Unsatisfiable Core Shrinking for Anytime Answer Set Optimization

Mario Alviano\(^1\) and Carmine Dodaro\(^2\)
\(^1\)DEMACS, University of Calabria, Italy
\(^2\)DIBRIS, University of Genova, Italy
alviano@mat.unical.it, dodaro@dibris.unige.it

Abstract

Efficient algorithms for the computation of optimum stable models are based on unsatisfiable core analysis. However, these algorithms essentially run to completion, providing few or even no suboptimal stable models. This drawback can be circumvented by shrinking unsatisfiable cores. Interestingly, the resulting anytime algorithm can solve more instances than the original algorithm.

1 Introduction

In answer set programming (ASP), programs are associated with stable models [Gelfond and Lifschitz, 1991; Niemelä, 1999; Marek et al., 2008; Lifschitz, 2008; Eiter et al., 2009; Brewka et al., 2011], i.e., classical models satisfying a stability condition: only necessary information is included in a model of the input program under the assumptions provided by the model itself for the unknown knowledge in the program, where unknown knowledge is encoded by means of default negation. Reasoning in presence of unknown knowledge is common for rational agents acting in the real world. It is also common that real world agents cannot meet all their desiderata, and therefore ASP programs may come with soft literals for representing numerical preferences over jointly incompatible conditions. Stable models are therefore associated with a cost given by the number of the unsatisfied soft literals, so that stable models of minimum cost are preferred.

It is important here to stress the meaning of the word preferred: any stable model describes a plausible scenario for the knowledge represented in the input program, even if it may be only an admissible solution of non optimum cost. In fact, many rational agents would still accept suboptimal solutions, possibly replacing those in the input program with less restricting constraints until an optimum stable model is found.

2 Background

Let \(\mathcal{A}\) be a set of (propositional) atoms comprising \(\bot\). A literal is an atom \(p\) preceded by zero or more occurrences of the default negation symbol \(\sim\). A rule \(r\) is an implication \(H(r) \leftarrow B(r)\), where \(H(r)\) is a disjunction of atoms, and \(B(r)\) is a conjunction of literals. \(H(r)\) and \(B(r)\) are called head and
body of \( \alpha \), and abusing of notation also denote the sets of their elements. If \( H(\alpha) \subseteq \{ \bot \} \), then \( \alpha \) is called integrity constraint. A program \( \Pi \) is a set of rules. Let \( \text{Attr}(\Pi) \) denote the set of atoms occurring in \( \Pi \).

An interpretation \( I \) is a set of atoms not containing \( \bot \). Relation \( \models \) is inductively defined as follows: for \( p \in \mathcal{A} \), \( I \models p \) if \( p \in I \); \( I \models \neg \alpha \) if \( I \not\models \alpha \); for a rule \( \alpha \), \( I \models \alpha \) if \( I \models \alpha \) for all \( \beta \in \beta(\alpha) \), and \( I \models \alpha \) if \( I \models H(\alpha) \neq \emptyset \) whenever \( I \models \alpha \); for a program \( \Pi \), \( I \models \alpha \) if \( I \models \alpha \) for all \( \alpha \in \Pi \). \( I \) is a model of a literal, rule, or program \( \alpha \) if \( I \models \alpha \).

The reduct \( \Pi' \) of a program \( \Pi \) with respect to an interpretation \( I \) is obtained from \( \Pi \) by removing any rule \( \alpha \) such that \( I \not\models \alpha \) is removed, and then by removing any negated literal. An interpretation \( I \) is a stable model of a program \( \Pi \) if \( I \models \alpha \) and there is no \( J \subset I \) such that \( J \models \Pi' \). Let \( \text{SM}(\Pi) \) denote the set of stable models of \( \Pi \). A program \( \Pi \) is coherent if \( \text{SM}(\Pi) \neq \emptyset \); otherwise, \( \Pi \) is incoherent.

In order to simplify the presentation, a program \( \Pi \) may include count constraints of the form \( \text{COUNT}(\{ t_1, \ldots, t_n \}) \geq k \), where \( t_1, \ldots, t_n (n \geq 0) \) are literals, and \( k \geq 0 \), to enforce \(| \{ t \in [1..N] \mid I \models t \} \geq k \) for all \( I \in \text{SM}(\Pi) \).

For a set \( S \) of literals, called soft, the cost of an interpretation \( I \) is \( S(I) := \{ t \in S \mid I \not\models t \} \), that is, the number of false soft literals. \( I \) is an optimal stable model of a program \( \Pi \) with respect to \( S \) if \( I \in \text{SM}(\Pi) \), and there is no \( J \in \text{SM}(\Pi) \) such that \( S(J) < S(I) \). Let \( \text{OSM}(\Pi, S) \) denote the set of optimal stable models of \( \Pi \) with respect to \( S \). Optimum stable model search is the following computational problem: Given a (coherent) program \( \Pi \) and a set of soft literals \( S \), compute an optimum stable model \( I^* \in \text{OSM}(\Pi, S) \).

Example 1 Let \( \Pi_1 \) be the following program:

\[
\begin{align*}
a & \leftarrow \neg \neg a \\
b \vee c & \leftarrow a \\
b \land d & \leftarrow a \\
c & \leftarrow \neg a
\end{align*}
\]

Its stable models are \( I_1 = \{ b, c \} \), \( I_2 = \{ a, b \} \) and \( I_3 = \{ a, c, d \} \), and the associated reducts are the following:

\[
\begin{align*}
\Pi_1^1 &: b \leftarrow \neg c \\
\Pi_1^2 &: a \leftarrow b \vee c \\
\Pi_1^3 &: a \leftarrow b \vee c \leftarrow a \land d \leftarrow a \\
\end{align*}
\]

If \( S = \{ \neg a, \neg b, \neg c, \neg d \} \) is a set of soft literals, the associated costs are \( S(I_1) = S(I_2) = 2 \), and \( S(I_3) = 3 \). Hence, \( \text{OSM}(\Pi_1, S) = \{ I_1, I_2 \} \).

3 Optimum Stable Model Search via Unsatisfiable Core Analysis

Modern ASP solvers accept as input a set \( L \) of literals, called assumptions, in addition to the usual logic program \( \Pi \), and return a stable model \( M \) of \( \Pi \) such that \( L \subseteq M \), if it exists; otherwise, they return a set \( C \subseteq L \) such that \( \Pi \cup \{ \bot \not\models \neg \alpha \mid \alpha \in C \} \) is incoherent, which is called unsatisfiable core.

Example 2 Consider program \( \Pi_1 \) from Example 1. If \( S = \{ \neg a, \neg b, \neg c, \neg d \} \) is the set of assumptions, the unsatisfiable cores are \( \{ \neg a, \neg b \} \), \( \{ \neg a, \neg c \} \), \( \{ \neg b, \neg c \} \), \( \{ \neg b, \neg d \} \), and their supersets. The algorithm presented in this paper is one, reported as Algorithm 1 (lines 4–10 will be injected later to shrink unsatisfiable cores). A stable model containing all soft literals is searched (line 2). If found, it is an optimum stable model. Otherwise, an unsatisfiable core \( \{ p_0, \ldots, p_n \} \) is returned; since at least one of \( p_0, \ldots, p_n \) must be false in any optimum stable model, the lower bound is increased by one, and the problem is relaxed so that the next call to function solve has to search for a stable model satisfying at least \( n \) literals among \( p_0, \ldots, p_n \). Symmetry breakers of the form \( \bot \leftarrow s_{i-1} \wedge \neg s_{i} \) are also added to \( \Pi \), so that \( s_i \) is true if and only if at least \( n - i + 1 \) literals among \( p_0, \ldots, p_n \) are true.

Example 3 Consider program \( \Pi_1 \) and soft literals \( S = \{ \neg a, \neg b, \neg c, \neg d \} \) from Example 1. A stable model for the program \( \Pi_1 \) and assumptions \( S \) is searched, and an unsatisfiable core is returned. Assume that the returned unsatisfiable core is \( S \) itself. The lower bound \( lb \) is set to 1, the set \( S \) is now equal to \( \{ s_1, s_2, s_3 \} \) and \( \Pi_1 \) is extended with the following rules:

\[
\begin{align*}
s_1 & \leftarrow \neg s_1 \\
s_2 & \leftarrow \neg s_2 \\
s_3 & \leftarrow \neg s_3 \\
\end{align*}
\]

A stable model for the assumptions \( \{ s_1, s_2, s_3 \} \) is searched and an unsatisfiable core, say \( \{ s_1 \} \), is returned. The lower bound \( lb \) is set to 2, the set \( S \) is now equal to \( \{ s_2, s_3 \} \). (Note that the program is extended with the count constraint \( \text{COUNT}(\{ s_1 \}) \geq 0 \), which however is trivially satisfied.) A stable model for the assumptions \( \{ s_2, s_3 \} \) is searched and the answer set \( I'_1 = \{ b, c, s_2, s_3 \} \) is found. Thus, the algorithm terminates returning \( I'_1 \cap \{ a, b, c, d \} = \{ b, c \} = I_1 \).

The analyzed unsatisfiable cores significantly influence the execution of the algorithm, as the set of assumptions and the introduced rules are different for different unsatisfiable cores.

Example 4 Suppose that the first unsatisfiable core returned by function solve for \( \Pi_1 \) and \( S \) from Example 1 is \( \{ \neg a, \neg b, \neg c \} \). Set \( S \) becomes \( \{ \neg b, s_1, s_2 \} \) and \( \Pi_1 \) is extended with the following rules:

\[
\begin{align*}
s_1 & \leftarrow \neg s_1 \\
s_2 & \leftarrow \neg s_2 \\
\end{align*}
\]

The next unsatisfiable core may be \( \{ \neg d, s_1 \} \); therefore, \( S \) becomes \( \{ s_2, s_3 \} \), and \( \Pi_1 \) is extended with the following rules:

\[
\begin{align*}
s_3 & \leftarrow \neg s_3 \\
\end{align*}
\]

At this point a stable model, say \( I'_2 = \{ b, c, s_2, s_3 \} \), is found, and \( I'_2 \cap \{ a, b, c, d \} = \{ b, c \} = I_1 \) is returned.

Note that the algorithm described in this section is completely silent, as it essentially runs to completion without printing any suboptimal stable models. The goal of the next section is to circumvent such a drawback.

3.1 Unsatisfiable Core Shrinking

Unsatisfiable cores returned by function solve are not subset minimal in general. The non-minimality of the unsatisfiable core is justified both theoretically and practically: linearly many coherence checks are required in general to verify
Algorithm 1: Unsatisfiable Core Analysis with ONE

Input: A coherent program $\Pi$, and a nonempty set of soft literals $S$.
Output: An optimum stable model $I^* \in OSM(\Pi, S)$.

1: $lb := 0$; $ub := \infty$; $V := Ar(\Pi)$; // init bounds and visible atoms
2: $(res, I, C) := solve(\Pi, \{p \in S\})$;
3: if $res$ is COHERENT then $I^* := I \cap V$; return $I^*$;
4: Let $C = \{p_0, \ldots, p_n\}$ (for some $n \geq 0$), and $s_1, \ldots, s_n$ be fresh atoms;
5: $\Pi := \Pi \cup \{s_i \leftarrow \neg s_i | i \in [1..n]\} \cup \{\bot \leftarrow s_i \neg s_{i+1} | i \in [1..n-1]\} \cup \{\text{COUNT}\{p_0, \ldots, p_n, \neg s_1, \ldots, \neg s_n\} \geq n\};$
6: $lb := lb + 1$; $S := (S \setminus C) \cup \{s_1, \ldots, s_n\}$; goto 2; // try to solve the relaxed problem

Algorithm 2: Unsatisfiable Core Shrinking with Reiterated Progression

1: $m := -1$; $pr := 1$;
2: Let $C = \{p_0, \ldots, p_n\}$ (for some $n \geq 0$);
3: $(res, I, C) := solve\_with\_budget(\Pi, \{p_i | i \in [0..m + pr]\})$;
4: if $res$ is INCOHERENT then $C := C'$; // smaller core found
5: if $res$ is COHERENT and $lb + S(I) < ub$ then $I^* := I \cap V$; $ub := lb + S(I)$;
6: if $m + 2 \cdot pr \geq |C| - 1$ then $m := m + pr$; $pr := 1/2$; // reiterate progression
7: if $m + 2 \cdot pr < |C| - 1$ then $pr := 2 \cdot pr$; goto 5; // increase progression

the minimality of an unsatisfiable core, hence giving a $\Delta^0_1$-
complete problem; on the other hand, extracting an unsatis-
fiable core after a stable model search failure is quite easy and
usually implemented by identifying the assumptions involved
in the refutation. The non minimality of the analyzed unsat-
fiable cores may affect negatively the performance of sub-
sequent calls to function solve due to aggregation over large
sets. However, it also gives an opportunity to improve Al-
gorithm 1: the size of unsatisfiable cores can be reduced by
performing a few stable model searches within a given budget
on the running time. In more detail, Algorithm 2 is injected
in Algorithm 1. It implements a progression search in the
unsatisfiable core $\{p_0, \ldots, p_n\}$: the size of the assumptions
passed to function solve\_with\_budget is doubled at each call
(line 10), and the progression is reiterated when all assump-
tions are covered (line 9). If solve\_with\_budget terminates
within the given budget, it either returns a smaller unsatis-
fiable core (line 7), or a stable model that possibly improves
the current upper bound (line 8).

Example 5 Consider again the program from Example 3 and
the unsatisfiable core $\{-a, \neg b, \neg c, \neg d\}$ returned after the first
call to function solve. The shrinking process searches a sta-
table model with assumption $\{\neg a\}$, and $I_1 = \{b, c\}$ may be found
within the allotted budget. In any case, a stable model sat-
fifying the assumptions $\{\neg a, \neg b\}$ is searched, and the unsatis-
fiable core $\{-a, \neg b\}$ may be returned if the budget is sufficient.
Otherwise, the progression is reiterated, and one more soft
literal is added to the assumptions. Hence, $\{-a, \neg b, \neg c\}$ may
be returned as an unsatisfiable core if the budget is sufficient.
Otherwise, the original unsatisfiable core is processed.

As an alternative, the shrinking procedure reported in Al-
gorithm 2 can be modified as follows: variable $pr$ is not dou-
bled in line 10, but instead it is incremented by one, i.e.,
$pr := pr + 1$. The resulting procedure is called linear based
shrinking. For unsatisfiable cores of size 4 or smaller, as those
considered in Example 5, the two shrinking procedures coin-
cide, while in general linear based shrinking performs more
stable model searches.

4 Implementation and Experiment

Algorithm ONE [Alviano et al., 2015c] has been imple-
mented in WASP, an ASP solver based on completion [Alviano and
Dodaro, 2016c] also supporting, among other algorithms, li-
near search sat-unsat (LINSU). Within LINSU, a first stable
model is searched to obtain an upper bound of the optimum
cost, and subsequent searches are constrained to improve
the current upper bound, until an incoherence arises. The
implementation of ONE optionally includes the two shrinking
procedures described in Section 3.1, so that both underesti-
mates and overestimates can be produced by WASP in any
case, weighted or unweighted. Currently, the time budget of
function solve\_with\_budget is fixed to 10 seconds, but the ar-
chitecture of WASP can easily accommodate alternative op-
tions, such as a budget proportional to the time required to
find the unsatisfiable core to be shrink.

WASP also implements disjoint cores analysis, which is es-
entially a preliminary step where only soft literals in the
input are passed as assumptions to function solve, while new

<table>
<thead>
<tr>
<th>$\varepsilon(ub, lb)$</th>
<th>WASP</th>
<th>CLASP+ best $lb$ by WASP</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0.00%$</td>
<td>84</td>
<td>88</td>
</tr>
<tr>
<td>$6.25%$</td>
<td>95</td>
<td>101</td>
</tr>
<tr>
<td>$12.50%$</td>
<td>86</td>
<td>96</td>
</tr>
<tr>
<td>$25.00%$</td>
<td>92</td>
<td>99</td>
</tr>
<tr>
<td>$50.00%$</td>
<td>102</td>
<td>99</td>
</tr>
<tr>
<td>$100.00%$</td>
<td>104</td>
<td>107</td>
</tr>
</tbody>
</table>

Table 1: Number of solved instances within a given error estimation (140 testcases).
Another advantage of unsatisfiable core shrinking is that better and better stable models are possibly discovered while computing an optimum stable model. In order to measure the impact of our strategies within this respect, let us define the estimate error $\varepsilon$ of the last stable model produced by Algorithm 1 as follows:

$$
\varepsilon(ub, lb) := \begin{cases} 
\frac{ub-lb}{lb} & \text{if } ub \neq \infty \text{ and } lb \neq 0; \\
\infty & \text{if } ub = \infty, \text{ or both } ub \neq 0 \text{ and } lb = 0; \\
0 & \text{if } ub = lb = 0.
\end{cases}
$$

Hence, the cost associated with the stable model returned by Algorithm 1 is at most $\varepsilon(ub, lb)$ times greater than the cost of an optimum stable model. Such a measure is not applicable to instances of Abstract Dialectical Framework and System Synthesis because of technicality not discussed in this paper.

Table 1 reports the number of instances for which WASP produced a stable model within a given error estimate. In particular, the first row shows the number of instances for which an optimum stable model was computed (error estimate is 0). The last row, instead, shows the number of instances solved with error estimate bounded by 1, and smaller values for the error estimate are considered in the intermediate rows. It is interesting to observe that the stable model produced after the analysis of all disjoint cores is already sufficient to obtain an error estimate bounded by 100% for many tested instances. However, many of these stable models have an error estimate greater than 25%. In this case, adding core shrinking leads to better results.

For the sake of completeness, also CLASP is included in Table 1. However, since CLASP does not print any lower bound, the best value for $lb$ produced by WASP is combined with the upper bounds given by CLASP running LINSU and OLL. If an error estimate of 100% is acceptable, then the number of stable models produced by CLASP is aligned with WASP, or even better. However, when the error estimate must be less or equal than 50%, the combination of disjoint cores analysis and core shrinking implemented by WASP leads to better results in this benchmark.

5 Conclusion

The combination of ASP programs and soft literals is important to ease the modeling of optimization problems. However, the computation of optimum stable models is often very hard, and suboptimal stable models may be the only affordable solutions in some cases. Despite that fact, efficient algorithms based on unsatisfiable core analysis are not anytime. A concrete strategy to turn them into anytime algorithms is given by a shrinking procedure applied to unsatisfiable cores before their analysis: better and better stable models are produced, and eventually a performance gain is obtained thanks to the reduced size of the analyzed unsatisfiable cores. (An alternative technique was introduced in MaxSAT, where one literal is iteratively removed from the unsatisfiable core, either obtaining a smaller unsatisfiable core, or a necessary literal in the processed unsatisfiable core [Nadel, 2010; Nadel et al., 2014].) On the instances of the Sixth ASP Competition, our implementation is often able to provide (suboptimal) stable models with a guarantee of distance to the optimum cost of around 10%.
References


