Computing Weighted Solutions
in Answer Set Programming

Duygu Çakmak, Esra Erdem, and Halit Erdoğan

Faculty of Engineering and Natural Sciences, Sabancı University, Istanbul, Turkey

Abstract. For some problems with many solutions, like planning and phylogeny reconstruction, one way to compute more desirable solutions is to assign weights to solutions, and then pick the ones whose weights are over (resp. below) a threshold. This paper studies computing weighted solutions to such problems in Answer Set Programming. We investigate two sorts of methods for computing weighted solutions: one suggests modifying the representation of the problem and the other suggests modifying the search procedure of the answer set solver. We show the applicability and the effectiveness of these methods in phylogeny reconstruction.

1 Introduction

In Answer Set Programming (ASP) [1, 8], a computational problem is described as an ASP program whose answer sets correspond to solutions, and answer sets for this program are computed using answer set solvers. Some problems, like planning and phylogeny reconstruction, have many solutions. Moreover, the correspondence between the answer sets and the solutions may not be one-to-one; there may be many answer sets that denote the same solution. For such problems, one way to compute more desirable solutions is to assign weights to solutions, and then pick the distinct solutions whose weights are over (resp. below) a threshold. For example, in a planning problem, we can define the weight of a plan in terms of the costs of actions (or action sequences), and then compute the distinct plans whose weights are less than a given value. In puzzle generation, we can define the weight of a puzzle instance by means of some difficulty measure, and then generate difficult puzzles whose weights are over a given value. Motivated by such applications, we study the problem of computing weighted solutions in ASP and show the applicability of our approach in phylogeny reconstruction (i.e., computing leaf-labeled trees, called phylogenies, to model the evolutionary history of a set of species).

We study two sorts of methods for computing weighted solutions: the representation-based methods and the search-based methods. In the former, the idea is to modify the ASP representation of the problem, to compute weighted solutions. In particular, we are interested in elaboration tolerant representations, where the weight of a solution is defined as an ASP program and added to the ASP representation of the problem. The latter, on the other hand, do not modify the ASP representation of the problem, but define the weight function externally (e.g., as a C++ program) and modify the search algorithm of the answer set solver to compute solutions over (resp. below) a given threshold. In this paper, we introduce such a search-based method for computing weighted solutions, and implement it by modifying the search algorithm of the answer set solver CLASP [7].

We apply these methods to phylogeny reconstruction. Reconstructing phylogenies for a given set of taxonomic units is important for various research such as historical linguistics, zoology, anthropology, archeology, etc.. For example, a phylogeny of parasites may help zoologists to understand the evolution of human diseases [4]; a phylogeny of languages may help scientists to better understand human migrations [11]. In this study, we define the weight of a phylogeny for the family of Indo-European languages studied in [2], in such a way as to reflect its plausibility and importance. Using this weight function, we show the applicability and effectiveness of the methods above (for computing weighted solutions) in reconstructing plausible phylogenies for Indo-European languages. No existing phylogenetic system has such utilities for experts to compute more plausible phylogenies, so our methods provide a useful tool for phylogenetics. Likewise, our methods provide a useful tool for various other applications of ASP.

2 Computing Weighted Solutions

We are interested in the following sorts of computational problems for computing weighted solutions:

> AT LEAST (resp. AT MOST) w-WEIGHTED SOLUTION: Given an ASP program $\mathcal{P}$ that formulates a computational problem P, a weight measure ω that maps a solution for P to a nonnegative integer, and a nonnegative integer w, decide whether a solution S exists for P such that $w(S) \geq w$ (resp. $w(S) \leq w$).

For instance, suppose that $\mathcal{P}$ describes the phylogeny reconstruction problem for Indo-European languages, and that ω describes the total weight of the characters compatible with the to-be-reconstructed phylogeny and takes into account some domain-specific information. Then finding phylogenies whose weights are at least 45 is an instance of the problem above.

We study two sorts of methods, representation-based and search-based, to compute at least/most w-weighted solutions in ASP.

The idea behind the representation-based methods is to modify the representation of the problem, to compute weighted solutions. For an elaboration tolerant representation $\mathcal{P}$, such modifications are done by means of adding some rules $\mathcal{W}$ describing the weight of a solution and some constraints $\mathcal{C}$ on the weights of the solutions (Fig. 1). Then we can compute at least (resp. most) w-weighted solutions by computing answer sets for the ASP program $\mathcal{P} \cup \mathcal{W} \cup \mathcal{C}$. In some cases, we do not have to define the weight of a solution explicitly; we can use aggregates (e.g., sum, count, times) to compute the weight, in the sense of [9, 6, 10]. Some other problems require an explicit definition of the weight of a solution. Phylogeny reconstruction problems we consider are in the latter group: the weight of a phylogeny does not only depend on the weights of some parts of the phylogeny but also some domain-specific information.

Search-based methods (as outlined in Fig. 2) on the other hand do not modify the ASP representation of the problem, but define the weight function externally (e.g., as a C++ program) and modify the search algorithm of the answer set solver to compute solutions over (resp. below) a given threshold. There is no answer set solver that can compute weighted solutions in such a way. Therefore, in our studies, we consider
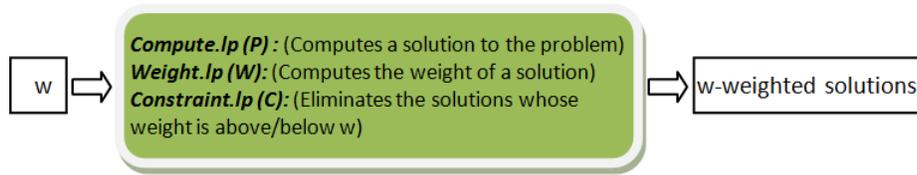
Fig. 1. Computing at most/least w-weighted solutions with representation-based method
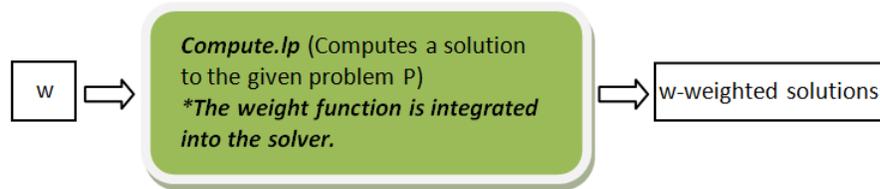

Fig. 2. Computing at most/least w-weighted solutions with search-based method

the solver CLASP [7], and modify its search algorithm to implement this search-based method. We call the new algorithm as CLASP-W. CLASP-W is shown in Algorithm 1: the parts in red denote the modifications we have made over CLASP's algorithm (i.e., if we remove the red parts, then we get CLASP's algorithm). Note that, compared to CLASP, CLASP-W has a new function called WEIGHT-ANALYZE. At each step of the search, CLASP has a partial solution and tries to complete it to find a solution to the given problem. The function WEIGHT-ANALYZE computes an upper bound (resp. lower bound) for the weight of a completion of the partial solution, so that CLASP-W does not perform redundant search towards a complete solution. WEIGHT-ANALYZE function is domain-specific; therefore, in order to use CLASP-W, we need to implement this function (in a separate file) according to the given weight measure for the particular problem. We do not need to modify CLASP-W for different problems.

3 Computing Weighted Phylogenies for Indo-European Languages

The evolutionary relations between species (or "taxonomic unit") based on their shared traits can be modeled as a phylogeny (or a phylogenetic tree). The problem of phylogeny reconstruction asks for "plausible" phylogenies for a given set of taxonomic units. There have been various studies to compute plausible phylogenies (see [2] for a discussion). In the following, we will consider a character-based cladistics with respect to the compatibility criterion, as in [2]. Our goal is to find a phylogeny with a small number of incompatible characters. The problem of reconstructing a phylogeny with at most k incompatible characters (let us call this problem as k-CP) is NP-hard [5].

While reconstructing phylogenies, some characters may give more information than the others. For instance, to model the evolutionary history of a family of languages, morphological/phonological characters are more informative than lexical characters. In

Algorithm 1 CLASP-W

Input: An ASP program Π and a nonnegative integer w
Output: An answer set for Π, that describes an at least (resp. at most) w-weighted solution

  $A \leftarrow \emptyset$ // current assignment of literals
  $\bigtriangledown \leftarrow \emptyset$ // set of conflicts
  while A does not represent an answer set **do**
    // propagate according to the current assignment and conflicts;update the current assignment
    NOGOOD-PROPAGATION($\Pi, A, \bigtriangledown$)
    // compute an upper (resp. lower) bound for the weight of a solution that contains A
    $weight \leftarrow$ WEIGHT-ANALYZE(A)
    // if the upper bound $weight$ is less than the desired weight value w
    // then no need to continue search to find an at least w-weighted solution
    if There is a conflict in unit-propagation OR $weight < w$ **then**
      RESOLVE-CONFLICT ($\Pi, A, \bigtriangledown$)   // learn and update the conflict set and do backtracking
    end if
    if Current assignment does not yield an answer set **then**
      SELECT($\Pi, A, \bigtriangledown$)   // select a literal to continue search
    else
      return A
    end if
  end while
  return false

---

order to emphasize the role of such characters in reconstructing a phylogeny, we define the concept of a weighted phylogeny.

A *weighted phylogeny* is a phylogeny along with a weight function Φ that maps every character $i \in I$ to a nonnegative integer. The *weight of a phylogeny* can be defined in various ways with respect to Φ; in the following, we consider the weight of a phylogeny as the sum of the weights of all characters that are compatible with that phylogeny.

With such a weight measure and an ASP program describing phylogeny reconstruction (like the one in [2, 3]), we can compute weighted phylogenies for the family of Indo-European languages described in [2], using the representation-based method or the search-based method described in Section 2.

To get more plausible phylogenies, we also incorporate further domain-specific information in the weight measure. It is told us by historical linguist Don Ringe that it is least likely that Greco-Armenian languages be siblings with Balto-Slavic languages. Similarly, but not as least likely as Greco-Armenian and Balto-Slavic, is the grouping of Greco-Armenian with Germanic languages. If the to-be-reconstructed phylogenies have such odd groupings of languages, we reduce some amount from the total weight of the phylogeny making sure that the weight of a phylogeny is not negative.

4   Experimental Results

We applied the computational methods described above (i.e., the representation-based method, and the search-based method) to reconstruct weighted phylogenies for Indo-

Table 1. The representation-based method vs. the search-based method: computing a phylogeny with at most c incompatible characters, and whose weight is at least w.

# of incompatible characters (c)	weight (w)	method	time (CPU sec.s)	program size	memory size (MB)
16	45	representation-based	15.52	# of atoms: 79229 # of rules: 1585419	369
		search-based	1.34	# of atoms: 3744 # of rules: 55219	22
17	45	representation-based	15.32	# of atoms: 79229 # of rules: 1585419	369
		search-based	1.30	# of atoms: 3744 # of rules: 55219	22
18	45	representation-based	15.47	# of atoms: 79229 # of rules: 1585419	369
		search-based	1.10	# of atoms: 3744 # of rules: 55219	22

European languages, as described in the previous section, with the dataset and the ASP program used in [2].

Let us consider computing an at least w-weighted phylogeny with at most c incompatible characters. In Table 1, for each problem, for each method, we present the computation time (CPU seconds), the size of the ground program (the number of atoms, and the number of rules), and the size of the memory (MB) used in computation.[1] For instance, let us consider computing a phylogeny with at most 17 incompatible characters, and whose weight is at least 45. With the representation-method, CLASP takes 15.32 CPU sec.s to compute such a phylogeny; the ground program has 79229 atoms and 1585419 rules; the computation of the phylogeny consumes 369 MB of memory. On the other hand, with the search-based method, CLASP-W takes 1.30 CPU sec.s to compute such a phylogeny; the ground program has 3744 atoms and 55219 rules; the computation of the phylogeny consumes 22 MB of memory.

Observe in Table 1 that in terms of both computation time and the memory used, the search-based method performs better than the representation-method. These results conforms with our expectations. The representation-based method explicitly defines the weight function, and thus the program/memory size is larger. The search-based method deals with the time consuming computation of weights of phylogenies, not at the representation level but at the search level, so it does not require an ASP representation of the weight function but requires a modification of the solver to guide it find a plausible phylogeny, and hence the smaller the program/memory size and the computation time.

In [2], after computing all 45 phylogenies, the authors examine them manually, and identify 14 of them as plausible and with at most 18 incompatible characters. With the weight measure defined in Section 3, and the representation/search-based methods described above for computing weighted phylogenies, we could automatically compute all plausible phylogenies with at most 18 incompatible characters in 22.35 CPU seconds.

---

[1] All CPU times are in seconds, for a workstation with a 1.5GHz Xeon processor and 4x512MB RAM, running Red Hat Enterprise Linux (Version 4.3).

5 Discussion

We studied the problem of computing weighted solutions in ASP, where the weight of a solution should be defined explicitly. We introduced a search-based method that implements the weight of a solution as a C++ program, and modifies the search algorithm of the ASP solver CLASP to compute weighted solutions with respect to that weight program. We call the modified version of CLASP as CLASP-W.

We showed the applicability and effectiveness of the search-based method in reconstructing phylogenies for Indo-European languages, where the weight of a phylogeny takes into account domain-specific information to characterize the plausibility of phylogenies. In particular, by computing at least w-weighted phylogenies, we could compute more plausible and less number of phylogenies for Indo-European languages. We observed that the search-based method (with CLASP-W) is better than the representation-based method (with CLASP) in terms of computation time and space.

Since no existing phylogenetic system can compute weighted phylogenies, our search-based methods (including the weight measure and the solver CLASP-W) provide a useful tool for experts to automatically analyze phylogenies online. There are many appealing ASP applications (e.g., product configuration, planning) for which finding weighted solutions could be useful; in this sense, our methods are useful also for ASP.

Acknowledgments

References

1. C. Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, 2003.
2. Daniel R. Brooks, Esra Erdem, Selim T. Erdogan, James W. Minett, and Donald Ringe. Inferring phylogenetic trees using answer set programming. *JAR*, 39(4):471–511, 2007.
3. Daniel R. Brooks, Esra Erdem, James W. Minett, and Donald Ringe. Character-based cladistics and answer set programming. In *Proc. of PADL*, pages 37–51, 2005.
4. D.R. Brooks and D.A. McLennan. *Phylogeny, Ecology, and Behavior: A Research Program in Comparative Biology*. University of Chicago Press, Chicago, IL, 1991.
5. William H. E. Day and David Sankoff. Computational complexity of inferring phylogenies by compatibility. *Systematic Zoology*, 35(2):224–229, 1986.
6. Wolfgang Faber, Gerald Pfeifer, Nicola Leone, Tina Dell'Armi, and Giuseppe Ielpa. Design and implementation of aggregate functions in the dlv system. *TPLP*, 8(5-6):545–580, 2008.
7. Martin Gebser, Benjamin Kaufmann, Andr Neumann, and Torsten Schaub. T.: Conflict-driven answer set solving. In *Proc. of IJCAI*, pages 386–392. MIT Press, 2007.
8. Vladimir Lifschitz. What is answer set programming? In *Proc. of AAAI*, 2008.
9. Patrik Simons and Timo Soininen. Stable model semantics of weight constraint rules. In *Proc. of LPNMR*, pages 317–331. Springer-Verlag, 1999.
10. Tran Cao Son and Enrico Pontelli. A constructive semantic characterization of aggregates in answer set programming. *TPLP*, 7(3):355–375, 2007.
11. J.P White and J.F. O'Connell. *A Prehistory of Australia, New Guinea, and Sahul*. Academic, San Diego, CA, 1982.