.5 : \$0]$
- (Recursively) a *lottery* — a probability distribution on lotteries over O
- Preferences via *expected* utility
  - assuming $u(\mathit{beef}) = 9$, $u(\mathit{veg}) = 11$ and $u(\mathit{fish}) = 12$
  - $u(\mathit{first}) = 9.7$; $u(\mathit{second}) = 10.6$
- But a word of caution
  - a gamble between a payoff of \$2,000,000 and 0 with equal odds
  - a gamble with sure payoff of \$999,999

The power of the expected utility approach

Under rather natural desiderata on preferences on lotteries

- Every preference relation can be so described!!
- That is, for every preference $\preceq$ on lotteries satisfying these desiderata
- There is a utility function on atomic outcomes, such that the corresponding expected utility function characterize the preference $\preceq$ on lotteries

  von Neumann-Morgenstern, 1944

The power of the expected utility approach

The desiderata:

- Completeness For every lotteries ℓ, ℓ', $\ell \succ \ell'$ or $\ell' \succ \ell$ or $\ell \approx \ell'$
- Transitivity If $\ell \succeq \ell'$ and $\ell' \succeq \ell''$ then $\ell \succeq \ell''$
- Substitutability If $\ell \approx \ell'$, then
$[p : \ell, p_2 : \ell_2, \ldots, p_k : \ell_k] \approx [p : \ell', p_2 : \ell_2, \ldots, p_k : \ell_k]$
- Decomposability If for every $o \in O$, $P_\ell(o) = P_{\ell'}(o)$, then $\ell \approx \ell'$
- Monotonicity If $\ell \succ \ell'$ and $p > q$ then
$[p : \ell, (1-p) : \ell'] \succ [q : \ell, (1-q) : \ell']$
- Continuity If $\ell \succ \ell' \succ \ell''$, then there is $p \in [0, 1]$ such that
$\ell' \approx [p : \ell, (1-p) : \ell''].$

The power of expected utility approach

von Neumann-Morgenstern Theorem, 1944

- If a relation $\succeq$ on lotteries satisfies these axioms then there is a function $u : O \rightarrow [0, 1]$ such that
  - $u(\ell) \geq u(\ell')$ iff $\ell \succeq \ell'$
  - $u([p_1 : o_1, \dots, p_k : o_k]) = \sum_{i=1}^{k} p_i u_i(o_i)$

Elegant and intuitive

But not free of problems

- Large spaces of (basic) outcomes
- The number of different dinners one can "construct" in a restaurant already too large for anybody to have any utility function for it
- And this is just a toy example — imagine ordering a plane
- Building "correct" utility functions on basic outcomes
  - difficult, costly and time consuming
  - error prone
  - and so, often impractical
  - even when we disregard the lotteries
- Perhaps worth the effort in building medical decision support systems
- Rarely in "non life-critical" applications

Alternatives?

Qualitative approaches

- Approximating the preference relation based on a limited number of qualitative statements
- CP-nets
  - Conditional preference networks
  - *Ceteris paribus* (or all else being equal) networks
- Equally intuitive and supporting preference elicitation
- Give rise to interesting combinatorial, algorithm design and computational complexity questions

Conditional preference networks Boutilier, Brafman, Hoos, Poole 1999

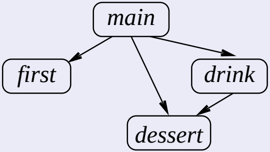
CP-net

- A structure to facilitate eliciting, representing and reasoning about preferences over *multivariable* domain
- The setting
  - A set V of variables $V = \{v_1, \ldots v_n\}$ with domains $D_1, \ldots, D_n$
- An outcome: a tuple $\langle a_1, \ldots, a_n \rangle$, where $a_i \in D_i$
- A dinner as an outcome
  - variables: first course, main course, drink and dessert
  - domains: $\{soup, salad\}$, $\{fish, beef\}$, $\ldots$
  - $\langle soup, fish, wine, cake \rangle$, $\langle salad, beef, beer, icecream \rangle$

CP-nets

Two key elements

- Dependency graph on variables



- Conditional preference tables
  Total order of values of a variable for all combinations of values of parent variables

Main
$b > f$

First	
$sp > sd$	b
$sd > sp$	f

Drink	
$br > wn$	b
$wn > br$	f

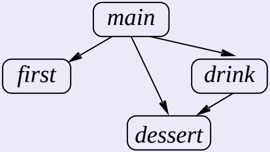
Dessert		
$c > ic$	b	br
$c > ic$	b	wn
$c > ic$	f	br
$ic > c$	f	wn

- **Preference elicitation made easier!!**

CP-nets

Two key elements

- Dependency graph on variables



- Conditional preference tables
  Total order of values of a variable for all combinations of values of parent variables

Main
$b > f$

First	
$sp > sd$	b
$sd > sp$	f

Drink	
$br > wn$	b
$wn > br$	f

Dessert		
$c > ic$	b	br
$c > ic$	b	wn
$c > ic$	f	br
$ic > c$	f	wn

- **Preference elicitation made easier!!**

CP-nets

Ceteris paribus — all else being equal

- With beef I prefer beer to wine
  - When making this statement, we assume *all else is equal*!!!
  - It allows to compare two dinners with beef
    identical except that one is with beer, the other with wine
- More generally
  - $\langle a_1, \ldots, a_i, \ldots, a_n \rangle$ is better than $\langle a_1, \ldots, a'_i, \ldots, a_n \rangle$
    if $a_i > a'_i$
    according to the appropriate row of the CPT for v_i,
  - worsening flip from $\langle a_1, \ldots, a_i, \ldots, a_n \rangle$ to $\langle a_1, \ldots, a'_i, \ldots, a_n \rangle$
  - improving flip from $\langle a_1, \ldots, a'_i, \ldots, a_n \rangle$ to $\langle a_1, \ldots, a_i, \ldots, a_n \rangle$

CP-nets

CP-net determined graph

- Vertices — outcomes
- (o, o') is an edge if there is an improving flip from o to o'

CP-net determined preference relation

- $o' \preceq o$ if there is a sequence of improving flips from o to o'
- Transitive closure of the graph determined by the CP-net
- The preference relation defined by the CP-net

Worsening flips: the dinner example

- $\langle sd, b, br, ic\rangle \succeq \langle sd, f, br, c\rangle \succeq \langle sd, f, br, ic\rangle$
- $\langle sp, b, br, ic\rangle \succeq \langle sd, b, br, ic\rangle \succeq \langle sd, f, br, ic\rangle$
- $\langle sp, b, br, c\rangle$ — an optimal dinner

CP-nets

CP-net determined graph

- Vertices — outcomes
- (o, o') is an edge if there is an improving flip from o to o'

CP-net determined preference relation

- $o' \preceq o$ if there is a sequence of improving flips from o to o'
- Transitive closure of the graph determined by the CP-net
- The preference relation defined by the CP-net

Worsening flips: the dinner example

- $\langle sd, b, br, ic\rangle \succeq \langle sd, f, br, c\rangle \succeq \langle sd, f, br, ic\rangle$
- $\langle sp, b, br, ic\rangle \succeq \langle sd, b, br, ic\rangle \succeq \langle sd, f, br, ic\rangle$
- $\langle sp, b, br, c\rangle$ — an optimal dinner

CP-nets

CP-net determined graph

- Vertices — outcomes
- (o, o') is an edge if there is an improving flip from o to o'

CP-net determined preference relation

- $o' \preceq o$ if there is a sequence of improving flips from o to o'
- Transitive closure of the graph determined by the CP-net
- The preference relation defined by the CP-net

Worsening flips: the dinner example

- $\langle sd, b, br, ic\rangle \succeq \langle sd, f, br, c\rangle \succeq \langle sd, f, br, ic\rangle$
- $\langle sp, b, br, ic\rangle \succeq \langle sd, b, br, ic\rangle \succeq \langle sd, f, br, ic\rangle$
- $\langle sp, b, br, c\rangle$ — an optimal dinner

Dominance and optimality

Dominance

- Outcome α dominates outcome β if there is a non-empty sequence of worsening flips from α to β
  - Slightly different from $\alpha \succeq \beta$
  - $\alpha \succeq \alpha$ always!
  - α dominates α only if α is on a non-empty cycle in the preference relation
- Outcome α strictly dominates outcome β if α dominates β but not the other way around
- DOMINANCE: given a CP-net and two outcomes α and β, decide whether α dominates β
- STRICT DOMINANCE – similarly

Optimality

- An outcome is optimal if it is not strictly dominated

Dominance and optimality

Dominance

- Outcome α dominates outcome β if there is a non-empty sequence of worsening flips from α to β
  - Slightly different from $\alpha \succeq \beta$
  - $\alpha \succeq \alpha$ always!
  - α dominates α only if α is on a non-empty cycle in the preference relation
- Outcome α strictly dominates outcome β if α dominates β but not the other way around
- DOMINANCE: given a CP-net and two outcomes α and β, decide whether α dominates β
- STRICT DOMINANCE – similarly

Optimality

- An outcome is optimal if it is not strictly dominated

Dominance and consistency

Consistency

- CP-net is consistent if no outcome dominates itself
  (no non-empty cycle of worsening flips)
- CONSISTENCY: given a CP-net, decide whether it is consistent

How hard are these problems?

For preference relations represented explicitly

- All problems easy – polynomial in the size of the representation
  - reduce to problems of reachability and belonging to cycles

But that misses the point!

- CP-net representation often is significantly more compact than the explicit one
  - If the number of parents bounded by a constant, the size is polynomial in the number of variables and the cardinality of the largest domain
- Under CP-net representation of global preference, the complexity of OPTIMALITY, DOMINANCE and CONSISTENCY no longer straightforward

How hard are these problems?

For preference relations represented explicitly

- All problems easy – polynomial in the size of the representation
  - reduce to problems of reachability and belonging to cycles

But that misses the point!

- CP-net representation often is significantly more compact than the explicit one
  - If the number of parents bounded by a constant, the size is polynomial in the number of variables and the cardinality of the largest domain
- Under CP-net representation of global preference, the complexity of OPTIMALITY, DOMINANCE and CONSISTENCY no longer straightforward

Some earlier results Boutilier, Brafman, Domshlak, Hoos and Poole, 2004

- OPTIMALITY for acyclic CP-nets is easy
  - a single sweep algorithm
  - arrange variables in topological order
  - proceeding according to this order
  - assign to each variable its optimal value given the values of its parent variables

- DOMINANCE is in P – for binary polytree CP-nets
- DOMINANCE is NP-complete – for binary directed-path singly connected CP-nets
- Hence: DOMINANCE is non-trivial already for binary acyclic CP-nets
- On the other hand: CONSISTENCY is assured for acyclic CP-nets
- Non-trivial in general

Some earlier results Boutilier, Brafman, Domshlak, Hoos and Poole, 2004

- OPTIMALITY for acyclic CP-nets is easy
  - a single sweep algorithm
  - arrange variables in topological order
  - proceeding according to this order
  - assign to each variable its optimal value given the values of its parent variables

- DOMINANCE is in P – for binary polytree CP-nets
- DOMINANCE is NP-complete – for binary directed-path singly connected CP-nets
- Hence: DOMINANCE is non-trivial already for binary acyclic CP-nets
- On the other hand: CONSISTENCY is assured for acyclic CP-nets
- Non-trivial in general

Dominance – special cases

Tree CP-nets

- The longest improving sequence bounded by $O(n^2)$
- There is an algorithm that decides dominance by constructing an improving sequence or "getting stuck" along they way
  - If a variable can be flipped when its descendants cannot be, flip it
  - Each time a leaf variable reaches its desired value, remove it
- There is a tree CP-net and two outcomes for which every improving sequence requires $\Theta(n^2)$ flips

Dominance – special cases

Polytree CP-nets

- Polytree – a dag whose underlying undirected graph is acyclic
- If we assume a fixed bound p on the number of parents – still in P
- Much more complex algorithm (adapted from a planning algorithm)
- The polynomial has p in the exponent
- But the longest “irreducible” improving sequence still bounded by $O(n^2)$

Dominance – special cases

Path-directed singly connected CP-nets

- The longest improving sequence still bounded by $O(n^2)$
- Thus, dominance is in NP
- And, is in fact NP-complete

General case of an acyclic CP-net

- In PSPACE but exact complexity not known
- There are acyclic CP-nets with outcomes between which exponential number of flips is necessary

Dominance – special cases

Path-directed singly connected CP-nets

- The longest improving sequence still bounded by $O(n^2)$
- Thus, dominance is in NP
- And, is in fact NP-complete

General case of an acyclic CP-net

- In PSPACE but exact complexity not known
- There are acyclic CP-nets with outcomes between which exponential number of flips is necessary

The general case?

Cyclic dependency nets make sense!

- If meat: red wine over white wine
  If fish: white wine over red wine
- If red wine: meat over fish
  If white wine: fish over meat
- A cycle in the dependency graph!

From now on: binary CP-nets

Each variable has a binary domain!

- If x is a variable, $\{x, \neg x\}$ — the domain of x.
- Leads to a “logical” representation of CP-nets

Generalized CPTs

- If x depends on $x_1, \ldots, x_k$
  - 2^k rows in the corresponding CPT
  - $x_1, \neg x_2, x_3, \ldots, \neg x_n$
  - in some rows $x > \neg x$; in others $\neg x > x$
  - CPT table for x can be represented by two propositional formulas over $\{x_1, \ldots, x_k\}$

From now on: binary CP-nets

Each variable has a binary domain!

- If x is a variable, $\{x, \neg x\}$ — the domain of x.
- Leads to a “logical” representation of CP-nets

Generalized CPTs

- If x depends on $x_1, \ldots, x_k$
  - 2^k rows in the corresponding CPT
  - $x_1, \neg x_2, x_3, \ldots, \neg x_n$
  - in some rows $x > \neg x$; in others $\neg x > x$
  - CPT table for x can be represented by two propositional formulas over $\{x_1, \ldots, x_k\}$

Generalized CPTs

Dessert		
$c > ic$	b	br
$c > ic$	b	wn
$c > ic$	f	br
$ic > c$	f	wn

- $\neg c$ for ic
- $\neg b$ for f
- $\neg wn$ for br
- $c > \neg c : b \vee (\neg b \wedge \neg wn)$
- $\neg c > c : \neg b \wedge wn$

- In fact just one of the formulas is enough
- But with two formulas further generalizations possible

Generalized form of CPTs

General CPT for a variable v

- A pair of formulas $(p^+(v), p^-(v))$
  - $p^+(v)$ – condition for $v > \neg v$
  - $p^-(v)$ – condition for $\neg v > v$
  - neither contains v or $\neg v$ (to exclude “self-dependencies”; not essential)

Generalized CP-nets (GCP-nets)

GCP-net

- A set V of binary variables
- A set of general CPTs: $\{(p^{+}(v), p^{-}(v)) : v \in V\}$

No dependency graph

All dependencies implicit in preference formulas

- w is a parent of v if w or $\neg w$ appears in $p^{+}(v)$ or $p^{-}(v)$

Flips, dominance, consistency

- As for CP-nets

Generalized CP-nets (GCP-nets)

GCP-net

- A set V of binary variables
- A set of general CPTs: $\{(p^+(v), p^-(v)) : v \in V\}$

No dependency graph

All dependencies implicit in preference formulas

- w is a parent of v if w or $\neg w$ appears in $p^+(v)$ or $p^-(v)$

Flips, dominance, consistency

- As for CP-nets

Generalized CP-nets (GCP-nets)

GCP-net

- A set V of binary variables
- A set of general CPTs: $\{(p^+(v), p^-(v)) : v \in V\}$

No dependency graph

All dependencies implicit in preference formulas

- w is a parent of v if w or $\neg w$ appears in $p^+(v)$ or $p^-(v)$

Flips, dominance, consistency

- As for CP-nets

GCP-nets: some subclasses

W/r to representation of conditional preferences

- Conjunctive: all formulas $p^+(v)$, $p^-(v)$ are in DNF
- Tabular: for every variable v, $p^+(v)$, $p^-(v)$ are DNF formulas whose every disjunct is full w/r to the set of parent variables of v

W/r to properties of conditional preferences

- Locally-consistent: all formulas $p^+(v) \wedge p^-(v)$ – consistent
- Complete: all formulas $p^+(v) \vee p^-(v)$ – tautologies
- CP-nets: GCP-nets that are locally consistent and complete

GCP-nets: some subclasses

W/r to representation of conditional preferences

- Conjunctive: all formulas $p^{+}(v)$, $p^{-}(v)$ are in DNF
- Tabular: for every variable v, $p^{+}(v)$, $p^{-}(v)$ are DNF formulas whose every disjunct is full w/r to the set of parent variables of v

W/r to properties of conditional preferences

- Locally-consistent: all formulas $p^{+}(v) \wedge p^{-}(v)$ – consistent
- Complete: all formulas $p^{+}(v) \vee p^{-}(v)$ – tautologies
- CP-nets: GCP-nets that are locally consistent and complete

GCP-nets

Comments on the size of representation

- Each next class offers, in general, exponentially more compact representations
  - Tabular GCP-nets
  - Conjunctive GCP-nets
  - GCP-nets
- Relevant for complexity studies

Complexity of DOMINANCE

GCP-DOMINANCE is PSPACE-complete

- Remains so under the restrictions to GCP-nets that are consistent and conjunctive
- Remains so under the restrictions to GCP-nets that are locally-consistent and conjunctive
- Remains so under the restriction to CP-nets
- Key difficulties:
  - proving PSPACE-completeness of GCP-DOMINANCE
  - dealing with the restriction to complete GCP-nets

Complexity of CONSISTENCY

GCP-CONSISTENCY is PSPACE-complete

- Remains so under the restriction to conjunctive GCP-nets
- Remains so under the restriction to CP-nets

reduction from locally-consistent GCP-nets to locally-consistent complete GCP-nets used for DOMINANCE preserves consistency

About proofs

Membership

- Standard

Hardness

- Exploit PSPACE-completeness of STRIPS planning problem
- Use it to establish the complexity of some restricted versions of propositional STRIPS planning problem
- Reduce to GCP-net reasoning problems

About proofs

Membership

- Standard

Hardness

- Exploit PSPACE-completeness of STRIPS planning problem
- Use it to establish the complexity of some restricted versions of propositional STRIPS planning problem
- Reduce to GCP-net reasoning problems

STRIPS PLAN is PSPACE-complete — Bylander, 1994

- V: set of propositional variables
- State over V: complete & consistent set of literals over V

  just like outcomes
- Action a: $(pre(a), post(a))$
  - $pre(a)$, $post(a)$: consistent conjunctions of literals
  - a leads from state s to state s' if
    - $s \not\models pre(a)$ and $s' = s$ (no change)
    - $s \models pre(a)$ and s' obtained from s by "flipping" literals whenever necessary to have $post(a) \subseteq s'$

  somewhat like flips
- Planning instance: $\langle V, ACT, s, g\rangle$
  - V: set of propositional variables, ACT: set of actions
  - s and g: initial and goal states
- STRIPS PLAN: given $\langle V, ACT, s, g\rangle$, is there a plan from s to g? (sequence of actions leading from s to g)

  somewhat like an improving sequence

Complexity of restricted STRIPS planning

Acyclic action sets

- *ACT* – acyclic: no non-trivial cycles in the state transition graph induced by *ACT*
- ACYCLIC STRIPS PLAN is PSPACE-complete
  - append states by counters
  - modify actions so that their execution increments the counter (in addition to their regular effect)
- ACTION SET ACYCLICITY is PSPACE-complete

ACTION SET ACYCLICITY **is PSPACE-complete**

Acyclicity through counters

- Assume n variables — 2^n states
- Add n fresh variables $\{z_1, \ldots, z_n\}$ to count from 0 to $2^n - 1$
  - $\neg z_1, \ldots, \neg z_n$ represents 0, and so on
  - z_1 and z_n represent most and least significant digits
- For each action a introduce actions a^i, $1 \leq i \leq n$, such that
  - $pre(a^i) = pre(a) \wedge \neg z_i \wedge z_{i+1} \wedge \ldots \wedge z_n$
  - $post(a^i) = post(a) \wedge z_i \wedge z_{i+1} \wedge \ldots \wedge z_n$
  - do what a does and increment counter by 1
- For each i introduce action b_i just for incrementing the counter

The new set of actions is *acyclic*!!

- There is a plan from s to g if and only if there is a plan from $s \neg z_1, \ldots, \neg z_n$ to $g z_1, \ldots, z_n$

ACTION SET ACYCLICITY is PSPACE-complete

Acyclicity through counters

- Assume n variables — 2^n states
- Add n fresh variables $\{z_1, \ldots, z_n\}$ to count from 0 to $2^n - 1$
  - $\neg z_1, \ldots, \neg z_n$ represents 0, and so on
  - z_1 and z_n represent most and least significant digits
- For each action a introduce actions a^i, $1 \leq i \leq n$, such that
  - $pre(a^i) = pre(a) \wedge \neg z_i \wedge z_{i+1} \wedge \ldots \wedge z_n$
  - $post(a^i) = post(a) \wedge z_i \wedge z_{i+1} \wedge \ldots \wedge z_n$
  - do what a does and increment counter by 1
- For each i introduce action b_i just for incrementing the counter

The new set of actions is *acyclic*!!

- There is a plan from s to g if and only if there is a plan from $s \neg z_1, \ldots, \neg z_n$ to $g z_1, \ldots, z_n$

Complexity of restricted STRIPS planning

Single-effect actions

- Postconditions – single literals
- Does not lower the complexity — still PSPACE-complete
- To simulate an action *a* with multiple-effects:
  - block other actions
  - enforce postcondition of *a* in a series of (new) single-effect actions;
  - unblock other actions

Single-effect actions

For each action a, a new variable x_a — a "in progress"

- X — no action in progress: $\bigwedge_a \neg x_a$
- X_a — a and only a in progress: $x_a \wedge \bigwedge_{b \neq a} \neg x_a$
- For each action a with $post(a) = l_1 \wedge \ldots \wedge l_q$
  - q actions a^i, $1 \leq i \leq q$
    - $pre(a^i) = pre(a) \wedge X \qquad post(a^i) = x_a$
  - q actions a^{q+i}, $1 \leq i \leq q$
    - $pre(a^{q+i}) = X_a \qquad post(a^{q+i}) = l_i$
  - one action a^{2q+1}
    - $pre(a^{2q+1}) = X_a \wedge post(a) \qquad post(a^{2q+1}) = \neg x_a$

Single-effect actions

Now all actions are single-effect ones

- There is a plan from s to g iff there is a plan from $s \wedge X$ to $g \wedge X$
- Acyclicity preserved!

And so ...

- SE ACYCLIC STRIPS and SE STRIPS are PSPACE-complete

Single-effect actions

Now all actions are single-effect ones

- There is a plan from s to g iff there is a plan from $s \wedge X$ to $g \wedge X$
- Acyclicity preserved!

And so ...

- SE ACYCLIC STRIPS and SE STRIPS are PSPACE-complete

From STRIPS planning to GCP-nets

STRIPS plan	GCP-net
states	outcomes
actions	generalized CPTs
preconditions	conjuncts in formulas $p^{+}(v)$ and $p^{-}(v)$
	"minus" postcondition
postcondition	better of v and $\neg v$
applying action	flipping
plan	flipping sequence
plan exists?	dominance?
set of actions acyclic?	GCP-net consistent?

And so ...

GCP-DOMINANCE is PSPACE-complete

- Even under restriction to
  - conjunctive and consistent GCP-nets
  - conjunctive and *locally* consistent GCP-nets

    consistency implies local consistency (no flipping back-and-forth)
- GCP-CONSISTENCY is PSPACE-complete
  - even under the restriction to conjunctive GCP-nets
- What about locally consistent and *complete* GCP-nets (that is, CP-nets)?

From GCP-nets to CP-nets

Reduction to complete locally-consistent GCP-nets

- Simulate a locally consistent GCP-net C by a locally consistent and complete one, say C'!

Variables

- $V = \{x_1, \ldots, x_n\}$ — variables of C
- $V' = V \cup \{y_1, \ldots, y_n\}$ — variables of C'

From GCP-nets to CP-nets

Reduction to complete locally-consistent GCP-nets

- Simulate a locally consistent GCP-net C by a locally consistent and complete one, say C'!

Variables

- $V = \{x_1, \ldots, x_n\}$ — variables of C
- $V' = V \cup \{y_1, \ldots, y_n\}$ — variables of C'

From GCP-nets to CP-nets

Preference conditions: q^+ and q^- for C'

- For each x_i:
- $q^+(x_i) = y_i$ and $q^-(x_i) = \neg y_i$
- For each y_i:
- $q^+(y_i) = f_i^+ \vee (\neg f_i^- \wedge x_i)$ and $q^-(y_i) = f_i^- \vee (\neg f_i^+ \wedge \neg x_i)$
  - $e_i = \bigwedge_{j \neq i}(x_j \leftrightarrow y_j)$
  - $f_i^+ = e_i \wedge p^+(x_i)$ and $f_i^- = e_i \wedge p^-(x_i)$
- Local consistency and completeness of C' evident
- C' is a CP-net

From GCP-nets to CP-nets

Some more notation

- Outcomes over $\{x_1, \ldots, x_n, y_1, \ldots, y_n\}$
- Written as concatenations $\alpha\beta$ of
  - outcomes α over $\{x_1, \ldots, x_n\}$ and
  - outcomes β over $\{y_1, \ldots, y_n\}$
- Let α be an outcome over $\{x_1, \ldots, x_n\}$
  - $x_i \mapsto y_i$
  - $\neg x_i \mapsto \neg y_i$
  - $\bar{\alpha}$

From GCP-nets to CP-nets

For a sequence $s = \alpha_0\alpha_1 \ldots \alpha_m$ of outcomes over V

- $L(s) = \alpha_0\bar{\alpha}_0\alpha_0\bar{\alpha}_1\alpha_1\bar{\alpha}_1 \ldots \alpha_m\bar{\alpha}_m$

For a sequence $t = \epsilon_0\epsilon_1 \ldots \epsilon_m$ of outcomes over V'

- Project each ϵ_i onto V
- Remove consecutive duplicate outcomes until no consecutive duplicates
- $L'(t)$

From GCP-nets to CP-nets

For a sequence $s = \alpha_0\alpha_1 \ldots \alpha_m$ of outcomes over V

- $L(s) = \alpha_0\bar{\alpha_0}\alpha_0\bar{\alpha_1}\alpha_1\bar{\alpha_1} \ldots \alpha_m\bar{\alpha_m}$

For a sequence $t = \epsilon_0\epsilon_1 \ldots \epsilon_m$ of outcomes over V'

- Project each ϵ_i onto V
- Remove consecutive duplicate outcomes until no consecutive duplicates
- $L'(t)$

From GCP-nets to CP-nets

With these definitions

- If s is an improving sequence from α to β in C, then $L(s)$ is an improving sequence from $\alpha\bar{\alpha}$ to $\beta\bar{\beta}$ in C'
- If t is an improving sequence from $\alpha\bar{\alpha}$ to $\beta\bar{\beta}$ in C', then $L'(t)$ is an improving sequence from α to β in C
- C is consistent if and only if C' is consistent

Thus

- Testing dominance in a locally consistent GCP C can be reduced to testing dominance in a CP-net C' (PSPACE-hardness of the latter follows)
- PSPACE-hardness of consistency of CP-nets follows in the same way

From GCP-nets to CP-nets

With these definitions

- If s is an improving sequence from α to β in C, then $L(s)$ is an improving sequence from $\alpha\bar{\alpha}$ to $\beta\bar{\beta}$ in C'
- If t is an improving sequence from $\alpha\bar{\alpha}$ to $\beta\bar{\beta}$ in C', then $L'(t)$ is an improving sequence from α to β in C
- C is consistent if and only if C' is consistent

Thus

- Testing dominance in a locally consistent GCP C can be reduced to testing dominance in a CP-net C' (PSPACE-hardness of the latter follows)
- PSPACE-hardness of consistency of CP-nets follows in the same way

Discussion

Our results extend to

- The class of GCP-nets with non-binary variables
- Dominance and consistency for preference theories (Wilson, 2004)

Open problems – complexity

- Conjunctive and tabular CP-net dominance and consistency?
- Conjunctive and tabular acyclic CP-net dominance
- Complexity of reasoning with GCP-nets, CP-nets and acyclic CP-nets whose variables have bounded number of parents

Discussion

Our results extend to

- The class of GCP-nets with non-binary variables
- Dominance and consistency for preference theories (Wilson, 2004)

Open problems – complexity

- Conjunctive and tabular CP-net dominance and consistency?
- Conjunctive and tabular acyclic CP-net dominance
- Complexity of reasoning with GCP-nets, CP-nets and acyclic CP-nets whose variables have bounded number of parents

Discussion

Problems on implicitly defined graphs

- The improving flip graph
- STRIPS planning graphs
- Upper bounds on flipping sequences needed to demonstrate dominance
- Lower bounds – showing what cannot be in general bested

Algorithms

- Find classes of GCP-nets (in particular classes of acyclic CP-nets) where efficient algorithms can be found

Discussion

Problems on implicitly defined graphs

- The improving flip graph
- STRIPS planning graphs
- Upper bounds on flipping sequences needed to demonstrate dominance
- Lower bounds – showing what cannot be in general bested

Algorithms

- Find classes of GCP-nets (in particular classes of acyclic CP-nets) where efficient algorithms can be found

References

- Carmel Domshlak, Ronen I. Brafman: CP-nets: Reasoning and Consistency Testing. KR 2002: 121-132
- Craig Boutilier, Ronen I. Brafman, Carmel Domshlak, Holger H. Hoos, David Poole: CP-nets: A Tool for Representing and Reasoning with Conditional Ceteris Paribus Preference Statements. J. Artif. Intell. Res. (JAIR) 21: 135-191 (2004)
- Judy Goldsmith, Jérôme Lang, Miroslaw Truszczynski, Nic Wilson: The Computational Complexity of Dominance and Consistency in CP-Nets. J. Artif. Intell. Res. (JAIR) 33: 403-432 (2008)