Answer Set Optimization

Gerhard Brewka
brewka@informatik.uni-leipzig.de

Universität Leipzig
Motivation

preferences determine how agents decide and act
pop up everywhere:

- coffee > tea
- car > train
- relax > work
- Borussia Dortmund > Bayern München
- Madonna > Britney Spears
- marry > don’t marry
- sleep > listen to talk
Preferences in AI

- diagnosis: prefer more plausible hypotheses
- planning/configuration: prefer cheaper plan; satisfy more important constraints
- revision: give up less preferred beliefs
- reasoning about action: prefer fewer unexplained changes
- ontologies: prefer more specific information
- legal/deontic reasoning: prefer more recent law
- linguistics: prefer more important constraints (optimality theory)
Issues

- how to represent space of outcomes: often used: constraints; here: answer sets
- how to represent preferences: traditionally: numbers; here: qualitative numbers difficult to obtain; not always necessary
- how to interpret preferences: strict vs. defeasible; ceteris paribus
- how to represent (in)dependencies: preferences almost always context dependent
Outline

1. Motivation (done)
2. Answer set programming
3. Qualitative optimization
4. Applications
5. Related issues
6. Conclusions
2. Answer set programming
Answer sets

- define semantics for logic programs with strict and default negation (extended LPs)
- rules of the form \((a, b_i, c_j \text{ literals})\):
  
  \[
  a \leftarrow b_1, \ldots, b_n, \text{not } c_1, \ldots, \text{not } c_m
  \]

- AS acceptable set of beliefs based on program requirements:
  - closed: all rules used to generate AS if all \(b_i \in \text{AS}\), no \(c_j \in \text{AS}\), then \(a \in \text{AS}\)
  - grounded: \(a\) in AS implies derivation of \(a\) from rules whose not-preconditions are not in AS
A simple test

To check whether $S$ is AS of $P$

- remove $S$-defeated rules (not $L$ in body, $L \in S$)
- remove not literals from remaining rules
- check whether $S = \text{closure of reduced program}$
A simple test

To check whether $S$ is AS of $P$

- remove $S$-defeated rules (not $L$ in body, $L \in S$)
- remove not literals from remaining rules
- check whether $S = \text{closure of reduced program}$

\[
a \leftarrow \text{not } b \\
b \leftarrow \text{not } c
\]
A simple test

To check whether $S$ is AS of $P$

- remove $S$-defeated rules ($\text{not } L \text{ in body, } L \in S$)
- remove $\text{not}$ literals from remaining rules
- check whether $S = \text{closure of reduced program}$

\[
\begin{align*}
a & \leftarrow \text{not } b \\
b & \leftarrow \text{not } c
\end{align*}
\]

$\{a\}$

NO, not closed
A simple test

To check whether $S$ is AS of $P$

- remove $S$-defeated rules (not $L$ in body, $L \in S$)
- remove not literals from remaining rules
- check whether $S = \text{closure of reduced program}$

\[
\begin{align*}
a & \leftarrow \text{not } b \\
 b & \leftarrow \text{not } c \\
\end{align*}
\]

\{a, b\}

NO, not grounded
A simple test

To check whether $S$ is AS of $P$

- remove $S$-defeated rules (not $L$ in body, $L \in S$)
- remove not literals from remaining rules
- check whether $S = \text{closure of reduced program}$

\[
\begin{align*}
  a & \leftarrow \text{not } b \\
  b & \leftarrow \text{not } c
\end{align*}
\]

\{b\}

YES, grounded and closed
Example: graph coloring

Description of graph:

\[ node(v_1), ..., node(v_n), edge(v_i, v_j), ... \]

Generate: every node needs exactly 1 color

\[ \text{col}(X, r) \leftarrow \text{node}(X), \text{not} \text{col}(X, b), \text{not} \text{col}(X, g) \]
\[ \text{col}(X, b) \leftarrow \text{node}(X), \text{not} \text{col}(X, r), \text{not} \text{col}(X, g) \]
\[ \text{col}(X, g) \leftarrow \text{node}(X), \text{not} \text{col}(X, r), \text{not} \text{col}(X, b) \]

Test: linked nodes cannot have same color

\[ \leftarrow \text{edge}(X, Y), \text{col}(X, Z), \text{col}(Y, Z) \]

Each answer set describes a solution!
Example: graph coloring

Description of graph:

\[ \text{node}(v_1), \ldots, \text{node}(v_n), \text{edge}(v_i, v_j), \ldots \]

Generate: every node needs exactly 1 color

\begin{align*}
\text{col}(X, r) & \leftarrow \text{node}(X), \\text{not}\ \text{col}(X, b), \\text{not}\ \text{col}(X, g) \\
\text{col}(X, b) & \leftarrow \text{node}(X), \\text{not}\ \text{col}(X, r), \\text{not}\ \text{col}(X, g) \\
\text{col}(X, g) & \leftarrow \text{node}(X), \\text{not}\ \text{col}(X, r), \\text{not}\ \text{col}(X, b)
\end{align*}

Test: linked nodes cannot have same color

\[ f \leftarrow \text{edge}(X, Y), \text{col}(X, Z), \text{col}(Y, Z), \text{not}\ f \]

Each answer set describes a solution!
A useful language extension

Bounds on number of satisfied literals:

$L\{a_1,\ldots, a_k\}U$

Read: at least $L$ and at most $U$ of the $a_i$s must be true

Allows us to replace 3 color assignment rules with

$$1\{\text{col}(X, r), \text{col}(X, b), \text{col}(X, g)\}1 \leftarrow \text{node}(X)$$
Why LPs under AS semantics?

- simple yet expressive language
- transitive closure, nonmonotonic rules, ...
  
  \[ \text{flies}(X) \leftarrow \text{not } ab(X), \text{bird}(X) \]
- simple epistemic distinctions, particularly useful for preference reasoning
  
  \[ \text{safe} > \text{not } \neg \text{safe} > \neg \text{safe} \]
- interesting implementations: dlv, Smodels, nomore, ASSAT ...
- interesting applications: configuration, diagnosis, planning, reasoning about action, shuttle control, model checking, information integration, ...
3. Qualitative Optimization
### Adding preferences to ASP

<table>
<thead>
<tr>
<th></th>
<th>Rule preference</th>
<th>Formula preference</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Fixed</strong></td>
<td>((P, &lt;))</td>
<td>((P, &lt;))</td>
</tr>
<tr>
<td></td>
<td>(&lt;) order on (P)</td>
<td>(&lt;) order on (Lit)</td>
</tr>
<tr>
<td></td>
<td>B-Eiter</td>
<td>Sakama-Inoue</td>
</tr>
<tr>
<td></td>
<td>Delgrande-Schaub</td>
<td>Foo-Zhang</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td><strong>Conditional</strong></td>
<td>&lt; predicate in (P) applied to rules</td>
<td>ordered disjunction</td>
</tr>
<tr>
<td></td>
<td>B-Eiter</td>
<td>ASO programs</td>
</tr>
<tr>
<td></td>
<td>Delgrande-Schaub</td>
<td>B-Niemelä-Syrjänänen</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>B-N-Truszczynski</td>
</tr>
</tbody>
</table>
Ordered disjunction

LPOD: finite set of rules of the form:

\[ C_1 \times \ldots \times C_n \leftarrow \text{body} \]

if body then some \( C_j \) must be true, preferably \( C_1 \), if impossible then \( C_2 \), if impossible \( C_3 \), etc.

- answer sets defined through split programs
- satisfy rules to different degrees, depending on best satisfied head literal
- use degrees to define global preference relation on answer sets
- different options how to do this
Preferences among answer sets

How to generate global preference ordering from satisfaction degrees?

Many options, for instance:

\[ P^i(S) = P\text{-rules } i\text{-satisfied in } S. \]

\[ S_1 > S_2 \iff \]

- some rule has better satisfaction degree in \( S_1 \)
  and no rule better degree in \( S_2 \),

- at smallest degree \( i \) with \( P^i(S_1) \neq P^i(S_2) \),
  \( S_1 \) satisfies superset of rules satisfied in \( S_2 \),

- at smallest degree \( i \) with \( |P^i(S_1)| \neq |P^i(S_2)| \),
  \( S_1 \) satisfies more rules than \( S_2 \).
Prioritized graph coloring

\[ \text{col}(X, r) \times \text{col}(X, b) \times \text{col}(X, g) \leftarrow \text{node}(X) \]
\[ \leftarrow \text{col}(X, C), \text{col}(Y, C), \text{edge}(X, Y) \]

\( M \) preferred over \( M' \) if

- \( \text{par} \) at least 1 node has nicer color in \( M \) than in \( M' \), no node less preferred color.
- \( \text{incl} \) nodes red in \( M \) superset of nodes red in \( M' \), or same nodes red in \( M \) and \( M' \) and nodes blue in \( M \) superset of nodes blue in \( M' \).
- \( \text{card} \) more nodes red in \( M \) than in \( M' \), or as many nodes red in \( M \) as in \( M' \) and more blue in \( M \).
The ASO approach

decoupled approach to answer set optimization
- logic program $G$ generates answer sets
- preference program $P$ used to compare them
- preference program set of rules

$[C_1 > \ldots > C_k \leftarrow body]$

$C_i$ boolean combination built using $\lor, \land, \neg, \text{not}$

rule satisfaction and combination as for LPODs
LPODs vs. ASO

- **ASO:** arbitrary generating programs, no implicit generation of options, general preferences:
  combinations of properties preferred over others:
  \[ a > (b \land c) > d \leftarrow f \]
  equally preferred options:
  \[ a > (b \lor c) > \neg d \leftarrow g \]
- **LPODs:** compact and readable representations
4. Applications
Configuration

- often represented as AND/OR trees
- simple representation with Smodels cardinalities:
  
  \[
  4\{\text{starter, main, dessert, drink}\}4 \leftarrow \text{dinner}
  
  1\{\text{soup, salad}\}1 \leftarrow \text{starter}
  
  1\{\text{fish, beef, lasagne}\}1 \leftarrow \text{main}
  \]

- add case description and preferences, e.g.
  
  \[
  \text{fish} \lor \text{beef} > \text{lasagne}
  
  \text{beer} > \text{wine} \leftarrow \text{beef}
  
  \text{wine} > \text{beer} \leftarrow \text{not beef}
  \]

- preferred answer sets: optimal configurations
Abductive diagnosis

\[ H : \text{measles, flu, migraine} \]
\[ O : \text{headache, fever} \]
\[ K : \text{fever} \leftarrow \text{measles} \]
\[ \text{red-spots} \leftarrow \text{measles} \]
\[ \text{headache} \leftarrow \text{migraine} \]
\[ \text{nausea} \leftarrow \text{migraine} \]
\[ \text{fever} \leftarrow \text{flu} \]
\[ \text{headache} \leftarrow \text{flu} \]
Abductive diagnosis

\[ H : \text{measles, flu, migraine} \]
\[ O : \text{headache, fever} \]
\[ K : \text{fever} \leftarrow \text{measles} \]
\[ \text{red-spots} \leftarrow \text{measles} \]
\[ \text{headache} \leftarrow \text{migraine} \]
\[ \text{nausea} \leftarrow \text{migraine} \]
\[ \text{fever} \leftarrow \text{flu} \]
\[ \text{headache} \leftarrow \text{flu} \]
\[ \neg \text{measles} \times \text{measles} \quad \neg \text{flu} \times \text{flu} \]
\[ \neg \text{migraine} \times \text{migraine} \]
Abductive diagnosis

\[ H : \text{measles, flu, migraine} \]
\[ O : \text{headache, fever} \]
\[ K : \text{fever} \leftarrow \text{measles} \]
\[ \text{red-spots} \leftarrow \text{measles} \]
\[ \text{headache} \leftarrow \text{migraine} \]
\[ \text{nausea} \leftarrow \text{migraine} \]
\[ \text{fever} \leftarrow \text{flu} \]
\[ \text{headache} \leftarrow \text{flu} \]

\[ \neg \text{measles} \times \text{measles} \]
\[ \neg \text{flu} \times \text{flu} \]
\[ \neg \text{migraine} \times \text{migraine} \]

\[ \leftarrow \text{not headache} \]
\[ \leftarrow \text{not fever} \]
Abductive diagnosis

\[ H : \text{measles, flu, migraine} \]
\[ O : \text{headache, fever} \]
\[ K : \text{fever} \leftarrow \text{measles} \]
\[ \text{red-spots} \leftarrow \text{measles} \]
\[ \text{headache} \leftarrow \text{migraine} \]
\[ \text{nausea} \leftarrow \text{migraine} \]
\[ \text{fever} \leftarrow \text{flu} \]
\[ \text{headache} \leftarrow \text{flu} \]

\[ \neg\text{measles} \times \text{measles} \]
\[ \neg\text{flu} \times \text{flu} \]
\[ \neg\text{migraine} \times \text{migraine} \]

\[ \leftarrow \text{not headache} \]
\[ \leftarrow \text{not fever} \]

inclusion preferred: \{\text{migraine, measles}\}, \{\text{flu}\}
program $P$ describes normal behavior using $ab$-predicates

diagnosis minimal subset $C'$ of components $C$ such that

\[
\{ab(c) | c \in C'\} \cup \{\neg ab(c) | c \in C \setminus C'\}
\]

e.xplains observations $O$

playing $P_{cd}(P, C, O)$:

\[
P \cup \{\leftarrow \text{not } o | o \in O\} \cup \{\neg ab(c) \times ab(c) | c \in C\}
\]
Inconsistency handling

- program $P$, possibly inconsistent; consistency restoring rules $R$
- names $N_P$ and $N_R$ for rules in $P$ and $R$
- generate weakening of $P \cup R$ by replacing

\[
\text{head} \leftarrow \text{body} \quad \text{with} \quad \text{head} \leftarrow \text{body}, r_i
\]

where $r_i$ rule’s name

- add $\{r \times \neg r \mid r \in N_P\} \cup \{\neg r \times r \mid r \in N_R\}$
- minimal set of $P$-rules switched off, minimal set of $R$-rules switched on
Solution coherence

- assume solution $S$ for problem $P$ was computed
- problem changes slightly to $P'$
- not interested in arbitrary solution of $P'$, but solution as close as possible to $S$.
- distance measure based on symmetric difference:
  \[ A \Delta B = A \setminus B \cup B \setminus A \]
  \[ S_1 \leq_S S_2 \text{ iff } S_1 \Delta S \subseteq S_2 \Delta S \]
- corresponding preference program:
  \[ \{ a > \neg a \mid a \in S \} \cup \{ \neg a > a \mid a \notin S \} \]
Game theory

Prisoners’ dilemma

<table>
<thead>
<tr>
<th></th>
<th>Coop.</th>
<th>Defect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coop.</td>
<td>3,3</td>
<td>0,5</td>
</tr>
<tr>
<td>Defect</td>
<td>5,0</td>
<td>1,1</td>
</tr>
</tbody>
</table>
Game theory

Prisoners’ dilemma

<table>
<thead>
<tr>
<th></th>
<th>Coop.</th>
<th>Defect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coop.</td>
<td>3,3</td>
<td>0,5</td>
</tr>
<tr>
<td>Defect</td>
<td>5,0</td>
<td>1,1</td>
</tr>
</tbody>
</table>

Player 1: $D_1 \times C_1 \leftarrow C_2$

Player 2: $D_2 \times C_2 \leftarrow C_1$

$D_1 \times C_1 \leftarrow D_2$

$D_2 \times C_2 \leftarrow D_1$

Move clause: $1\{C_1, D_1\}1$
Game theory

Prisoners’ dilemma

<table>
<thead>
<tr>
<th></th>
<th>Coop.</th>
<th>Defect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coop.</td>
<td>3,3</td>
<td>0,5</td>
</tr>
<tr>
<td>Defect</td>
<td>5,0</td>
<td>1,1</td>
</tr>
</tbody>
</table>

Player 1: $D_1 \times C_1 \leftarrow C_2$

Player 2: $D_2 \times C_2 \leftarrow C_1$

$D_1 \times C_1 \leftarrow D_2$

Move clause: $1\{C_1, D_1\}$

Preferred answer set = Nash equilibrium 1,1
5. Related Issues
What else has been done?

- meta-preferences: one preference rule/ordered disjunction more important than another
- preference description language: combines different preference strategies; integrates qualitative with quantitative methods
- implementation: *generate and improve* method; iterative calls to answer set solver generate sequence of strictly improving answer sets
- integration with CP-nets: combines graph based methods with flexibility of ASO preferences
CP-nets

The problem:
- given variables $v_1, \ldots, v_n$, domains $D_1, \ldots, D_n$
- describe preference relation on value assignments

The solution:
- specify preference dependency graph
- specify total order of values given parent values
- use \textit{ceteris paribus} interpretation for preferences:

  \textit{red cars preferred over green cars =}
  \begin{align*}
    &\text{if 2 cars differ only in color,} \\
    &\text{then the red one is better than the green one}
  \end{align*}
Example

\[
\text{Main} \left\{ Steak, Fish \right\}
\]

\[
\begin{align*}
\text{Starter} & \quad \text{Drink} \\
\{ Soup, Salad \} & \quad \{ Red, White \}
\end{align*}
\]
Example

Steak \succ Fish

\begin{align*}
\text{Main} & : \{\text{Steak, Fish}\} \\
\text{Starter} & : \{\text{Soup, Salad}\} \\
\text{Drink} & : \{\text{Red, White}\}
\end{align*}

\begin{align*}
\text{Steak} : \text{Salad} \succ \text{Soup} & \quad \text{Steak} : \text{Red} \succ \text{White} \\
\text{Fish} : \text{Soup} \succ \text{Salad} & \quad \text{Fish} : \text{White} \succ \text{Red}
\end{align*}
Flips

- flip replaces value of a single variable
- flip improving: new value better according to relevant preference rule
- assignment $c_1$ better than $c_2$: there is a sequence of improving flips from $c_2$ to $c_1$.

Fish, Soup, White
\[\Downarrow\]
Steak, Soup, White
\[\Downarrow\]
Steak, Salad, White
\[\Downarrow\]
Steak, Salad, Red
Combining the approaches

- **ASO approach:** complex multi-criteria preferences, conflicting, incomplete, indifferent
  but: no explicit (in)dependencies

- **CP-nets:** structured preference representation and elicitation through explicit (in)dependencies
  but: restricted preferences among variable values

- to obtain the best of both worlds
  - identify variables with programs, their values with answer sets
  - use ASO multi-criteria preferences to specify preferences among answer sets
  - use CP-techniques to represent dependencies
Component systems at a glance

<table>
<thead>
<tr>
<th>CP-nets</th>
<th>Component systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>variables</td>
<td>programs</td>
</tr>
<tr>
<td>values</td>
<td>answer sets</td>
</tr>
<tr>
<td>value assignment</td>
<td>combination of answer sets</td>
</tr>
<tr>
<td>dependencies</td>
<td>dependencies</td>
</tr>
<tr>
<td>cond. pref. table</td>
<td>preference program</td>
</tr>
<tr>
<td></td>
<td>(+ language restrictions)</td>
</tr>
<tr>
<td>value flip</td>
<td>new answer set</td>
</tr>
</tbody>
</table>
6. Conclusions
What has been achieved?

- ASP a promising declarative paradigm
- simple yet expressive, interesting applications
- interesting answer set solvers
- adding preferences has great potential, conditional formula preferences very useful
- two related approaches presented
- a number of possible applications discussed
- combination with ideas from CP-nets leads to flexible framework for structured preference description and elicitation
What needs to be done?

- more refined implementation techniques
- better integration of qualitative and quantitative methods
- more convincing real world applications
  - trust negotiation (with P. Bonatti)
  - policy description languages (A. Mileo)
  - qualitative decision making (R. Grabos)