Complex Preferences for Answer Set Optimization

Gerhard Brewka
brewka@informatik.uni-leipzig.de

Universität Leipzig
Outline

1. Motivation
2. Optimization programs
3. The preference description language
4. Application and computation
5. Conclusions
Answer sets

- extend stable models (Gelfond, Lifschitz) to extended logic programs
- rules of the form \((a, b_i, c_j\) literals):
  \[ a \leftarrow b_1, \ldots, b_n, \neg c_1, \ldots, \neg c_m \]
- \(S\) answer set iff
  - closed: if all \(b_i \in S\), no \(c_j \in S\), then \(a \in S\)
  - grounded: \(a \in S\) implies non-circular derivation of \(a\) from rules whose negative preconds not in \(S\).
- problem solving style: answer set programming
The success of ASP

Main factors:

- availability of interesting implementations: dlv, Smodels, ASSAT ...
- shift of perspective from theorem proving to constraint programming/model generation
- many interesting applications in planning, reasoning about action, configuration, diagnosis, space shuttle control, ...

Natural next step: qualitative optimization brings in a lot of new interesting applications
Optimization programs

Basis for work presented here
(with I. Niemelä, M. Truszczynski, IJCAI-03)

• answer set generation independent of quality assessment

• $P_{gen}$ generates answer sets, preference program

• $P_{pref}$ compares them

• $P_{pref}$ uses rules of the form

$$C_1 > \ldots > C_k \leftarrow \text{body}$$

$C_i$ boolean combination built using $\lor$, $\land$, $\neg$, not .
$\neg$ in front of atoms, not in front of literals only.
ASO programs

\( P_{pref} \) used to select best answer set(s)

- answer sets satisfy rules to different degrees
- use degrees to define global preference relation on answer sets
- many options: inclusion based, cardinality based, Pareto, lexicographic, ...
- all potentially useful, may appear in combination
- \( \Rightarrow \) language that allows us to combine them
Preference description language

- allows us to combine preference strategies
- $PDL$ expression replaces $P_{pref}$ in $ASO$ programs
- $PDL$ consists of generalized preference rules and (possibly nested) expressions

$$(comb \ e_1 \ldots \ e_n)$$

where $comb$ is a combination strategy, $e_i$ an appropriate $PDL$ expression

- expressions induce preorder on answer sets
Generalized preference rules

\[ C_1:p_1 > \ldots > C_k:p_k \leftrightarrow \text{body} \]

\( C_i \) boolean combinations
\( p_i \) integer penalties satisfying \( p_i < p_j \) whenever \( i < j \).

\[ C_1 > C_2 > \ldots > C_k \leftrightarrow \text{body} \]

abbreviates
\[ C_1:0 > C_2:1 > \ldots > C_k:k-1 \leftrightarrow \text{body} \]
Syntax of PDL

\( PDL^p \) and \( PDL \) expressions:

1. \( r \) is preference rule \( \Rightarrow r \in PDL^p \),
2. \( e_1, \ldots, e_k \in PDL^p \Rightarrow (psum e_1 \ldots e_k) \in PDL^p \),
3. \( e \in PDL^p \Rightarrow e \in PDL \),
4. \( e_1, \ldots, e_k \in PDL^p \Rightarrow \)
   \( (inc e_1 \ldots e_k), (rinc e_1 \ldots e_k), (card e_1 \ldots e_k) \)
   and \( (rcard e_1 \ldots e_k) \in PDL \),
5. \( e_1, \ldots, e_k \in PDL \Rightarrow \)
   \( (pareto e_1 \ldots e_k) \) and \( (lex e_1 \ldots e_k) \in PDL \).
Penalties and rule semantics

1. \( \text{prex} = C_1:p_1 > \ldots > C_k:p_k \leftarrow \text{body} \)
   
   \( S \) satisfies \( \text{body} \) and at least one \( C_i: \)
   
   \( \text{pen}(S, \text{prex}) = p_j, \text{ where } j = \min \{ i \mid S \models C_i \} \),
   
   otherwise: \( \text{pen}(S, \text{prex}) = 0. \)

2. \( \text{prex} = (p\text{sum } e_1 \ldots e_k) \)
   
   \( \text{pen}(S, \text{prex}) = \sum_{i=1}^{k} \text{pen}(S, e_i). \)

3. \( \text{Ord}(\text{prex}) \) preorder associated with \( \text{prex} \), \( r \) rule:
   
   \( (S_1, S_2) \in \text{Ord}(r) \text{ iff } \text{pen}(S_1, r) \leq \text{pen}(S_2, r). \)
Complex expressions

$\geq_i$ preorder ($\geq_i$ partial order) represented by $e_i$, $i, j$ range over $\{1, \ldots, k\}$, $P^p_S = \{j \mid \text{pen}(S, e_j) = p\}$

- $(S_1, S_2) \in \text{Ord}(\text{pareto } e_1 \ldots e_k)$ iff $S_1 \geq_j S_2$ for all $j$.
- $(S_1, S_2) \in \text{Ord}(\text{lex } e_1 \ldots e_k)$ iff $S_1 \geq_j S_2$ for all $j$ or $S_1 >_j S_2$ for some $j$, and for all $i < j$: $S_1 \geq_i S_2$.
- $(S_1, S_2) \in \text{Ord}(\text{inc } e_1 \ldots e_k)$ iff $P^0_{S_1} \supseteq P^0_{S_2}$.
- $(S_1, S_2) \in \text{Ord}(\text{rinc } e_1 \ldots e_k)$ iff $\text{pen}(S_1, e_j) = \text{pen}(S_2, e_j)$ for all $j$ or $P^p_{S_1} \supseteq P^p_{S_2}$ for some $p$ and $P^q_{S_1} = P^q_{S_2}$ for $q < p$. 

Complex expressions, ctd.

- \((S_1, S_2) \in \text{Ord}(\text{card} \ e_1 \ldots e_k)\) iff 
  \[ |P^0_{S_1}| \geq |P^0_{S_2}|.\]

- \((S_1, S_2) \in \text{Ord}(\text{rcard} \ e_1 \ldots e_k)\) iff 
  \[ |P^p_{S_1}| = |P^p_{S_2}| \text{ for all } p \text{ or} \]
  \[ |P^p_{S_1}| > |P^p_{S_2}| \text{ for some } p, \text{ and} \]
  \[ |P^q_{S_1}| = |P^q_{S_2}| \text{ for all } q < p.\]

- \((S_1, S_2) \in \text{Ord}(\text{psum} \ e_1 \ldots e_k)\) iff 
  \[ \sum_{i=1}^{k} \text{pen}(S_1, o_i) \leq \sum_{i=1}^{k} \text{pen}(S_2, o_i).\]
Example

Assigning lecturers/time slots/rooms to courses:

\[
\begin{align*}
1 & \{ \text{teaches}(L, C) : \text{lecturer}(L) \} & \leftarrow & \text{course}(C) \\
1 & \{ \text{in}(R, C) : \text{room}(R) \} & \leftarrow & \text{course}(C) \\
1 & \{ \text{at}(S, C) : \text{slot}(S) \} & \leftarrow & \text{course}(C)
\end{align*}
\]

hard constraints: one course per lecturer, no clashes

\[
\begin{align*}
& \leftarrow \text{teaches}(L, C), \text{teaches}(L, C'), C \neq C' \\
& \leftarrow \text{in}(R, C), \text{in}(R, C'), \text{at}(S, C), \text{at}(S, C'), C \neq C'
\end{align*}
\]

lecturers’ preferences about courses, time slots, rooms, some of them more important than others
A possible preference model

lecturer $l_i$ specifies set $C_i$ of atoms $\text{teaches}(l_i, c) : p$
such that $\sum_{\text{teaches}(l_i,c):p} p = 10$.

lecturer $l_i$ specifies set $P_i$ of time and room
preferences, e.g.

$$am(S) > pm(S) \iff \text{teaches}(l_i, C), \text{at}(S, C)$$

$C_p$ union of all $C_i$ such that $l_i$ professor, $C_a$ union of
all $C_i$ such that $l_i$ assistant, $P_p$ and $P_a$ defined
similarly from $P_i$. A possible strategy:

$$(\text{lex} (psum C_p)(psum C_a)(\text{pareto} P_p)(\text{pareto} P_a))$$
Special cases

1. preference progs \( \{r_1, \ldots, r_k\} \): \((\text{pareto } r_1 \ldots r_k)\)

2. ranked preference progs:
   \((\text{lex } (\text{pareto } r_{1,1} \ldots r_{1,k_1}) \ldots (\text{pareto } r_{n,1} \ldots r_{n,k_n}))\)

3. cardinality and inclusion based combinations:
   use \(rinc\) and \(rcard\)

4. \(dlv\)’s weak constraints:
   \(\leftarrow \text{body. } [w]: \quad \text{use } \top:w \leftarrow \text{body with } psum\)
   \(\leftarrow \text{body. } [w:l]: \quad \text{group wrt. priority level } l:\)
   \((\text{lex } (psum \ r_{1,1} \ldots r_{1,k_1}) \ldots (psum \ r_{n,1} \ldots r_{n,k_n}))\)

5. minimize\(\{a_1 = w_1, \ldots, a_k = w_k\}\) statements:
   single statement: \((psum \ a_1:w_1 \ldots a_k:w_k)\)
   sequence: \((\text{lex}(psum \ldots ) \ldots (psum \ldots ))\)
Tester programs

- \( T(P, M, prex) \) based on generating program \( P \), current answer set \( M \), compilation of \( prex \)
- generates answer sets strictly better than \( M \)
- generate and improve optimization strategy
- compilation example (\( lex \ e_1 \ldots e_k \)):

\[
\begin{align*}
geq_i & \leftarrow geq_{i.1}, \ldots, geq_{i.k} \\
geq_i & \leftarrow better_i \\
better_i & \leftarrow better_{i.1} \\
better_i & \leftarrow geq_{1.1}, better_{i.2} \\
& \ldots \\
better_i & \leftarrow geq_{i.1}, \ldots geq_{i.k-1}, better_{i.k}
\end{align*}
\]
Conclusion

- ASP: successful problem solving paradigm
- optimization methods increase applicability: diagnosis, planning, inconsistency, configuration...
- context dependent preferences among formulas flexible and powerful
- developed a preference description language for specifying flexible optimization strategies
- future work: partially ordered goals, ASO methodology, combination with CP-net ideas