Answer Set Optimization

G. Brewka, I. Niemelä, M. Truszczyński

brewka@informatik.uni-leipzig.de

Universität Leipzig

Outline

1. Answer sets and answer set programming
2. Describing the quality of solutions
3. Optimization programs
4. Example: solution coherence in meeting scheduling
5. Conclusions

Why are AS interesting?

- provide meaning to logic programs with default negation not
- support problem solving paradigm where models (not theorems) represent solutions
- many interesting applications in planning, reasoning about action, configuration, diagnosis, space shuttle control, ...
- several useful extensions: disjunctive LPs, cardinality constraints, weight constraints ...
- interesting implementations: dlv, Smodels

Extended logic programs

Syntax of rules:

$$A \leftarrow B_1, \ldots, B_n, \mathrm{not}\, C_1, \ldots, \mathrm{not}\, C_m$$

where A, the B_i and the C_j are ground literals.

2 types of negation:

- classical negation $\neg$
- default negation not

Answer sets

S answer set of program P iff S is

- closed under P:
  $A \in S$ whenever
  $A \leftarrow B_1, \ldots, B_n, \text{not}\, C_1, \ldots, \text{not}\, C_m \in P$,
  $B_1, \ldots, B_n \in S$ and $C_1, \ldots, C_m \notin S$,
- logically closed:
  S consistent or equal to set of all literals.
- grounded in P:
  $A \in S$ implies there is a derivation for A from P based on rules whose not-Literals are not in S.

Good and bad solutions

- many problems have solutions of different quality
- basic ASP paradigm provides no distinction
- how to compare answer sets?
- quantitative measures, e.g.
  weights and maximize statements in $Smodels$,
  weak constraints in dlv
- here: qualitative measures based on preferences

Preference relations on AS

- different ways of adding preferences to LPs
- preferences between rules vs preferences between literals/formulas
- fixed vs. context dependent (the latter requires preference expressions within programs)
- here: context dependent preferences between literals/formulas

LPs with ordered disjunction

finite set of rules of the form:

$$C_1 \times \ldots \times C_n \leftarrow A_1, \ldots, A_m, \text{not } B_1, \ldots, \text{not } B_k$$

C_i, A_j, B_l ground literals.
if body then some C_j must be true, preferably C_1, if impossible then C_2, if impossible C_3, etc.

- Answer sets satisfy rules to different degrees.
- Use degrees to define global preference relation on answer sets.
- Different options how to do this (inclusion based, cardinality based etc.).

Optimization programs

- LPODs amalgamate generation of answer sets with quality assessment
- different types of programs available (disjunctive, cardinality constraints etc.)
- want more general preferences, possibly among unavailable options
- how to obtain more modularity and generality?
- use program P_{gen} to generate answer sets, preference program P_{pref} to compare them
- all we require is that P_{gen} generates sets of literals

Preference programs

Finite set of rules of the form

$$C_1 > \ldots > C_k \leftarrow a_1, \ldots, a_n, \mathrm{not}\, b_1, \ldots, \mathrm{not}\, b_m$$

a_i, b_j literals, C_i boolean combination:
built using $\vee$, $\wedge$, $\neg$, not .
$\neg$ in front of atoms, not in front of literals only.

additional expressiveness:
combinations of properties preferred over others:

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

equally preferred options:

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

Preference rule satisfaction

Consider $r = C_1 > \ldots > C_k \leftarrow \mathit{body}$.

For the degree of satisfaction $v_S(r)$ of r given set S of literals, there are three cases:

1. body not satisfied in S:
   r inapplicable thus *irrelevant*: $v_S(r) = I$
2. body satisfied and no C_i satisfied in S:
   rule specifies irrelevant preferences: $v_S(r) = I$
3. body satisfied and at least one C_i satisfied in S:
   $v_S(r) = \min\{i \colon S \models C_i\}$.

Satisfaction preorder

Views on irrelevance:

- I incomparable to other values, or
- I better than 2, 3, ... because no preference is violated

adopt latter view here:

$$\begin{array}{c} 1, I \\ | \\ 2 \\ | \\ \ldots \end{array}$$

Preference satisfaction ordering

$P_{pref} = \{r_1, \ldots, r_n\}$, AS S induces *satisfaction vector* $V_S = (v_S(r_1), \ldots, v_S(r_n))$.

Extend po on satisfaction degrees
to po on satisfaction vectors and answer sets:

S_1, S_2 answer sets.

$V_{S_1} \geq V_{S_2}$ if $v_{S_1}(r_i) \geq v_{S_2}(r_i)$, for all $i \in \{1, \ldots, n\}$.
$V_{S_1} > V_{S_2}$ if $V_{S_1} \geq V_{S_2}$ and not $V_{S_2} \geq V_{S_1}$.

$S_1 \geq S_2$ ($S_1 > S_2$) iff $V_{S_1} \geq V_{S_2}$ ($V_{S_1} > V_{S_2}$)

Meta preferences

- Preference rules themselves may be of different importance
- Put rules in subsets R_1, R_2, ... of decreasing importance
- Select answer sets most preferred according to R_1, among those answer sets most preferred according to R_2 etc.
- Allows for distinction among different criteria

Example: 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:

$$\{a > \text{not}\, a : a \in S\} \cup \{\text{not}\, a > a : a \notin S\}.$$

Meeting scheduling

$$\begin{array}{lll} part(p_1, m_1) & part(p_3, m_2) & unav(p_1, s_4) \\ part(p_2, m_1) & part(p_3, m_3) & unav(p_2, s_4) \\ part(p_2, m_2) & part(p_4, m_3) & unav(p_4, s_2) \end{array}$$

Meetings need 1 slot (using cardinality constraints):

$$1\{slot(M, S) : slot(S)\}1 \leftarrow meeting(M)$$

Constraints:

$$\begin{array}{ll} \leftarrow & part(P, M), slot(M, S), unav(P, S) \\ \leftarrow & part(P, M), part(P, M'), M \neq M', \\ & slot(M, S), slot(M', S) \end{array}$$

Meeting scheduling, ctd.

A solution: $slot(m_1, s_1), slot(m_2, s_2), slot(m_3, s_3)$

p_4 becomes unavailable at s_3: $unav(p_4, s_3)$

Preference rules:
$slot(m_1, s_1) > \text{not } slot(m_1, s_1)$,
$slot(m_2, s_2) > \text{not } slot(m_2, s_2), \ldots$

Former solution invalid. Some new solutions:

$$S_1 : slot(m_1, s_1), slot(m_2, s_2), slot(m_3, s_4)$$
$$S_2 : slot(m_1, s_2), slot(m_2, s_1), slot(m_3, s_4)$$
$$S_3 : slot(m_1, s_3), slot(m_2, s_2), slot(m_3, s_1)$$

inclusion based strategy: S_1 better than S_2.
cardinality based strategy: S_1 better than S_2 and S_3.

More stuff in the paper

- *complexity:*
  one extra layer of complexity, e.g.
  $\exists$ optimal AS S with $l \in S$? Σ_2^P-complete
  (extended LPs, possibly with cardinality or weight constraints)
- *implementation:*
  iterated improvement of current solution generated by tester program
- *relationship to CP-networks:*
  different interpretation of preferences: ceteris paribus vs. multi-criteria, theorems show CP-ordering can be approximated

Conclusion

- answer set programming: interesting declarative problem solving paradigm
- inclusion of optimization facilities increases applicability
- context dependent preferences among formulas flexible and powerful
- possible applications: configuration with weak constraints, diagnosis, planning, inconsistency handling ...
- future work: general optimization language for specifying qualitative preferences and optimization strategies