Preferences, Planning, and Control

Ronen I. Brafman
Department of Computer Science
Ben-Gurion University

September 18, 2008
1 Why preferences?
2 A graphical preference model
3 Planning and Preferences
   1 Planning with Preferences
   2 Graphical Models in Preference and Planning
   3 Preference Elicitation as Planning
4 Preferences and Control:
   Relational Preference Models
Collaborators

- Carmel Domshlak
- Craig Boutilier
- Yuri Chernyavsky
- Holger Hoos
- David Poole
- Yannis Dimopolous
Why Preferences?
Why Preferences?
The *Goal* Notion

**Goal concept**
- Central to classical planning
- Rigid: all or nothing
Why Preferences?
The *Goal* Notion

<table>
<thead>
<tr>
<th>Goal concept</th>
</tr>
</thead>
<tbody>
<tr>
<td>Central to classical planning</td>
</tr>
<tr>
<td>Rigid: all or nothing</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Goal concept inadequate:</th>
</tr>
</thead>
<tbody>
<tr>
<td>It is difficult to formulate a goal when you’re not familiar with a domain</td>
</tr>
<tr>
<td>Planning a vacation in a place you don’t know well</td>
</tr>
<tr>
<td>Information retrieval</td>
</tr>
<tr>
<td>Autonomous systems in uncertain environments can’t ask user for revised goals</td>
</tr>
</tbody>
</table>
Preference relations

- Convey more complete information about user objectives
- Can be (repeatedly) consulted when primary goal unachievable
Preference relations

- Convey more complete information about user objectives
- Can be (repeatedly) consulted when primary goal unachievable

A *preference* relation over a set $\Omega$ is a transitive binary relation $\succeq$ over $\Omega$. If for every $o, o' \in \Omega$ either $o \succeq o'$ or $o' \succeq o$ then $\succeq$ is a *total* order. Otherwise, it is a *partial* order.
Preferences are simple to specify if:

- Single objective with natural order
  - Minimize cost
  - Maximize quality

- Very small set of simple alternatives

  Marriott ≻ Best-Western ≻ Student Housing ≻ A bench across the opera house
Easy to Understand; Hard to Get.
Preference Specification is Difficult!

Preferences are simple to specify if:

- Single objective with natural order
  - Minimize cost
  - Maximize quality
- Very small set of simple alternatives
  - Marriott ≻ Best-Western ≻ Student Housing ≻ A bench across the opera house

Preferences are difficult to specify if:

- Multiple objectives
  - Minimize cost and maximize quality ⇒ Complicated tradeoffs
- Large set of alternatives
  - Hundreds of MP3 players
Preference Languages

Basic assumption
Outcomes/alternatives are structured – have attributes

Allow users to describe preference order *implicitly*
- Users provide *preference statements*
- Languages that mimic natural language utterances, make it easier to receive information from users
- Statements interpreted as partial order over set of alternatives
Preference Elicitation

Elicitation Techniques

- Limit amount of explicit information provided by user
- Reduce user’s cognitive burden (fewer, simpler questions)
- Domain knowledge + previous input $\Rightarrow$ focused questions
Why Preferences?

Summary

- Many applications call for replacing goals with preferences
- Explicit preference relationships are hard to construct and obtain
- Preference languages help users implicitly express a preference order using natural statements
- Preference elicitation technique focus user’s effort on most relevant preference information
CP-nets: A Graphical Preference Model
**Preference expression**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$</td>
<td>I prefer minivans to SUVs</td>
</tr>
<tr>
<td>$s_2$</td>
<td>In minivans, I prefer red exterior to white</td>
</tr>
<tr>
<td>$s_3$</td>
<td>In SUVs, I prefer white exterior to red</td>
</tr>
<tr>
<td>$s_4$</td>
<td>In white cars, I prefer dark interior to bright</td>
</tr>
<tr>
<td>$s_5$</td>
<td>In red cars, I prefer bright interior to dark</td>
</tr>
</tbody>
</table>

**Outcome space**

<table>
<thead>
<tr>
<th></th>
<th>category</th>
<th>ext-color</th>
<th>int-color</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>minivan</td>
<td>red</td>
<td>bright</td>
</tr>
<tr>
<td>$t_2$</td>
<td>minivan</td>
<td>red</td>
<td>dark</td>
</tr>
<tr>
<td>$t_3$</td>
<td>minivan</td>
<td>white</td>
<td>bright</td>
</tr>
<tr>
<td>$t_4$</td>
<td>minivan</td>
<td>white</td>
<td>dark</td>
</tr>
<tr>
<td>$t_5$</td>
<td>SUV</td>
<td>red</td>
<td>bright</td>
</tr>
<tr>
<td>$t_6$</td>
<td>SUV</td>
<td>red</td>
<td>dark</td>
</tr>
<tr>
<td>$t_7$</td>
<td>SUV</td>
<td>white</td>
<td>bright</td>
</tr>
<tr>
<td>$t_8$</td>
<td>SUV</td>
<td>white</td>
<td>dark</td>
</tr>
</tbody>
</table>

**CP-net**

- $C_{mv} \succ C_{sv}$
- $C_{mv} \succ E_r \succ E_w$
- $E_r \succ I_b \succ I_d$
- $E_w \succ I_d \succ I_b$

**Preference order**

1. $t_1 
2. t_2 \rightarrow t_4 \rightarrow t_8 \rightarrow t_6
3. t_3 \rightarrow t_7 \rightarrow t_5
4. t_2 \rightarrow t_4 \rightarrow t_8 \rightarrow t_6
5. t_3 \rightarrow t_7 \rightarrow t_5
6. t_2 \rightarrow t_4 \rightarrow t_8 \rightarrow t_6
7. t_3 \rightarrow t_7 \rightarrow t_5
8. t_2 \rightarrow t_4 \rightarrow t_8 \rightarrow t_6
9. t_3 \rightarrow t_7 \rightarrow t_5
10. t_2 \rightarrow t_4 \rightarrow t_8 \rightarrow t_6
11. t_3 \rightarrow t_7 \rightarrow t_5
12. t_2 \rightarrow t_4 \rightarrow t_8 \rightarrow t_6
13. t_3 \rightarrow t_7 \rightarrow t_5
14. t_2 \rightarrow t_4 \rightarrow t_8 \rightarrow t_6
15. t_3 \rightarrow t_7 \rightarrow t_5
16. t_2 \rightarrow t_4 \rightarrow t_8 \rightarrow t_6
17. t_3 \rightarrow t_7 \rightarrow t_5
18. t_2 \rightarrow t_4 \rightarrow t_8 \rightarrow t_6
19. t_3 \rightarrow t_7 \rightarrow t_5
20. t_2 \rightarrow t_4 \rightarrow t_8 \rightarrow t_6
21. t_3 \rightarrow t_7 \rightarrow t_5
What is the Graphical Representation Good For?
CP-nets

1. Convenient(?) input/elicitation tool
2. Convenient “map of independence”
3. Graph structure related to query processing complexity
4. Some algorithms utilize topological ordering over CP-net
Various queries given a set of preference statements $S$

<table>
<thead>
<tr>
<th>Verification</th>
<th>Does $S$ convey an ordering?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimization</td>
<td>Find $o \in \Omega$, such that $\forall o' \in \Omega : o', o$.</td>
</tr>
<tr>
<td>Comparison</td>
<td>Given $o, o' \in \Omega$, does $S \models o \succ o'$?</td>
</tr>
<tr>
<td>Sorting</td>
<td>Given $\Omega' \subseteq \Omega$, order $\Omega'$ consistently with $S$.</td>
</tr>
</tbody>
</table>
### Queries, Complexity, and Graphical Structure

| Various queries given a set of preference statements $S$ |
|-----------------|--------------------------------------------------|
| **Verification** | Does $S$ convey an ordering?                     |
|                  | - “YES” for acyclic CP-nets! [Boutilier et al. 2004] |
|                  | - Tractable for *certain* classes of cyclic CP-nets [B&Domshlak 2002] |
|                  | - PSPACE-hard in general [Goldsmith et al. 2005] |
| **Optimization** | Find $o \in \Omega$, such that $\forall o' \in \Omega : o' \not\succ o.$ |
| **Comparison**   | Given $o, o' \in \Omega$, does $S \models o \succ o'$? |
| **Sorting**      | Given $\Omega' \subseteq \Omega$, order $\Omega'$ consistently with $S$. |
### Various queries given a set of preference statements $S$

<table>
<thead>
<tr>
<th>Verification</th>
<th>Does $S$ convey an ordering?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimization</td>
<td>Find $o \in \Omega$, such that $\forall o' \in \Omega : o' \not\succ o$.</td>
</tr>
<tr>
<td></td>
<td><strong>Linear time</strong> for acyclic CP-nets.</td>
</tr>
<tr>
<td></td>
<td><strong>Tractable</strong> for <em>certain</em> classes of cyclic CP-nets</td>
</tr>
<tr>
<td>Comparison</td>
<td>Given $o, o' \in \Omega$, does $S \models o \succ o'$?</td>
</tr>
<tr>
<td>Sorting</td>
<td>Given $\Omega' \subseteq \Omega$, order $\Omega'$ consistently with $S$.</td>
</tr>
</tbody>
</table>
Pairwise Comparison: Given $o, o' \in \Omega$, does $S \models o \succ o'$?

[Boutilier,B,Domshlak,Hoos&Poole 2004][Goldsmith,Lang,Truszczynski&Wilson2005]

<table>
<thead>
<tr>
<th>Graph topology</th>
<th>Comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td>Directed Tree</td>
<td>$O(n^2)$</td>
</tr>
<tr>
<td>Polytree (indegree $\leq k$)</td>
<td>$O(2^{2k} n^{2k+3})$</td>
</tr>
<tr>
<td>Polytree</td>
<td>NP-complete</td>
</tr>
<tr>
<td>$\delta$-Connected</td>
<td>NP-complete</td>
</tr>
<tr>
<td>DAG</td>
<td>PSPACE-hard</td>
</tr>
<tr>
<td>General case</td>
<td>PSPACE-complete</td>
</tr>
</tbody>
</table>

Boolean variables

Multi-valued variables

NP-hard...
Various queries given a set of preference statements $S$:

**Verification** Does $S$ convey an ordering?

**Optimization** Find $o \in \Omega$, such that $\forall o' \in \Omega : o' \not\succ o$.

**Comparison** Given $o, o' \in \Omega$, does $S \models o \succ o'$?

**Sorting** Given $\Omega' \subseteq \Omega$, order $\Omega'$ consistently with $S$. 

Ronen I. Brafman

Preferences, Planning, and Control
Some Good News

Sorting is easy!

For *acyclic* CP-nets, sorting is doable in $O(n \log n)$ time
Summary

- **Language**: Conditional preferences over single attributes
  Summer ∧ Family: Eilat ≻ Jerusalem

- **Interpretation**:
  - Statements interpreted using *ceteris paribus* semantics
  - Statements combined via union and transitive closure

- **Representation**: Annotated, directed, graph
  - **Nodes**: Attributes
  - **Edges**: Direct dependency (condition → conditioned)
  - **Annotations**: Conditional preference tables (CPTs)

- **Model**: Partial order

- **Complexity**: Related to graph properties
Preferences & Planning: Planning with Goal Preferences
Preferences & Planning
1. Planning with Goal Preferences

Goal oriented planning $\Rightarrow$ preference-based planning

- Replace goal in domain description with a CP-net
- Find a plan for best feasible goal
Goal oriented planning $\Rightarrow$ preference-based planning

- Replace goal in domain description with a CP-net
- Find a plan for best feasible goal

Can we solve it effectively in practice?
Goal oriented planning $\Rightarrow$ preference-based planning

- Replace goal in domain description with a CP-net
- Find a plan for best feasible goal

Can we solve it effectively in practice?

For an acyclic CP-net — a qualified yes
Planning ⇒ CSP [Do&Kambhampati2001]
Planning with CP-nets ⇒ CSP + CP-nets
CSP + CP-net = (Discrete, qualitative) Constrained optimization
Constrained optimization = Find best (according to CP-net) feasible (according to CSP) solution
Constraint solver ⇒ Constraint optimizer
Preferences as constraints on solver
[Boutilier,B,Domshlak,Hoos&Poole2004]

Conceptually simple algorithm:

- Use your favorite DPLL/Tree-search-based CSP solver
- CP-net constrains variable/value orderings
- Parents must be assigned before children
- Preferred values must be assigned first
- First solution is optimal!
Solving Constrained Optimization

Constraints: $a \rightarrow \neg b$

Ronen I. Brafman
Preferences, Planning, and Control
Solving Constrained Optimization: Pruning

Constraints: $a \rightarrow \neg b$

$\begin{array}{c|c|c}
\text{true} & b \succ b \\
\hline
a \land b & c \succ c \\
\bar{a} \lor b & \bar{c} \succ c
\end{array}$
### Some Results

[B&Chernyavski05]

<table>
<thead>
<tr>
<th># scenario</th>
<th>time[sec.] (avg.)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BGP-CSP (time1)</td>
<td>Oracle (time2)</td>
</tr>
<tr>
<td>1</td>
<td>0.42</td>
<td>0.01</td>
</tr>
<tr>
<td>2</td>
<td>2.60</td>
<td>0.10</td>
</tr>
<tr>
<td>3</td>
<td>10.37</td>
<td>0.32</td>
</tr>
<tr>
<td>4</td>
<td>2.83</td>
<td>0.20</td>
</tr>
<tr>
<td>5</td>
<td>34.88</td>
<td>0.20</td>
</tr>
<tr>
<td>6</td>
<td>0.13</td>
<td>0.03</td>
</tr>
<tr>
<td>7</td>
<td>0.49</td>
<td>0.08</td>
</tr>
<tr>
<td>8</td>
<td>3.44</td>
<td>0.08</td>
</tr>
<tr>
<td>9</td>
<td>2.92</td>
<td>0.19</td>
</tr>
<tr>
<td>10</td>
<td>2.11</td>
<td>0.26</td>
</tr>
<tr>
<td>11</td>
<td>19.31</td>
<td>0.43</td>
</tr>
<tr>
<td>12</td>
<td>36.85</td>
<td>0.42</td>
</tr>
<tr>
<td>13</td>
<td>15.37</td>
<td>0.72</td>
</tr>
<tr>
<td>14</td>
<td>25.82</td>
<td>1.58</td>
</tr>
<tr>
<td>15</td>
<td>11.37</td>
<td>0.91</td>
</tr>
</tbody>
</table>

**Table:** Average plan search times
Preference with Planning as Satisfiability
[Giunchiglia & Maratea 2006]

Many others

- Oversubscription Planning [Smith 2004, van den Briel, Sanchez Nigenda, Do & Kambhampati 2004, Workshop, Competition,...]
- Qualitative preferences [Bienvenu, Fritz, & McIlraith 2006,...]
Planning with preferences can be solved as a constrained optimization problem. Given a CP-net, we can do constrained optimization using a CSP solver + ordering meta-constraints. Efficiency depends on the number of variables involved. With few goal variables, the method is quite efficient.
Preferences & Planning: Graphical Models
CP-nets query complexity

Complexity is related to simple properties of an intuitive graph

Planning complexity

Can we related planning complexity to simple properties of an intuitive graph?
Variables = Attributes

States = Outcomes

Actions = Rows in CPTs

Action transform an outcome to a less preferred outcome

**Cond**: \( X = x_i ≻ X = x_j \)  
**Precondition**: \( \text{Cond} \land X = x_i \);  
**Effect**: \( X = x_j \)

- \( a_1 \): **Pre**: category = minivan  
  **Eff**: category = SUV
- \( a_2 \): **Pre**: category = minivan \land ext = red  
  **Eff**: ext = white

\( o_1 ≻ o_2 \) \( \Rightarrow \) Is there a plan from \( o_1 \) to \( o_2 \)?
CP-net “actions” have unary effects.
- Planning with unary effects is as hard as regular planning
CP-net “actions” have special properties
- Actions are never reversible

Can we relate properties of a similar graphical structure for planning to plan generation complexity?
- CPT $\Rightarrow$ action
- CP-net $\Rightarrow$ ?
Causal Graphs

CP-nets ⇒ Causal Graph

- Nodes: state variables
- Edges: an edge from $x$ to $x'$ if $x$ is a precondition of an action that affects $x'$

Causal Graph for Logistics Domain
## Complexity of Planning with Unary Effects

<table>
<thead>
<tr>
<th>Graph topology</th>
<th>Plan Generation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Directed Tree</td>
<td>$O(n^2)$</td>
</tr>
<tr>
<td>Polytree (indegree $\leq k$)</td>
<td>$O(2^{2k}n^{2k+3})$</td>
</tr>
<tr>
<td>Polytree</td>
<td>NP-complete</td>
</tr>
<tr>
<td>$\delta$-Connected (indegree $\leq k$)</td>
<td>NP-complete</td>
</tr>
<tr>
<td>DAG</td>
<td>PSPACE-hard</td>
</tr>
<tr>
<td>General case</td>
<td>PSPACE-complete</td>
</tr>
</tbody>
</table>

## Multi-valued variables

NP-hard...[Domshlak&Dinitz2001][Chen&Gimenez2008]
Causal graph based analysis works!
- Same definition for general operators
- Complexity related to graph structure through tree-width
What about Standard Planning?

- Causal graph based analysis works!
  - Same definition for general operators
  - Complexity related to graph structure through tree-width

**Theorem** [B & Domshlak, 2007]
Given bounded-tree width, when short, “balanced” plans exist, they can be computed in polynomial time.
What else can we do with the causal graph?
Causal Graph and Complexity
What else can we do with the causal graph?

Tractable Substructures [Katz&Domshlak 2008]
Use tractable substructures as abstractions

CG(Π)

\{Π_{G_v^f}, Π_{G_v^{if}}\}_{v \in V}

CG(Π_{c_1}^f)

CG(Π_{p_1}^{if})

Ronen I. Brafman
Preferences, Planning, and Control
What else can we do with the causal graph?

**Tractable Substructures** [Katz & Domshlak 2008]
Use tractable substructures as abstractions

**Heuristics and Causal Graphs** [Helmert 2004, Geffner & Helmert 2008]
We can compute heuristic values by propagating information along the causal graph
Causal Graph and Complexity
What else can we do with the causal graph?

**Tractable Substructures [Katz&Domshlak 2008]**
Use tractable substructures as abstractions

**Heuristics and Causal Graphs [Helmert2004,Geffner&Helmert2008]**
We can compute heuristic values by propagating information along the causal graph

**Distributed Systems [B&Domshlak 2008]**
Agent-interaction graph generalizes causal graph in distributed systems
Non-Unary Operators

Non-unary operators in planning

Preferences over multiple attributes

Example

In England, I prefer Fish, Chips, and Beer to Veal, Potatoes, and Wine

Main Obstacle: Length

- Dominance queries are like plan existence queries
- In planning, we can focus on short plans
- In preference handling, we can’t
Research Challenge

From Planning to Complex Preferences

- Are there tractable classes of planning problems that induce tractable reasoning about complex preferences?
  - This is not only about CP-nets!
- Can we reason efficiently with preferences over more than one attribute?
Summary

Causal Graph—
Summary

- Causal Graph—
  - Planning complexity
  - Heuristics
  - CP-nets motivated its initial development
Planning for/with Elicitation
Elicitation as planning under uncertainty [Boutilier2002]

- **State**: user’s true preferences
  - We are uncertain about its value
- **Actions**: queries to user + “stop” (time to decide)
- **Cost & Reward**: 
  - Query cost depends on cognitive difficulty
  - “Stop”: *true* value of *perceived* best outcome
- **Planning problem**: maximize expected reward
- **Challenge**: trade of cost of queries with value of better knowledge of true preference model
- **Can be formulated precisely as a POMDP**
Elicitation as planning under uncertainty [Boutilier2002]

- **State:** user’s true preferences + environment state
  - We are uncertain about user’s preference
  - We may be uncertain about true world state

- **Actions:** queries to user + regular actions

- **Cost & Reward:**
  - Query cost depends on cognitive difficulty
  - Regular action’s value depend on user’s preference

- **Planning problem:** maximize expected reward

- **Challenge:** trade of cost of queries with value of better knowledge of true preference model

- Can be formulated precisely as a POMDP
Efficient Planning for Elicitation

POMDPs for elicitation have special structure

- Special state space
- Special actions
- Fixed state in the first case

Can we provide planning algorithms that reason effectively in this domain?
Preference elicitation can be formulated as, and integrated elegantly into, decision theoretic planning models.
Preferences & Control:
Relational Preference Models
Motivation: Command & Control GUI

Imagine controlling emergency forces in NYC
- Dynamic situation
  - New events (fire, injured, etc.)
  - Personal/equipment change (forces added/removed)
- Much information to monitor
  - Sensors on personal, equipment, buildings
- Much relevant information to access
  - Maps, building specs, simulations, ...

Our Goal:
- Proactively manage decision maker’s display
- Same system can work when personal and equipment change
- Same system can be used in New York and Sydney
How do We Solve This Problem?

1. Model this is as a decision-theoretic planning problem
   - Not there yet!
   - Dynamic universe, relational model, huge state space, huge action space, probabilities hard to assess

2. Let user provide an explicit policy
   - Users wouldn’t be able to handle this
   - Dynamic universe, relational model, huge state space, huge action space

3. Something in between: preferences over choices + optimization

Ronen I. Brafman
Preferences, Planning, and Control
## A Simple Relational Preference Language

### The language

1. A set of rules
2. Class definitions (possibly implicit)

### rule-body  →  rule-head : \(\langle (v_1, w_1), \ldots, (v_k, w_k) \rangle\)

- **rule-body**: \(\text{class}_1(x_1) \land \ldots \land \text{class}_k(x_k) \land \alpha_1 \land \ldots \land \alpha_m\)
  - \(\alpha_i\): \(x_i\).path REL value or \(x_i\).path_i REL \(x_j\).path_j.

- **rule-head**: \(x_j\).path
  - \(x_j\).path denotes a controllable attribute.
(1) fireman(x) ∧ fire(y) ∧ x.location = y.location
    → x.camera.display : 〈("on",4),("off",0)〉.

(2) fireman(x) ∧ fire(y) ∧ x.location = y.location
    → x.camera.display : 〈("on",4)〉.

(3) fireman(x) ∧ x.camera.display = "on"
    → x.rank : 〈("high",4)〉
Semantics

Preference Rules + DB of objects $\Rightarrow$ value function over the possible assignments to their controlable attributes.

A relational extension of GAI value functions.

$$v_{\mathcal{R}, \mathcal{O}}(\bar{a}) = \sum_{\text{ground instances } r' \text{ of } r \in \mathcal{R} \text{ satisfied by } \bar{a}} w(r, v(\bar{a}, r))$$

- $\mathcal{R}$: set of rules
- $\mathcal{O}$: set of objects
- $\bar{a}$: an assignment to their controlable attributes
- $w(r, v(\bar{a}, r))$: weight assigned in $r$ to its head given $\bar{a}$. 
Example

\[(1) \text{fireman}(x) \land \text{fire}(y) \land x.\text{location} = y.\text{location} \rightarrow x.\text{camera.display} : \langle(\"on\",4),(\"off\",0)\rangle.\]

Rule (1) + fireman(Alice), fireman(Bob), fire(Fire1)

- First instantiation: \(1a\) Alice.location = Fire1.location \(\rightarrow\)
  Alice.camera.display : \(\langle(\"on\",4),(\"off\",0)\rangle\)

- Second instantiation: \(1b\) Bob.location = Fire1.location \(\rightarrow\)
  Bob.camera.display : \(\langle(\"on\",4),(\"off\",0)\rangle\)
Methods

- Local search
- Branch & Bound
- Transform into probabilistic relational model:
  - Input (RPR): $b \rightarrow h\langle(v_1, w_1), \ldots, (v_k, w_k)\rangle$.
  - Output (ML): $\{b \land h = v_i : w_i\}$. 
**Preference Rules Vs. Rules**

**Rules are rigid**
- Context sensitivity must be built in explicitly
- Can be inconsistent

**Preference rules are flexible**
- Context sensitivity built in
- Inference replaced by optimization
- Easier to define approximations
To support complex applications, personalization, and autonomy, we need preference handling techniques.

Preference handling and planning have many synergies:

- Planning with preferences can be done using simple techniques that transform a solver into an optimizer.
- Strong relationship between CP-nets and causal graphs and complexity questions in preference and planning.
- Causal graphs have emerged as an interesting and useful representation of a planning domain.
- Preference elicitation can be formulated as a decision-theoretic planning problem.