Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence

Tools for Preference Reasoning

Ying Zhu
University of Kentucky
Department of Computer Science
Lexington, KY 40504, USA
ying.zhu@uky.edu

Introduction and Background

The problem of computing similar and dissimilar solutions to a given one has received much attention in constraint satisfaction (Hebrard et al. 2005) and answer set programming (ASP) (Lifschitz 1999; Eiter et al. 2011; Marek and Truszczynski 1998). In many practical applications involving product configuration or planning, it is often the case that there are many valid solutions. To help the user see a small but representative sample, one needs algorithms that compute sets of dissimilar solutions. Once the user "zooms" in on one or two that she likes the most, it still makes sense to present several alternatives that are similar to the selected ones so that the user can find one that truly corresponds to her needs.

Preferences play an important role in AI applications especially those involving decision making (Kaci 2011). They are used to shrink the range of feasible solutions reflecting preferences specified by the user. Some concepts of optimality are relatively weak and still leave many optimal solutions to consider. Thus it makes sense to consider similar/dissimilar solutions also in preference formalisms. My work concerns the problem of computing similar and dissimilar optimal solutions in the preference formalism called answer set optimization (ASO) (Brewka, Niemela, and Truszczynski 2003). I am interested in the computational complexity of such problems, and in effective computing methods. In particular, I am interested in applications of ASP to represent preference formalisms and model preference problems, and in the use of ASP solvers to perform preference optimization.

Preferences can be modeled in quantitative and qualitative ways. In the quantitative way, preferences are combined with quantitative values to express the satisfaction or rejection levels, and the satisfaction degrees of solutions are expressed by numbers. In the qualitative way, preferences are partial or total orders on the values of some combination of variables. Thus the solutions are ordered by pairwise comparisons. ASO programs are the combination of answer set programming and qualitative optimization techniques. They have two parts, a program P_{gen} and a program P_{pref}. The role of a generating program is to produce possible solutions and the role of the preference program is to capture preferences that need to be optimized. The preference program P_{pref} consists of a set of preference rules in the form

$$C_1 > \cdots > C_k \leftarrow a_1, \ldots, a_n, \neg\, b_1, \ldots, \neg\, b_m$$

where a_is and b_is are literals and C_is are boolean combinations. Informally, this rule reads: if an answer set contains $a_1, \ldots, a_n$ and does not contain any of the literal $b_1, \ldots, b_m$, then C_1 is preferred over C_2, C_2 is preferred to C_3, etc. The satisfaction degree of an answer set S on a preference rule r is represented by $v_S(r) = min\{i : S \models C_i\}$. Given two answer sets S_1 and S_2, we say $S_1 \succeq S_2$ if $v_{S_1}(r) \leq v_{S_2}(r)$ for all rules, and $S_1 \succ S_2$ if $S_1 \succeq S_2$ and for some rule $v_{S_1}(r) < v_{S_2}(r)$. An answer set S is an optimal answer set if there is no answer set S' such that $S' \succ S$. To illustrate the ASO formalism, we consider a simple example. Let us assume P_{gen} is any theory generating 4 answer sets:

$$S_1 = \{soup, beef\},\ S_2 = \{salad, beef\},$$
$$S_3 = \{soup, fish\},\ S_4 = \{salad, fish\}.$$

For example, we can take for P_{gen} an answer set program:

$$1\{soup, salad\}1,\ 1\{beef, fish\}1,$$

or a propositional theory:

$$(soup \vee salad) \wedge (beef \vee fish) \wedge$$
$$\neg(soup \wedge salad) \wedge \neg(beef \wedge fish).$$

Assuming P_{pref} is:

$$soup > salad$$
$$beef > fish,$$

the satisfaction vectors for the 4 answer sets are $V_1 = (1, 1)$, $V_2 = (2, 1)$, $V_3 = (1, 2)$, $V_4 = (2, 2)$. Thus, S_1 is the optimal answer set, S_4 is the worst answer set, and S_2 and S_3 are incomparable.

Problem Statement

I consider the problems of finding a similar/dissimilar optimal solution to/from a given one. In each problem we assume that an ASO program P, an interpretation S, a distance measure Δ that maps two solutions for P to a nonnegative integer, and a nonnegative integer k is given.

k-SIMILAR OPTIMAL SOLUTION Decide whether there is an optimal answer set S' such that $\Delta(S, S') \leq k$.

k-DISSIMILAR OPTIMAL SOLUTION Decide whether there is an optimal answer set S' such that $\Delta(S, S') \geq k$.

Assuming that deciding whether $\Delta(S, S') \leq k$ for a given k is in P, the problems to find a similar/dissimilar optimal answer set are $\sum_2^p$-complete. The membership can be showed by guessing an interpretation S', verifying whether S' is an optimal answer set which has been proved is coNP-complete (Brewka, Niemela, and Truszczynski 2003), and checking whether $\Delta(S, S') \leq k$ in polynomial time. For the hardness, we construct a reduction from the problem to decide whether an optimal answer set including a given literal exists, which is $\sum_2^p$-complete (Brewka, Niemela, and Truszczynski 2003). The details depend on the definition of the distance function. The theorems we present below assume that the Hamming distance is used to measure how far from each other are the solutions.

Theorem 1. *Given an ASO program P, an interpretation S, and a nonnegative integer k, deciding whether there is an optimal answer set S' such that $HD(S, S') \leq k$ is $\sum_2^p$-complete.*

Theorem 2. *Given an ASO program P, an interpretation S, and a nonnegative integer k, deciding whether there is an optimal answer set S' such that $HD(S, S') \geq k$ is $\sum_2^p$-complete.*

Before talking about computing methods, I briefly introduce a method to find optimal answer sets in ASO programming. The method is based on answer set programming (ASP) (Lifschitz 1999; Marek and Truszczynski 1998). To get the optimal answer sets, we design a "tester" ASP program. It takes an answer set as input, and an ASP solver, such as CLASP (Gebser et al. 2007), applied to it generates a strictly better one if one exists. Thus, to find an optimal answer set, we can start from an arbitrary answer set, which can be found by an ASP solver, and use the tester iteratively, each time taking the output of last run as the input, until it fails. To find another optimal answer set, we first find an answer set which is not worse than the first optimal answer set, which also can be modeled as an ASP problem and solved by ASP solvers, and then use it as a start point to find an optimal answer set. To find all optimal answer set, we continue until it is no longer possible to find an answer set incomparable with the optimal answer sets found so far.

Solving the problem, the offline methods compute all optimal answer sets and store them. To compute k-similar solutions to a given interpretation U_0, the distances from U_0 to all optimal solutions are computed and the solutions whose distances from U_0 are $\leq k$ are reported. To compute k-dissimilar solutions, similarly, the outcomes whose distances from U_0 are $\geq k$ are reported.

Two important variants of these problems are to find a set of n optimal solutions, in which the distance between each pair of solutions is $\leq k$ or $\geq k$. Similarly, all optimal solutions and the distances between each pair of them are computed and some clustering methods are used to find the proper cluster. I do not discuss these problems here due to space limitation.

Conclusion and Future Work

So far I have studied the problem of computing similar or dissimilar optimal answer sets in ASO programming, analyzed its computational complexity, and introduced offline computing methods. My work so far demonstrates that ASP is a convenient tool to represent preference optimization problems and ASP solver a promising computational tool for the preference reasoning domain.

I am currently working on online computing methods which can find the similar/dissimilar optimal solution without computing all optimal solutions. The idea is to model the computation of an optimal answer set, the distance function, and the constraints on the distance into one ASP program. This program takes the given interpretation as input and gives a similar/dissimilar optimal answer set if it exists. The key point is to build a program which can find an optimal answer set in one call. To build that program, I am studying encodings of the problem as a quantified boolean formula (QBF). Once an appropriate QBF encoding is found, I will use standard methods to translate it further into a disjunctive logic program (Eiter and Gottlob 1995). That will allow me to use disjunctive ASP solvers such as claspD (Gebser et al. 2007) and DLV (Leone et al. 2006). However, I also intend to apply QBF solvers directly on the QBF encoding. Once all these techniques are well developed, I will perform extensive experiments to study their performance.

References

Brewka, G.; Niemela, I.; and Truszczynski, M. 2003. Answer set optimization. In *PROC. IJCAI-03*, 867–872. Morgan Kaufmann.

Eiter, T., and Gottlob, G. 1995. On the computational cost of disjunctive logic programming: Propositional case.

Eiter, T.; Erdem, E.; Erdogan, H.; and Fink, M. 2011. Finding similar/diverse solutions in answer set programming. *CoRR* abs/1108.3260.

Gebser, M.; Kaufmann, B.; Neumann, A.; and Schaub, T. 2007. T.: Conflict-driven answer set solving. In *In: Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07*, 386–392. MIT Press.

Hebrard, E.; Hnich, B.; O'Sullivan, B.; and Walsh, T. 2005. Finding diverse and similar solutions in constraint programming. In *Proceedings of the 20th national conference on Artificial intelligence - Volume 1*, AAAI'05, 372–377. AAAI Press.

Kaci, S. 2011. *Working with Preferences: Less Is More*. Cognitive Technologies. Springer.

Leone, N.; Pfeifer, G.; Faber, W.; Eiter, T.; Gottlob, G.; Perri, S.; and Scarcello, F. 2006. The dlv system for knowledge representation and reasoning. *ACM Trans. Comput. Logic* 7(3):499–562.

Lifschitz, V. 1999. Answer set planning. 23–37. The MIT Press.

Marek, V. W., and Truszczynski, M. 1998. Stable models and an alternative logic programming paradigm. *CoRR* cs.LO/9809032.