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 literals):

$$\boxed{a \leftarrow b_1, \ldots, b_n, \mathrm{not}\, c_1, \ldots, \mathrm{not}\, c_m}$$

- AS acceptable set of beliefs based on program
- requirements:
  closed: all rules used to generate AS
  if all $b_i \in$ AS, no $c_j \in$ AS, then $a \in$ AS
  grounded: a in AS implies derivation of a from rules whose not -preconds are not in AS

A simple test

To check whether S is AS of P

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

A simple test

To check whether S is AS of P

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

$$a \leftarrow \mathrm{not}\, b$$
$$b \leftarrow \mathrm{not}\, c$$

A simple test

To check whether S is AS of P

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

$$a \leftarrow \cancel{\mathrm{not}\, b}$$
$$b \leftarrow \cancel{\mathrm{not}\, c}$$

$$\{a\}$$

NO, not closed

A simple test

To check whether S is AS of P

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

$$\cancel{a \leftarrow \mathrm{not}\, b}$$
$$b \leftarrow \cancel{\mathrm{not}\, c}$$

$$\{a, b\}$$

NO, not grounded

A simple test

To check whether S is AS of P

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

$$\cancel{a \leftarrow \mathrm{not}\, b}$$
$$b \leftarrow \cancel{\mathrm{not}\, c}$$

$$\{b\}$$

YES, grounded and closed

Example: graph coloring

Description of graph:

$$node(v_1), ..., node(v_n), edge(v_i, v_j), \ldots$$

Generate: every node needs exactly 1 color

$$\begin{aligned}
col(X,r) &\leftarrow node(X), \mathrm{not}\, col(X,b), \mathrm{not}\, col(X,g) \\
col(X,b) &\leftarrow node(X), \mathrm{not}\, col(X,r), \mathrm{not}\, col(X,g) \\
col(X,g) &\leftarrow node(X), \mathrm{not}\, col(X,r), \mathrm{not}\, col(X,b)
\end{aligned}$$

Test: linked nodes cannot have same color

$$\leftarrow edge(X,Y), col(X,Z), col(Y,Z)$$

Each answer set describes a solution!

Example: graph coloring

Description of graph:

$$node(v_1), ..., node(v_n), edge(v_i, v_j), \ldots$$

Generate: every node needs exactly 1 color

$$\begin{aligned}
col(X, r) &\leftarrow node(X), \mathrm{not}\ col(X, b), \mathrm{not}\ col(X, g) \\
col(X, b) &\leftarrow node(X), \mathrm{not}\ col(X, r), \mathrm{not}\ col(X, g) \\
col(X, g) &\leftarrow node(X), \mathrm{not}\ col(X, r), \mathrm{not}\ col(X, b)
\end{aligned}$$

Test: linked nodes cannot have same color

$$f \leftarrow edge(X, Y), col(X, Z), col(Y, Z), \mathrm{not}\ f$$

Each answer set describes a solution!

A useful language extension

Bounds on number of satisfied literals:

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

Read: at least L and at most U of the a_is must be true

Allows us to replace 3 color assignment rules with

$$1\{col(X,r), col(X,b), col(X,g)\}1 \leftarrow node(X)$$

Why LPs under AS semantics?

- simple yet expressive language
- transitive closure, nonmonotonic rules, ...

$$flies(X) \leftarrow \mathrm{not}\, ab(X), bird(X)$$

- simple epistemic distinctions, particularly useful for preference reasoning

$$safe > \mathrm{not}\, \neg safe > \neg safe$$

- 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

	rule preference	formula preference
fixed	$(P, <)$ $<$ order on P B-Eiter Delgrande-Schaub ...	$(P, <)$ $<$ order on Lit Sakama-Inoue Foo-Zhang ...
conditional	$<$ predicate in P applied to rules B-Eiter Delgrande-Schaub ...	ordered disjunction ASO programs B-Niemelä-Syrjänen B-N-Truszczyński

Ordered disjunction

LPOD: finite set of rules of the form:

$$\boxed{C_1 \times \ldots \times C_n \leftarrow body}$$

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-rules i-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

$$col(X,r) \times col(X,b) \times col(X,g) \leftarrow node(X)$$
$$\leftarrow col(X,C), col(Y,C), edge(X,Y)$$

M preferred over M' if

- par at least 1 node has nicer color in M than in M', no node less preferred color.
- $incl$ 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'.
- $card$ 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

$$\boxed{C_1 > \ldots > C_k \leftarrow \mathit{body}}$$

  C_i boolean combination built using $\vee$, $\wedge$, $\neg$, 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:

  $$a > (b \wedge c) > d \leftarrow f$$

  equally preferred options:

  $$a > (b \vee c) > \mathrm{not}\, d \leftarrow g$$

- LPODs: compact and readable representations

4. Applications

Configuration

- often represented as AND/OR trees
- simple representation with Smodels cardinalities:

$$\begin{aligned} 4\{starter, main, dessert, drink\}4 &\leftarrow dinner \\ 1\{soup, salad\}1 &\leftarrow starter \\ 1\{fish, beef, lasagne\}1 &\leftarrow main \end{aligned}$$

- add case description and preferences, e.g.

$$\begin{aligned} fish \vee beef > lasagne & \\ beer > wine &\leftarrow beef \\ wine > beer &\leftarrow \text{not}\, beef \end{aligned}$$

- preferred answer sets: optimal configurations

Abductive diagnosis

$$
\begin{array}{ll}
H: & measles, flu, migraine \\
O: & headache, fever \\
K: & fever \leftarrow measles \\
 & red\text{-}spots \leftarrow measles \\
 & headache \leftarrow migraine \\
 & nausea \leftarrow migraine \\
 & fever \leftarrow flu \\
 & headache \leftarrow flu
\end{array}
$$

Abductive diagnosis

$$
\begin{array}{ll}
H: & measles, flu, migraine \\
O: & headache, fever \\
K: & fever \leftarrow measles \\
 & red\text{-}spots \leftarrow measles \\
 & headache \leftarrow migraine \\
 & nausea \leftarrow migraine \\
 & fever \leftarrow flu \\
 & headache \leftarrow flu
\end{array}
$$

$\neg measles \times measles$ $\quad\quad \neg flu \times flu$

$\neg migraine \times migraine$

Abductive diagnosis

$$\begin{array}{ll} H: & measles, flu, migraine \\ O: & headache, fever \\ K: & fever \leftarrow measles \\ & red\text{-}spots \leftarrow measles \\ & headache \leftarrow migraine \\ & nausea \leftarrow migraine \\ & fever \leftarrow flu \\ & headache \leftarrow flu \end{array}$$

$\neg measles \times measles$ $\quad$ $\neg flu \times flu$

$\neg migraine \times migraine$

$\leftarrow \text{not}\, headache$ $\quad$ $\leftarrow \text{not}\, fever$

Abductive diagnosis

$$
\begin{array}{ll}
H: & measles, flu, migraine \\
O: & headache, fever \\
K: & fever \leftarrow measles \\
 & red\text{-}spots \leftarrow measles \\
 & headache \leftarrow migraine \\
 & nausea \leftarrow migraine \\
 & fever \leftarrow flu \\
 & headache \leftarrow flu
\end{array}
$$

$\neg measles \times measles$ $\qquad$ $\neg flu \times flu$

$\neg migraine \times migraine$

$\leftarrow \text{not}\, headache$ $\qquad$ $\leftarrow \text{not}\, fever$

inclusion preferred: $\{migraine, measles\}$, $\{flu\}$

Consistency based diagnosis

- program P describes normal behavior using ab-predicates
- diagnosis minimal subset C' of components C such that

$$\{ab(c) \mid c \in C'\} \cup \{\neg ab(c) \mid c \in C \setminus C'\}$$

explains observations O

- corresponding LPOD $P_{cd}(P, C, O)$:

$$P \cup \{\leftarrow \mathrm{not}\, o \mid o \in O\} \cup \{\neg ab(c) \times ab(c) \mid c \in C\}$$

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

$$\boxed{head \leftarrow body} \quad \text{with} \quad \boxed{head \leftarrow body, r_i}$$

  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: ($A \Delta B = A \setminus B \cup B \setminus A$)

$$S_1 \leq_S S_2 \textit{ iff } S_1 \Delta S \subseteq S_2 \Delta S$$

- corresponding preference program:

$$\{a > \mathrm{not}\, a \mid a \in S\} \cup \{\mathrm{not}\, a > a \mid a \notin S\}.$$

Game theory

Prisoners' dilemma

	Coop.	Defect
Coop.	3,3	0,5
Defect	5,0	1,1

Game theory

Prisoners' dilemma

	Coop.	Defect
Coop.	3,3	0,5
Defect	5,0	1,1

Player 1:

$$D_1 \times C_1 \leftarrow C_2$$
$$D_1 \times C_1 \leftarrow D_2$$

Player 2:

$$D_2 \times C_2 \leftarrow C_1$$
$$D_2 \times C_2 \leftarrow D_1$$

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

Game theory

Prisoners' dilemma

	Coop.	Defect
Coop.	3,3	0,5
Defect	5,0	1,1

Player 1:

$$D_1 \times C_1 \leftarrow C_2$$

$$D_1 \times C_1 \leftarrow D_2$$

Player 2:

$$D_2 \times C_2 \leftarrow C_1$$

$$D_2 \times C_2 \leftarrow D_1$$

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

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:

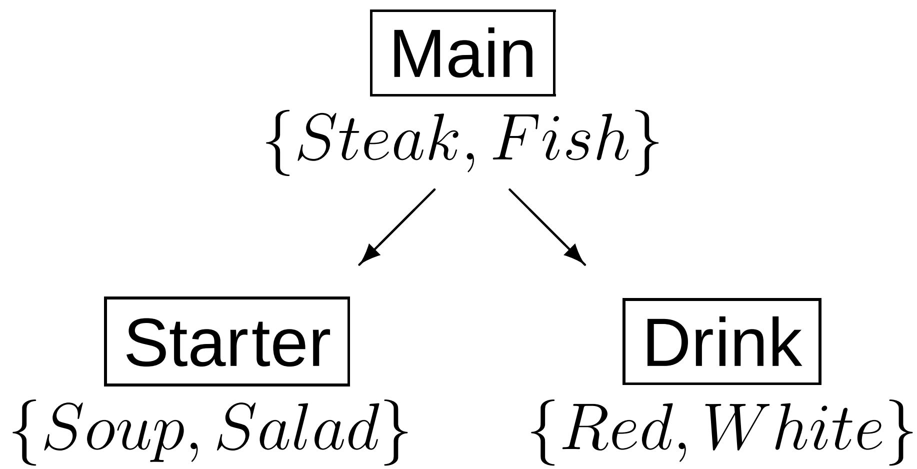
- given variables $v_1, \ldots, v_n$, domains $\mathcal{D}_1, \ldots, \mathcal{D}_n$
- describe preference relation on value assignments

The solution:

- specify preference dependency graph
- specify total order of values given parent values
- use *ceteris paribus* interpretation for preferences:

red cars preferred over green cars =
if 2 cars differ only in color,
then the red one is better than the green one

Example



Example

$$Steak > Fish$$

$$\begin{tikzcd} & \begin{array}{c}\boxed{\text{Main}} \\ \{Steak, Fish\}\end{array} \arrow[dl] \arrow[dr] & \\ \begin{array}{c}\boxed{\text{Starter}} \\ \{Soup, Salad\}\end{array} & & \begin{array}{c}\boxed{\text{Drink}} \\ \{Red, White\}\end{array} \end{tikzcd}$$

$$Steak : Salad > Soup$$
$$Fish : Soup > Salad$$

$$Steak : Red > White$$
$$Fish : White > Red$$

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.

$$\mathbf{Fish}, Soup, White$$
$$\Downarrow$$
$$Steak, \mathbf{Soup}, White$$
$$\Downarrow$$
$$Steak, Salad, \mathbf{White}$$
$$\Downarrow$$
$$Steak, Salad, Red$$

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

CP-nets	Component systems
variables	programs
values	answer sets
value assignment	combination of answer sets
dependencies	dependencies
cond. pref. table	preference program
	(+ language restrictions)
value flip	new answer set

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)