Max-ASP: Maximum Satisfiability of Answer Set Programs*

Emilia Oikarinen and Matti Järvisalo

Helsinki University of Technology TKK
Department of Information and Computer Science
PO Box 5400, FI-02015 TKK, Finland
emilia.oikarinen@tkk.fi, matti.jarvisalo@tkk.fi

Abstract. This paper studies answer set programming (ASP) in the generalized context of soft constraints and optimization criteria. In analogy to the well-known Max-SAT problem of maximum satisfiability of propositional formulas, we introduce the problems of unweighted and weighted Max-ASP. Given a normal logic program P, in Max-ASP the goal is to find so called optimal Max-ASP models, which minimize the total cost of unsatisfied rules in P and are at the same time answer sets for the set of satisfied rules in P. Inference rules for Max-ASP are developed, resulting in a complete branch-and-bound algorithm for finding optimal models for weighted Max-ASP instances. Differences between the Max-ASP problem and earlier proposed related concepts in the context of ASP are also discussed. Furthermore, translations between Max-ASP and Max-SAT are studied.

1 Introduction

Answer set programming (ASP) is a well-studied declarative programming paradigm that has proven to be an effective approach to knowledge representation and reasoning in various hard combinatorial problem domains. The task of answer set solvers is to find answer sets of ASP programs, representing solutions to the underlying decision problem instance at hand. However, it can often be the case that the problem instance has no solutions since it may be over-constrained. While answer set solvers can in this case prove the non-existence of answer sets, instead of a simple "no" answer, a "near-solution" would be of interest, i.e., an interpretation that is optimal with respect to a specific minimization or maximization criterion, such as the number of unsatisfied rules in the program. For example, in debugging ASP programs (see e.g. [1] and references therein), such an interpretation, or *optimal solution*, could give hints to the reasons for the non-existence of answer sets through a minimal set of unsatisfied rules.

In the field of Boolean satisfiability (SAT), which has close connections to ASP especially from the viewpoint of solver technology, interest in methods for solving the Max-SAT problem (the optimization variant (or generalization) of SAT) has risen especially during recent years [2–4]. Motivation for Max-SAT, where the interest is in optimal truth assignments with respect to the number of unsatisfied clauses, and especially its weighted variant, comes from the possibilities of expressing and solving various optimization and probabilistic reasoning tasks via (weighted) Max-SAT.

* This work is financially supported by Academy of Finland under the project *Methods for Constructing and Solving Large Constraint Models* (grant #122399).

In this paper, we study the problem analogous to Max-SAT for normal logic programs under the stable model semantics, namely *Max-ASP*, or *maximum satisfiability of answer set programs*. In other words, given a normal logic program and integer weights for each rule in the program, in weighted Max-ASP the goal is to find *optimal Max-ASP models* that minimize the total cost (sum of weights) of unsatisfied rules in the program and are at the same time a stable model for remaining (satisfied) rules of the program.

Our contributions are many-fold. In addition to considering basic properties of (optimal) Max-ASP models, we develop various inference (or transformation) rules for reasoning about the optimal cost of Max-ASP instances. Based on the transformation rules, we present a complete branch-and-bound algorithm for determining the optimal cost and an associated optimal model for any Max-ASP instance, also in the weighted case. In fact, the algorithm can be viewed as a generalization of complete search methods proposed for ASP, as some of the presented transformation rules are in a sense generalizations of tableau rules [5] for ASP inference applied in ASP solvers. We also study the relation between Max-ASP and Max-SAT with translations which preserve the solutions between the problems. Furthermore, we discuss differences between Max-ASP and other generalizations [6–9] of answer sets/answer set programs which have a similar flavor. For example, in contrast to Max-ASP, often costs are assigned on literals instead of rules, penalizing the inclusion or exclusion of specific atoms in an answer set of the program at hand, and often answer sets for the whole program are still sought.

This paper is organized as follows. After necessary concepts related to ASP (Sect. 2), we define the Max-ASP problem and discuss properties of optimal Max-ASP models (Sect. 3). We then (Sect. 4) define various transformation rules which preserve the cost of all Max-ASP models and present a complete algorithm for determining the optimal cost of any weighted Max-ASP instance. Before conclusions, translations between Max-ASP and Max-SAT (Sect. 5), and the question of how earlier proposed related concepts can be expressed in Max-ASP (Sect. 6), are considered.

2 Preliminaries

We consider *normal logic programs* (NLPs) in the *propositional* case. A normal logic program Π consists of a finite set of rules of the form

$$r \;:\; h \leftarrow a_1, \ldots, a_n, {\sim}b_1, \ldots, {\sim}b_m, \tag{1}$$

where h, a_i's, and b_j's are propositional atoms. A rule r consists of a *head*, $\mathsf{head}(r) = h$, and a *body*, $\mathsf{body}(r) = \{a_1, \ldots, a_n, {\sim}b_1, \ldots, {\sim}b_m\}$. The symbol "$\sim$" denotes *default negation*. A *default literal* is an atom a, or its default negation ${\sim}a$. The set of atoms occurring in a program Π is $\mathsf{atom}(\Pi)$, and $\mathsf{dlits}(\Pi) = \{a, {\sim}a \mid a \in \mathsf{atom}(\Pi)\}$ is the set of default literals in Π. We use the shorthands $L^+ = \{a \mid a \in L\}$ and $L^- = \{a \mid {\sim}a \in L\}$ for a set L of default literals. Furthermore, we define $\mathsf{body}(\Pi) = \bigcup_{r \in \Pi}\{\mathsf{body}(r)\}$, and $\mathsf{def}(a, \Pi) = \{r \in \Pi \mid \mathsf{head}(r) = a\}$.

In ASP, we are interested in *stable models* [10] (or *answer sets*) of a program Π. An *interpretation* $M \subseteq \mathsf{atom}(\Pi)$ defines which atoms of Π are true ($a \in M$) and which are false ($a \notin M$). An interpretation $M \subseteq \mathsf{atom}(\Pi)$ satisfies a set L of literals, denoted $M \models L$, if and only if $L^+ \subseteq M$ and $L^- \cap M = \emptyset$; and M satisfies a rule $r \in \Pi$, denoted $M \models r$, if and only if $M \models \mathsf{body}(r)$ implies $\mathsf{head}(r) \in M$. An interpretation

$M \subseteq \mathsf{atom}(\Pi)$ is a *(classical) model* of Π, denoted by $M \models \Pi$, if and only if $M \models r$ for each rule $r \in \Pi$. A model M of a program Π is a stable model of Π if and only if there is no model $M' \subset M$ of Π_M, where $\Pi_M = \{\mathsf{head}(r) \leftarrow \mathsf{body}(r)^+ \mid r \in \Pi \text{ and } \mathsf{body}(r)^- \cap M = \emptyset\}$ is called the *Gelfond-Lifschitz reduct* of Π with respect to M. The set of stable models of Π is denoted by $\mathsf{SM}(\Pi)$.

Additional concepts relevant in this work are related to *loops*. The *positive dependency graph* of Π, denoted by $\mathsf{Dep}^+(\Pi)$, is a directed graph with $\mathsf{atom}(\Pi)$ and $\{\langle b, a\rangle \mid \exists r \in \Pi$ such that $b = \mathsf{head}(r)$ and $a \in \mathsf{body}(r)^+\}$ as the sets of vertices and edges, respectively. A non-empty set $L \subseteq \mathsf{atom}(\Pi)$ is a loop in $\mathsf{Dep}^+(\Pi)$ if for any $a, b \in L$ there is a path of non-zero length from a to b in $\mathsf{Dep}^+(\Pi)$ such that all vertices in the path are in L; $\mathsf{loop}(\Pi)$ denotes the set of all loops in $\mathsf{Dep}^+(\Pi)$. The set of *external bodies* of a loop L in Π is

$$\mathsf{eb}_\Pi(L) = \{\mathsf{body}(r) \mid r \in \Pi,\ \mathsf{head}(r) \in L,\ \mathsf{body}(r)^+ \cap L = \emptyset\}.$$

3 Max-ASP

As a central starting point of this work, in this section we define unweighted and weighted Max-ASP and discuss some interesting properties of Max-ASP models.

Definition 1. *Given a NLP Π, an interpretation $M \subseteq \mathsf{atom}(\Pi)$ is a* Max-ASP model *for Π, if M is a stable model for some subset-maximal $\Pi' \subseteq \Pi$ (there is no $\Pi'' \supset \Pi'$ such that $M \in \mathsf{SM}(\Pi'')$). The* cost *of M is $|\Pi \setminus \Pi'|$. A Max-ASP model is* optimal *if it has minimum cost over all Max-ASP models for Π.*

In this work we are especially interested in finding optimal Max-ASP models. We denote the set of all optimal Max-ASP models of a NLP Π by $\mathsf{MaxSM}(\Pi)$.

Example 1. Consider $\Pi = \{a \leftarrow {\sim}b.\ b \leftarrow {\sim}c.\ c \leftarrow {\sim}a\}$. Now $\mathsf{SM}(\Pi) = \emptyset$, but $M = \{a\}$ is a Max-ASP model for Π, since $M \in \mathsf{SM}(\Pi')$ for $\Pi' = \{a \leftarrow {\sim}b.\ c \leftarrow {\sim}a\}$. The cost of M is $|\{b \leftarrow {\sim}c\}| = 1$. Also $\emptyset$ (cost 3), $\{b\}$ (cost 1), and $\{c\}$ (cost 1) are Max-ASP models of Π. Thus $\mathsf{MaxSM}(\Pi) = \{\{a\}, \{b\}, \{c\}\}$ since $\emptyset$ is not optimal.

Notice that Max-ASP models have the following basic properties:

1. Every NLP Π has a Max-ASP model; at least $\emptyset \subseteq \Pi$ trivially has a stable model.
2. If M is a Max-ASP model for Π such that $M \in \mathsf{SM}(\Pi')$ for subset-maximal $\Pi' \subseteq \Pi$, then $M \not\models r$ for all $r \in \Pi \setminus \Pi'$.
3. The sets of optimal Max-ASP models and stable models for Π coincide if and only if Π has a stable model, i.e., $\mathsf{MaxSM}(\Pi) = \mathsf{SM}(\Pi)$ if and only if $\mathsf{SM}(\Pi) \neq \emptyset$.
4. If $M \in \mathsf{MaxSM}(\Pi)$ such that $M \in \mathsf{SM}(\Pi')$ for subset-maximal $\Pi' \subseteq \Pi$, then $\mathsf{SM}(\Pi'') = \emptyset$ for all $\Pi'' \supset \Pi'$.

3.1 Weighted Max-ASP

In analogy with weighted Max-SAT, Max-ASP allows for a natural extension to the weighted case where the rules can be weighted with integer costs.

Definition 2. *A* weighted normal logic program *is a pair $\mathcal{P} = \langle \Pi, W \rangle$, where Π is a NLP and $W : \Pi \rightarrow \mathbb{N}$ is a function that associates a nonnegative integer (a* weight*) with each rule in Π.*

We use the notation introduced in Section 2 in analogous way for weighted NLPs, e.g., $\mathsf{atom}(\mathcal{P}) = \mathsf{atom}(\Pi)$ for $\mathcal{P} = \langle \Pi, W \rangle$.

The concept of a *weighted* Max-ASP model is then naturally defined as follows.

Definition 3. *Given a weighted NLP $\mathcal{P} = \langle \Pi, W \rangle$, a Max-ASP model M for Π is a* (weighted) Max-ASP model *for $\mathcal{P}$. The cost of a Max-ASP model M which is a stable model for subset-maximal $\Pi' \subseteq \Pi$ is*

$$\sum_{r \in \Pi \setminus \Pi'} W(r).$$

A Max-ASP model is optimal *if it has minimum cost over all Max-ASP models for $\mathcal{P}$.*

We denote by $\mathsf{MaxSM}(\mathcal{P})$ the set of all optimal (weighted) Max-ASP models of $\mathcal{P}$. Notationally, we represent a weighted NLP $\mathcal{P} = \langle \Pi, W \rangle$ as a set of pairs

$$\{(r; w) \mid r \in \Pi \text{ and } w = W(r)\}.$$

Example 2. Consider $\mathcal{P} = \{(a \leftarrow {\sim}b; 5), (b \leftarrow {\sim}a; 5), (c \leftarrow a, b; 1), (d \leftarrow {\sim}c, {\sim}d; 1)\}$. Now, $\mathsf{SM}(\mathcal{P}) = \emptyset$, as the first two rules choose either a or b to be true, while the last rule requires that c is true. But to satisfy this, both a and b need to be true. The weights assigned for the rules imply that the mutual exclusion of a and b is more important than satisfaction of the constraint for c. Thus, $\mathsf{MaxSM}(\mathcal{P}) = \{\{a\}, \{b\}\}$ which both have cost 1, as the rule $d \leftarrow {\sim}c, {\sim}d$ is not satisfied. Notice that there is no Max-ASP model for $\mathcal{P}$ such that c is true, because no subset of the rules in $\mathcal{P}$ has such a stable model.

It is worth noticing that the properties of unweighted Max-ASP models discussed in Section 3 also hold in the weighted case.

The weighted variant of Max-ASP is in fact more expressive than the unweighted case. Namely, the decision problems of determining whether a (weighted) NLP has an optimal (weighted) Max-ASP model of cost less than a given value is in $\mathrm{P}^{\mathrm{NP}[\log n]}$ and P^{NP} for the unweighted and weighted case, respectively[1]. This is due to similar results for the well-known Max-SAT and weighted Max-SAT problems [11].

4 Branch-and-Bound for Max-ASP

In this section we present a branch-and-bound algorithm for finding the cost of an optimal Max-ASP model of a weighted NLP $\mathcal{P}$. The algorithm applies a set of equivalence-preserving transformation rules. For presenting these transformations and the branch-and-bound algorithm, we start with some additional notation.

We will assume that an explicit proper upper bound $\top$ is known for the cost of an optimal Max-ASP model for a weighted NLP $\mathcal{P} = \langle \Pi, W \rangle$. This is in analogy with [3],

[1] The unweighted case can be decided in deterministic polynomial time using logarithmic number of calls to an NP-oracle, while a linear number of calls are required for the weighted case.

where a similar approach is applied in the context of Max-SAT. Notice that any value larger than the sum of the weights of all the rules of a program gives such an upper bound $\top$. Given a weighted NLP $\mathcal{P} = \langle \varPi, W \rangle$, an upper bound $\top$, and an interpretation $M \subseteq \mathsf{atom}(\mathcal{P})$, the cost of M in $\mathcal{P}$, denoted by $\mathsf{cost}(M, \mathcal{P}, \top)$, is $c = \sum_{r \in \varPi \setminus \varPi'} W(r)$ if $M \in \mathsf{SM}(\varPi')$ for subset maximal $\varPi' \subseteq \varPi$ such that $c < \top$, and $\top$ otherwise. Furthermore, M is a Max-ASP model for $\mathcal{P}$ if $\mathsf{cost}(M, \mathcal{P}, \top) < \top$, and M is optimal if it has minimum cost over all Max-ASP models for $\mathcal{P}$.

For a given upper bound $\top$, all rules which have weight $w > \top$ must necessarily be satisfied. Thus, such rules can be interpreted as *hard*, whereas rules with a weight less than $\top$ are *soft*. Without loss of generality, we can limit all the costs to the interval $[0 \dots \top]$ and define $w_1 \oplus w_2 = \min(w_1 + w_2, \top)$. Finally, we use the symbol $\Box$ to denote *falsity*, i.e., a rule that is always unsatisfied. Thus, if $(\Box, w) \in \mathcal{P}$, then the cost of any Max-ASP model for $\mathcal{P}$ is at least w, and if $w = \top$, then $\mathcal{P}$ is *unsatisfiable*, i.e., has no Max-ASP models.

Remark 1. By setting $\top = 1$, the problem of finding a Max-ASP model for a weighted NLP $\mathcal{P} = \langle \varPi, W \rangle$ reduces to the problem of finding a stable model for $\varPi$.

Next, we will present the transformation rules which form a central part of our branch-and-bound algorithm for weighted Max-ASP.

4.1 Equivalence-Preserving Transformations

For presenting transformations preserving Max-ASP models, we begin by defining when two weighted NLPs are equivalent.

Definition 4. *Weighted NLPs $\mathcal{P}_1$ and $\mathcal{P}_2$ with a common upper bound $\top$ are equivalent, denoted by $\langle \mathcal{P}_1, \top \rangle \equiv \langle \mathcal{P}_2, \top \rangle$, if*

1. $\mathsf{atom}(\mathcal{P}_1) = \mathsf{atom}(\mathcal{P}_2)$*, and*
2. $\mathsf{cost}(M, \mathcal{P}_1, \top) = \mathsf{cost}(M, \mathcal{P}_2, \top)$ *for all* $M \subseteq \mathsf{atom}(\mathcal{P}_1) = \mathsf{atom}(\mathcal{P}_2)$.

Notice that in order $\mathcal{P}_1$ and $\mathcal{P}_2$ to be equivalent they need to have the same upper bound. Furthermore, notice that it is not sufficient that $\mathsf{MaxSM}(\mathcal{P}_1) = \mathsf{MaxSM}(\mathcal{P}_2)$ holds, but in addition, the cost of each interpretation has to be the same.

Remark 2. With $\top = 1$, i.e., when a stable model is sought, the relation $\equiv$ turns out to be the same as *ordinary* or *weak equivalence*, which requires that $\mathcal{P}_1$ and $\mathcal{P}_2$ have the same set of stable models. Notice that the additional condition $\mathsf{atom}(\mathcal{P}_1) = \mathsf{atom}(\mathcal{P}_2)$ can always be satisfied, e.g., by adding rules of the form $a \leftarrow a$.

Given weighted NLPs $\mathcal{P}$, $\mathcal{P}_1 \subseteq \mathcal{P}$, and $\mathcal{P}_2$, we use $\mathcal{P}_1 \equiv_{\mathcal{P}} \mathcal{P}_2$ as a shorthand for

$$\langle \mathcal{P}, \top \rangle \equiv \langle (\mathcal{P} \setminus \mathcal{P}_1) \cup \mathcal{P}_2, \top \rangle.$$

Finally, we use shorthands $(a; w)$ and $({\sim}a; w)$ where $a \in \mathsf{atom}(\varPi)$ for weighted literals; and $(B; w)$ and $({\sim}B; w)$ where $B \in \mathsf{body}(\varPi)$ for weighted bodies and their complements which can appear in a weighted NLP $\mathcal{P} = \langle \varPi, W \rangle$. These shorthands are easily presented with weighted rules, e.g., $(\{l_1, \dots, l_n\}; w)$ is a shorthand for weighted rules $(f \leftarrow {\sim}f, {\sim}l; w)$ and $(l \leftarrow l_1, \dots, l_n; \top)$; and $({\sim}\{l_1, \dots, l_n\}; w)$ for $(f \leftarrow {\sim}f, l; w)$ and $(l \leftarrow l_1, \dots, l_n; \top)$, where f and l do not appear in $\mathsf{atom}(\varPi)$.

First we present transformation rules for *aggregation*, *hardening*, and *lower-bounding*.

Proposition 1. *Let $\mathcal{P} = \langle \Pi, W\rangle$ be a weighted NLP with an upper bound $\top$. If ψ is a rule $r \in \Pi$ (only in Items 1 and 2), a default literal $l \in \mathsf{dlits}(\Pi)$, or a body $B \in \mathsf{body}(\Pi)$ or its complement $\sim B$, then the following equivalences hold:*

1. *Aggregation:* $\{(\psi; w_1), (\psi; w_2)\} \equiv_{\mathcal{P}} \{(\psi; w_1 \oplus w_2)\}$
2. *Hardening:* $\{(\psi; w_1), (\Box; w_2)\} \equiv_{\mathcal{P}} \{(\psi; \top), (\Box; w_2)\}$, *if* $w_1 \oplus w_2 = \top$
3. *Lower-bounding:* $\{(\psi; \top), (\sim\psi; w)\} \equiv_{\mathcal{P}} \{(\psi; \top), (\Box; w)\}$

The hardening rule allows one to identify rules that are equivalent to hard counterparts: the violation of $(\psi; w_1)$ has cost $\top$. The lower-bounding rule makes the lower bound implied by the violation of a hard constraint explicit. The proof of Proposition 1 is omitted due to space constraints.

The next transformations are for inference between bodies and literals in the bodies.

Proposition 2. *Let $\mathcal{P} = \langle \Pi, W\rangle$ be a weighted NLP, $\top$ an upper bound, l and l' literals in $\mathsf{dlits}(\Pi)$, and $B \in \mathsf{body}(\Pi)$. The following equivalences hold:*

4. *Forward true body (FTB):* $\{(l; \top) \mid l \in B\} \equiv_{\mathcal{P}} \{(l; \top) \mid l \in B\} \cup \{(B; \top)\}$
5. *Backward false body (BFB):*
$$\{(\sim B; \top)\} \cup \{(l; \top) \mid l \in B \setminus \{l'\}\} \equiv_{\mathcal{P}} \{(\sim B; \top)\} \cup \{(l; \top) \mid l \in B \setminus \{l'\}\} \cup \{(\sim l'; \top)\}$$
6. *Forward false body (FFB):* $\{(\sim l; \top)\} \equiv_{\mathcal{P}} \{(\sim l; \top), (\sim B; \top)\}$, *if* $l \in B$
7. *Backward true body (BTB):* $\{(B; \top)\} \equiv_{\mathcal{P}} \{(B; \top), (l; \top))\}$, *if* $l \in B$

All these transformation rules require the constraints involved be hard, and this way a body is interpreted as a conjunction of its default literals. Without going into further details, we note that the correctness of these transformations follows from the similarity with sound ASP inference rules for normal logic programs presented in [5].

Finally, we have transformations relating head atoms with the rules defining them.

Proposition 3. *Let $\mathcal{P} = \langle \Pi, W\rangle$ be a weighted NLP and $\top$ an upper bound. The following equivalences hold:*

8. *Forward true atom (FTA):*
$$\{(B; \top), (h \leftarrow B; w)\} \equiv_{\mathcal{P}} \{(B; \top), (h \leftarrow B; 0), (h; w)\}$$
9. *Backward false atom (BFA):*
$$\{(\sim h; \top), (h \leftarrow B; w)\} \equiv_{\mathcal{P}} \{(\sim h; \top), (h \leftarrow B; 0), (\sim B; w)\}$$
10. *Forward false atom (FFA):*
$$\{(\sim B; \top) \mid B \in \mathsf{body}(\mathsf{def}(h, \Pi))\} \equiv_{\mathcal{P}} \{(\sim B; \top) \mid B \in \mathsf{body}(\mathsf{def}(h, \Pi))\} \cup \{(\sim h; \top)\}$$
11. *Forward loop (FL): Let $h \in L$ and $L \in \mathsf{loop}(\Pi)$. Then*
$$\{(\sim B; \top) \mid B \in \mathsf{eb}_{\Pi}(L)\} \equiv_{\mathcal{P}} \{(\sim B; \top) \mid B \in \mathsf{eb}_{\Pi}(L)\} \cup \{(\sim h; \top)\}$$
12. *Backward true atom (BTA):*
$$\{(\sim B; \top) \mid B \in \mathsf{body}(\mathsf{def}(h, \Pi)) \setminus \{B'\}\} \cup \{(h; \top)\} \equiv_{\mathcal{P}} \{(\sim B; \top) \mid B \in \mathsf{body}(\mathsf{def}(h, \Pi)) \setminus \{B'\}\} \cup \{(h; \top), (B'; \top)\}$$

For understanding these transformations, we note that since a Max-ASP model needs to be a stable model for some subset of rules, only hard constraints are inferred in *FFA*, *FL*, and *BTA*. On the other hand, the "soft" transformation rules *FTA* and *BFA* correspond to satisfaction of rules (the cost of a Max-ASP model M comes from the rules r such that $M \not\models r$). Moreover, the precise form of *FTA* and *BFA* is related to the global nature of *FFA*, *FL*, and *BTA*. Instead of removing the rule in question when applying *FTA* or *BFA*, we change its cost to zero. There is no cost involved if a rule with zero cost is unsatisfied, but nevertheless, the effect of such rules needs to be taken into account when applying *FFA*, *FL*, and *BTA*. We now consider the correctness of the transformation rules *FTA* and *FFA* in more detail. The other cases are similar.

FTA: Let

$$\mathcal{P} = \mathcal{P}' \cup \{(B;\top), (h \leftarrow B; w)\} \text{ and } \mathcal{P}'' = \mathcal{P}' \cup \{(B;\top)\}, (h \leftarrow B; 0), (h; w),$$

and consider arbitrary $M \subseteq \mathsf{atom}(\mathcal{P})$. If $M \not\models B$, then $\mathsf{cost}(M, \mathcal{P}, \top) = \top$ and $\mathsf{cost}(M, \mathcal{P}'', \top) = \top$. If $M \models B$, then $M \not\models h \leftarrow B$ if and only if $h \notin M$. Thus, either $\{(h \leftarrow B; w)\}$ and $\{(h; w), (h \leftarrow B; 0)\}$ are both satisfied in M, or neither of them is satisfied, and $\mathsf{cost}(M, \mathcal{P}, \top) = \mathsf{cost}(M, \mathcal{P}'', \top)$.

FFA: Let

$$\mathcal{P} = \langle \Pi, W \rangle = \mathcal{P}' \cup \{(\sim B; \top) \mid B \in \mathsf{body}(\mathsf{def}(h, \Pi))\} \text{ and}$$
$$\mathcal{P}'' = \mathcal{P}' \cup \{(\sim B; \top) \mid B \in \mathsf{body}(\mathsf{def}(h, \Pi))\} \cup \{(\sim h; \top)\},$$

and consider arbitrary $M \subseteq \mathsf{atom}(\mathcal{P})$. If $h \notin M$, then it holds that $\mathsf{cost}(M, \mathcal{P}'', \top) = \mathsf{cost}(M, \mathcal{P}, \top)$. If $h \in M$, then $\mathsf{cost}(M, \mathcal{P}'', \top) = \top$. Assume that $\mathsf{cost}(M, \mathcal{P}, \top) < \top$. Then there is $\Pi' \subseteq \Pi$ such that $M = \mathsf{SM}(\Pi')$. However, $M \not\models B$ for all $B \in \mathsf{body}(\mathsf{def}(h, \Pi))$. This implies that there can be no rule r in Π' such that $\mathsf{head}(r) = h$ and $M \models \mathsf{body}(r)$, and furthermore, $h \notin M$. This is a contradiction, and thus, $\mathsf{cost}(M, \mathcal{P}, \top) = \top$.

4.2 Branch-and-Bound

We are now ready to introduce a depth-first branch-and-bound algorithm which, given a weighted NLP $\mathcal{P}$ and a cost upper bound $\top$, determines the cost of the optimal weighted Max-ASP models of $\mathcal{P}$ given that the optimal cost is less than $\top$. The method, presented as Algorithm 1, applies the equivalence-preserving transformations introduced in Propositions 1, 2, and 3 in PROPAGATE($\mathcal{P}, \top$) (Line 1). After applying the transformations, the algorithm makes *choices* by case analysis on $(l; \top)$, where $l \in \mathsf{dlits}(\mathcal{P})$, such that $(l; \top) \notin \mathcal{P}$ (represented by the choice heuristic SELECTLITERAL($\mathcal{P}$) on Line 4). This leads to a complete search for determining the cost of Max-ASP models of weight less than $\top$. The algorithm can also easily be modified to also return an optimal model in addition to its cost (as demonstrated in Example 3). If there are no models with cost less than $\top$, the algorithm returns $\top$ (Line 2). Recall that the lower-bounding rule guarantees that $\{(a; \top), (\sim a; \top)\} \subseteq \mathcal{P}$ is impossible.

While a formal correctness proof is omitted here, the correctness is based on the fact that, once $(a; \top) \in \mathcal{P}$ or $(\sim a; \top) \in \mathcal{P}$ for all $a \in \mathsf{atom}(\mathcal{P})$ (Line 2), the transformation

Algorithm 1 MAXASP($\mathcal{P}, \top$).

```
1. $\mathcal{P} := \text{PROPAGATE}(\mathcal{P}, \top)$
2. if $(\Box, \top) \in \mathcal{P}$ then return $\top$
3. if $\forall a \in \mathsf{atom}(\mathcal{P})$ either $(a; \top) \in \mathcal{P}$ or $(\sim a; \top) \in \mathcal{P}$ then
   3a. if $(\Box, w) \in \mathcal{P}$ then return $w$
   3b. else return $0$
4. $l := \text{SELECTLITERAL}(\mathcal{P})$
5. $v := \text{MAXASP}(\mathcal{P} \cup \{(l; \top)\}, \top)$
6. $v := \text{MAXASP}(\mathcal{P} \cup \{(\sim l; v)\}, v)$
7. return $v$
```

rules are complete in the sense that they assure that the lower bound can not be tightened further. Since the transformation rules also guarantee that the current weighted NLP remains equivalent to the original one, the current lower bound is the optimal cost.

The following example illustrates a run of the algorithm.

Example 3. Consider $\mathcal{P} = \{(b \leftarrow a;\ 1), (a \leftarrow b;\ 2), (a \leftarrow \sim c;\ 3)\}$ with $\top = 7$. We start with PROPAGATE($\mathcal{P}, \top$). Since atom c has no defining rules, we can apply *FFA* and obtain $\mathcal{P} \cup \{(\sim c; \top)\}$. We continue by applying *FTB* and *FTA* and get $\mathcal{P}_1 = \{(b \leftarrow a;\ 1), (a \leftarrow b;\ 2), (a \leftarrow \sim c;\ 0), (\sim c; \top), (\{\sim c\}; \top), (a;\ 3)\}$. None of the transformation rules is applicable to $\mathcal{P}_1$, and the stopping criteria on Line 2 and Line 3 do not hold. Thus we make a choice. Let SELECTLITERAL($\mathcal{P}$) return $\sim a$.

– We add $(\sim a; \top)$ to $\mathcal{P}_1$ and after propagation using *FFB*, *lower-bounding*, *FFA*, and *FFB*, we obtain

$$\mathcal{P}_2 = \{(b \leftarrow a;\ 1), (a \leftarrow b;\ 2), (a \leftarrow \sim c;\ 0), (\sim c; \top), (\{\sim c\}; \top), (\sim a; \top), (\sim\{a\}; \top), (\Box;\ 3), (\sim b; \top), (\sim\{b\}; \top)\}.$$

No further propagation is applicable, and the condition on Line 3 holds. Now, $\{a \in \mathsf{atom}(\mathcal{P}) \mid (a; \top) \in \mathcal{P}_2\} = \emptyset$ and $\mathsf{cost}(\emptyset, \mathcal{P}, \top) = \mathsf{cost}(\emptyset, \mathcal{P}_2, \top) = 3$. Furthermore, $\emptyset$ is a (not necessarily optimal) Max-ASP model for $\mathcal{P}_2$, and thus also for $\mathcal{P}$. We set $\top = 3$ and continue the search by backtracking.

– After adding $(a;\ \top)$ to $\mathcal{P}_1$, we get $\{(b \leftarrow a;\ 0), (a \leftarrow b;\ 2), (a \leftarrow \sim c;\ 0), (\sim c; \top), (\{\sim c\}; \top), (a; \top), (\{a\}; \top), (b; 1)\}$ by application of *aggregation*, *FTB*, and *FTA*. Let SELECTLITERAL($\mathcal{P}$) return b. After the use of *aggregation* and *FTB* we have

$$\mathcal{P}_3 = \{(b \leftarrow a;\ 0), (a \leftarrow b;\ 0), (a \leftarrow \sim c;\ 0), (\sim c; \top), (\{\sim c\}; \top), (a; \top), (\{a\}; \top), (b;\ \top), (\{b\}; \top)\}.$$

Now, $M = \{a, b\} = \{a \in \mathsf{atom}(\mathcal{P}) \mid (a; \top) \in \mathcal{P}_3\}$ and $\mathsf{cost}(M, \mathcal{P}, \top) = \mathsf{cost}(M, \mathcal{P}_3, \top) = 0$. Thus we have found an optimal Max-ASP model, which in this case is a stable model for the NLP.

Remark 3. In the case $\top = 1$, the transformation rules in Propositions 2 and 3 resemble closely inference rules in tableau calculi for ASP proposed in [5]. In fact, Algorithm 1 can be viewed as a generalization of complete search methods proposed for ASP, as some of the presented transformation rules are generalizations of tableau rules [5]. However, for compactness we have intentionally left out additional transformation rules (related to *well-founded sets* and loops) which would generalize their counterparts in [5].

5 From Max-ASP to Max-SAT

In this section we analyze the relationship between Max-ASP and Max-SAT, giving solution-preserving translations between these problems. For this we first briefly go through necessary concepts related to Max-SAT.

Let X be a set of Boolean variables. Associated with every variable $x \in X$ there are two *literals*, the positive literal, denoted by x, and the negative literal, denoted by $\bar{x}$. A *clause* is a disjunction of distinct literals. We view a clause as a finite set of literals and a *CNF formula* as a finite set of clauses. A *truth assignment* τ associates a truth value $\tau(x) \in \{\mathsf{false}, \mathsf{true}\}$ with each variable $x \in X$. A truth assignment *satisfies* a CNF formula if and only if it satisfies every clause in it. A clause is satisfied if and only if it contains at least one satisfied literal, where a literal x ($\bar{x}$, respectively) is satisfied if $\tau(x) = \mathsf{true}$ ($\tau(x) = \mathsf{false}$, respectively). A CNF formula is *satisfiable* if there is a truth assignment that satisfies it, and *unsatisfiable* otherwise.

A *weighted CNF formula* is a pair $\mathcal{W} = \langle \mathcal{C}, W \rangle$, where $\mathcal{C}$ is a CNF formula and $W : \mathcal{C} \to \mathbb{N}$ is a function that associates a nonnegative integer with each clause in $\mathcal{C}$.

Definition 5. *Given a weighted CNF formula $\mathcal{W} = \langle \mathcal{C}, W \rangle$, an upper bound $\top$, and a truth assignment τ for $\mathcal{C}$, the cost of τ in $\mathcal{W}$, denoted by* $\mathsf{cost}(\tau, \mathcal{W}, \top)$*, is the sum of weights of the clauses not satisfied by τ. If* $\mathsf{cost}(\tau, \mathcal{W}, \top) < \top$*, τ is a* Max-SAT model *for $\mathcal{W}$, and τ is* optimal *if it has minimum cost over all Max-SAT models for $\mathcal{W}$.*

5.1 Max-ASP as Max-SAT

The translation from Max-ASP to Max-SAT is based on a typical translation from ASP to SAT, i.e., Clark's completion [12, 13] with loop formulas [14]. In the following, we assume without loss of generality that $r_i \neq r_j$ for all rules $r_i, r_j \in \Pi$ in a weighted NLP $\mathcal{P} = \langle \Pi, W \rangle$. Furthermore, we use shorthands as follows: if l is a default literal, i.e., a or $\sim a$, then $x_l = x_a$ if $l = a$, and $x_l = \bar{x}_a$ if $l = \sim a$.

Definition 6. *Given a weighted NLP $\mathcal{P} = \langle \Pi, W \rangle$ with an upper bound $\top$, the translation* $\mathsf{MaxSAT}(\mathcal{P}, \top)$ *consists of the following weighted clauses:*

1. *For each $a \in \mathsf{atom}(\Pi)$:*
   $(\{x_a, \bar{x}_{B_1}\}; w_1), \ldots, (\{x_a, \bar{x}_{B_m}\}; w_m)$ *and* $(\{x_{B_1}, \ldots, x_{B_m}, \bar{x}_a\}; \top)$,
   where $B_i = \mathsf{body}(r_i)$ and $w_i = W(r_i)$ for each $r_i \in \mathsf{def}(a, \Pi)$.
2. *For each $B = \{l_1, \ldots, l_n\} \in \mathsf{body}(\Pi)$:*
   $(\{x_{l_1}, \bar{x}_B\}; \top), \ldots, (\{x_{l_n}, \bar{x}_B\}; \top)$*, and* $(\{x_B, \bar{x}_{l_1}, \ldots, \bar{x}_{l_n}\}; \top)$.
3. *For each $L \in \mathsf{loop}(\Pi)$ and each $a \in L$:*
   $(\{x_{B_1}, \ldots, x_{B_m}, \bar{x}_a\}; \top)$ *where* $\mathsf{eb}_\Pi(L) = \{B_1, \ldots, B_m\}$.

There is a bijective correspondence between the Max-ASP models of $\mathcal{P}$ and Max-SAT models of its translation $\mathsf{MaxSAT}(\mathcal{P}, \top)$.

Theorem 1. *Given a weighted NLP $\mathcal{P} = \langle \Pi, W \rangle$, an upper bound $\top$, and a Max-ASP model M for $\mathcal{P}$ with cost $w < \top$, the truth assignment*

$$\tau(x) = \begin{cases} \mathsf{true}, & \textit{if } x = x_a \textit{ for } a \in \mathsf{atom}(\Pi) \textit{ and } a \in M, \\ \mathsf{true}, & \textit{if } x = x_B \textit{ for } B \in \mathsf{body}(\Pi) \textit{ and } M \models B, \\ \mathsf{false}, & \textit{otherwise.} \end{cases}$$

is a Max-SAT model for $\mathsf{MaxSAT}(\mathcal{P}, \top)$ *with cost w.*

Proof. Let M be a Max-ASP model for $\mathcal{P} = \langle \Pi, W \rangle$ with cost w, τ defined as in Definition 6, and $\Pi' \subseteq \Pi$ subset-maximal such that $M \in \mathsf{SM}(\Pi')$.

Consider first the clauses in Item 2 of Definition 6. Assume that clause $(\{x_l, \bar{x}_B\}; \top)$ is not satisfied by τ, i.e., $\tau(x_l) = \mathsf{false}$ and $\tau(x_B) = \mathsf{true}$. But then $M \not\models l$ and $M \models B$ which leads to contradiction as $l \in B$. Similarly, τ satisfies all clauses of the form $(\{x_B, \bar{x}_{l_1}, \ldots, \bar{x}_{l_n}\}; \top)$; and furthermore, τ satisfies all clauses in Item 2.

Next, notice that the hard clauses in Item 3 are effectively the loop formulas of Π in clausal form [14]. Assume now that there is a clause $(\{x_{B_1}, \ldots, x_{B_m}, \bar{x}_a\}; \top)$ that τ does not satisfy, i.e, $\tau(x_{B_i}) = \mathsf{false}$ for all i and $\tau(x_a) = \mathsf{true}$. Thus $M \not\models B$ for all $B \in \mathsf{eb}_\Pi(L)$ and there is $a \in M \cap L$. Since $a \in M$ and $M = \mathsf{LM}((\Pi')_M)$, there is a rule $r \in (\Pi')_M$ such that $a = \mathsf{head}(r)$ and $M \models \mathsf{body}(r)$. Moreover, $\mathsf{body}(r) \cap L \neq \emptyset$, since $M \not\models B$ for all $B \in \mathsf{eb}_\Pi(L)$. Thus there is $b \in \mathsf{body}(r) \cap L \cap M$. Continuing this process, one notices that $M \models L$ and moreover, $L \in \mathsf{loop}(\Pi')$. Since $\Pi' \subseteq \Pi$, also $\mathsf{eb}_{\Pi'}(L) \subseteq \mathsf{eb}_\Pi(L)$. Therefore $M \not\models B$ for all $B \in \mathsf{eb}_{\Pi'}(L)$, and there is a loop formula of Π' not satisfied in M. This is a contradiction to $M \in \mathsf{SM}(\Pi')$, since a stable model of a program must satisfy all its loop formulas [14]. Thus, τ satisfies all the clauses in Item 3.

As regards the weighted clauses in Item 1, we notice that if $a \in M$, then there is a rule $a \leftarrow B \in \mathsf{def}(a, \Pi') \subseteq \mathsf{def}(a, \Pi)$ such that $M \models B$, i.e., $\tau(x_a) = \mathsf{true}$ and $\tau(x_B) = \mathsf{true}$ for some $B \in \mathsf{body}(\mathsf{def}(a, \Pi))$. Thus τ satisfies all weighted clauses in Item 1. On the other hand, if $a \notin M$, then we notice that $M \not\models \mathsf{body}(r)$ for all $r \in \mathsf{def}(a, \Pi) \cap \Pi'$ and $M \models \mathsf{body}(r)$ for all $r \in \mathsf{def}(a, \Pi) \setminus \Pi'$. The cost of clauses related to $\mathsf{def}(a, \Pi)$ which τ does not satisfy is the same as the cost of rules $r \in \mathsf{def}(a, \Pi)$ such that $M \not\models r$. Thus, the overall cost of violated clauses is exactly the cost of rules in $\Pi \setminus \Pi'$. □

Remark 4. Note that if $\top = 1$, all clauses are hard, and $\mathsf{MaxSAT}(\mathcal{P}, \top)$ is a clausal form of Clark's completion of $\mathcal{P}$ with loop formulas of $\mathcal{P}$. Thus, $M \in \mathsf{SM}(\mathcal{P})$ if and only if τ satisfies $\mathsf{MaxSAT}(\mathcal{P}, 1)$ [14].

Theorem 2. *Given a weighted NLP $\mathcal{P} = \langle \Pi, W \rangle$, an upper bound $\top$, and a Max-SAT model τ with cost $w < \top$ for* $\mathsf{MaxSAT}(\mathcal{P}, \top)$, *the interpretation*

$$\{a \in \mathsf{atom}(\Pi) \mid \tau(x_a) = \mathsf{true}\}$$

is a Max-ASP model for $\mathcal{P}$ with cost w.

Proof. Assume that τ is a Max-SAT model for $\mathsf{MaxSAT}(\mathcal{P}, \top)$ with cost $w < \top$, and $M = \{a \in \mathsf{atom}(\Pi) \mid \tau(x_a) = \mathsf{true}\}$. Define $\Pi' = \{r \in \Pi \mid M \models r\}$ and $\mathcal{P}' = \langle \Pi', W' \rangle$ such that $W'(r) = W(r)$ for all $r \in \Pi'$.

Let us consider $\mathsf{MaxSAT}(\mathcal{P}', 1)$. Since $w < \top$, the clauses in Item 2 of Definition 6 of $\mathsf{MaxSAT}(\mathcal{P}, \top)$ force that $\tau(x_B) = \mathsf{true}$ if and only if $\tau(x_l) = \mathsf{true}$ for all $l \in B$. Furthermore, since $\mathsf{body}(\Pi') \subseteq \mathsf{body}(\Pi)$, τ satisfies all the clauses in Item 2 of $\mathsf{MaxSAT}(\mathcal{P}', 1)$ as well.

Consider arbitrary $a \in \mathsf{atom}(\Pi)$. If $a \in M$, i.e., $\tau(x_a) = \mathsf{true}$, then $\mathsf{def}(a, \Pi') = \mathsf{def}(a, \Pi)$. Then $\mathsf{MaxSAT}(\mathcal{P}', 1)$ contains the same clauses related to $\mathsf{def}(a, \Pi')$ as $\mathsf{MaxSAT}(\mathcal{P}, \top)$, and furthermore, τ satisfies all these clauses. If $a \notin M$, i.e., $\tau(x_a) = \mathsf{false}$, then for all $r \in \mathsf{def}(a, \Pi')$ it holds that $M \not\models \mathsf{body}(r)$. Thus τ satisfies all clauses related to $\mathsf{def}(a, \Pi')$.

Consider an arbitrary $L \in \mathsf{loop}(\Pi') \subseteq \mathsf{loop}(\Pi)$. If $\mathsf{eb}_{\Pi'}(L) = \mathsf{eb}_{\Pi}(L)$, then τ satisfies all clauses in Item 3 of $\mathsf{MaxSAT}(\mathcal{P}', 1)$. Assume $\mathsf{eb}_{\Pi'}(L) \subset \mathsf{eb}_{\Pi}(L)$ and consider arbitrary $a \in L$. If $a \notin M$, i.e., $\tau(x_a) = \mathsf{false}$, then τ satisfies the clause in Item 3 related to a. If $a \in M$, i.e., $\tau(x_a) = \mathsf{true}$, then there is $B \in \mathsf{body}(\mathsf{def}(a, \Pi'))$ such that $\tau(x_B) = \mathsf{true}$, i.e., $M \models B$, since τ satisfies the clause $(\{x_{B_1}, \ldots, x_{B_m}, \bar{x}_a\}; \top)$ for $\{B_1, \ldots, B_m\} = \mathsf{body}(\mathsf{def}(a, \Pi'))$. If $B \in \mathsf{eb}_{\Pi'}(L)$ then τ satisfies all clauses in Item 3 related to L. If $B \notin \mathsf{eb}_{\Pi'}(L)$, then $B \cap L \neq \emptyset$, and thus $M \models B$ implies that there is $a' \in L \cap M$, i.e., $\tau(x_{a'}) = \mathsf{true}$. Again, there must be $B' \in \mathsf{body}(\mathsf{def}(a', \Pi'))$ such that $\tau(x_{B'}) = \mathsf{true}$, i.e., $M \models B'$. If $B' \in \mathsf{eb}_{\Pi'}(L)$, we are done as τ satisfies all clauses in Item 3 related L. Assume now that $\tau(x_B) = \mathsf{false}$ for all $B \in \mathsf{eb}_{\Pi'}(L)$. Continuing the process we notice that $a \in M$ for all $a \in L$. Since τ satisfies the clauses in Item 3 of $\mathsf{MaxSAT}(\mathcal{P}, \top)$ there is $B \in \mathsf{eb}_{\Pi}(L) \setminus \mathsf{eb}_{\Pi'}(L)$ such that $\tau(x_B) = \mathsf{true}$. Thus, there is $r \in \Pi \setminus \Pi'$ such that $B = \mathsf{body}(r)$ and $\mathsf{head}(r) \in L$. Since $r \notin \Pi'$, it holds that $\mathsf{head}(r) \notin M$. This is in contradiction with $L \subseteq M$, and therefore τ satisfies all clauses in Item 3 of $\mathsf{MaxSAT}(\mathcal{P}', 1)$.

Now, τ is a Max-SAT model for $\mathsf{MaxSAT}(\mathcal{P}', 1)$ with cost 0, i.e., M is a stable model of Π'. Finally, note that the sum of the weights of the rules in $\Pi \setminus \Pi'$ is w. $\square$

5.2 Max-SAT as Max-ASP

There is a natural linear-size translation from CNF formulas to NLPs so that there is a bijective correspondence between the satisfying truth assignments of any CNF formula and stable models of its translation [15]. We give a generalization of the translation to establish a similar correspondence between Max-SAT and Max-ASP. If l is a literal, i.e., variable x or its negation $\bar{x}$, then $\sim a_l = a_x$ if $l = \bar{x}$ and $\sim a_l = \sim a_x$ if $l = x$. We assume without loss of generality that $C_i \neq C_j$ for all clauses $C_i, C_j \in \mathcal{C}$ in a weighted CNF formula $\mathcal{W} = \langle \mathcal{C}, W \rangle$.

Definition 7. *Given a weighted CNF formula $\mathcal{W} = \langle \mathcal{C}, W \rangle$ with an upper bound $\top$, the translation $\mathsf{MaxASP}(\mathcal{W}, \top)$ consists of the following weighted rules:*

1. *For each clause $C_i = \{l_1, \ldots, l_n\} \in \mathcal{C}$:*
   $(f_i \leftarrow \sim f_i, \sim a_{l_1}, \ldots, \sim a_{l_n}; w_i)$, where $w_i = W(C_i)$.
2. *For each variable x in $\mathcal{C}$: $(\hat{a}_x \leftarrow \sim a_x; \top)$, and $(a_x \leftarrow \sim \hat{a}_x; \top)$.*

Theorem 3. *Let $\mathcal{W} = \langle \mathcal{C}, W \rangle$ be a weighted CNF formula, and $\top$ an upper bound.*

– *If τ is a Max-SAT model for $\mathcal{W}$ with cost $w < \top$, then*

$$\{a_x \mid \tau(x) = \mathsf{true}\} \cup \{\hat{a}_x \mid \tau(x) = \mathsf{false}\}$$

is a Max-ASP model for $\mathsf{MaxASP}(\mathcal{W}, \top)$ *with cost* w.

– *If* M *is a Max-ASP model for* $\mathsf{MaxASP}(\mathcal{W}, \top)$ *with cost* $w < \top$, *then*

$$\tau(x) = \begin{cases} \mathsf{true}, & \textit{if } a_x \in M, \textit{ and} \\ \mathsf{false}, & \textit{if } \hat{a}_x \in M. \end{cases}$$

is a Max-SAT model for $\mathcal{W}$ *with cost* w.

6 Related Approaches in ASP

In this section we discuss the similarities and differences of Max-ASP and some extensions of stable models most closely related to Max-ASP models.

Simons et al. [7] consider an extended ASP language including cardinality and choice rules and, especially, *optimize statements*. An optimize statement is a *minimize statement* or its dual, a *maximize statement*. A minimize statement is of the form $\textsc{minimize}(a_1 = w_{a_1}, \ldots, a_n = w_{a_n}, {\sim}b_1 = w_{b_1}, \ldots, {\sim}b_m = w_{b_m})$ where a_i's and b_j's are atoms. First, recall that several optimize statements can be encoded into a single one with suitable weights [7]. Now, given a program $\mathit{\Pi}$ and a minimize statement, the goal is to find a stable model M for $\mathit{\Pi}$ such that $\sum_{a_i \in M} w_{a_i} + \sum_{b_i \notin M} w_{b_i}$ is minimized. For an NLP $\mathit{\Pi}$, we can view this optimization problem as a weighted NLP $\mathcal{P}$ where all the rules from $\mathit{\Pi}$ are hard, and in addition, there are soft constraints $(f \leftarrow {\sim}f, a_i; w_{a_i})$ for each $1 \leq i \leq n$ and $(f \leftarrow {\sim}f, {\sim}b_i; w_{b_i})$ for each $1 \leq i \leq m$.

Buccafurri et al. [6] consider *strong and weak constraints* in *disjunctive Datalog programs*. A strong constraint is of the form $\leftarrow a_1, \ldots, a_n, {\sim}b_1, \ldots, {\sim}b_m$ which can be viewed as a rule $f \leftarrow {\sim}f, a_1, \ldots, a_n, {\sim}b_1, \ldots, {\sim}b_m$. Thus, a stable model must necessarily satisfy all the strong, i.e., hard constraints. A weak constraint is of the form $\Leftarrow a_1, \ldots, a_n, {\sim}b_1, \ldots, {\sim}b_m$, and it is possible to set different priorities for weak constraints. The goal is to minimize the number of unsatisfied weak constraints according to their priorities. In the case of NLPs, weak constraints can be represented in a natural way in Max-ASP by using weights for expressing priorities.

On the other hand, while (weighted) Max-ASP can be expressed with NLPs and either minimize statements or weak constraints (all three problems have the same complexity), we do not see an immediate and natural way of encoding Max-ASP in the other two formalisms.

Gebser et al. [9] introduce ι-stable models, which have similar properties as (unweighted) Max-ASP models. An interpretation $M \subseteq \mathsf{atom}(\mathit{\Pi})$ is a *ι-stable model* for NLP $\mathit{\Pi}$, if $M = \mathsf{LM}(\mathit{\Pi}'_\emptyset)$ for some subset maximal $\mathit{\Pi}' \subseteq \mathit{\Pi}$ such that $\mathsf{body}^+(\mathit{\Pi}') \subseteq \mathsf{LM}(\mathit{\Pi}'_\emptyset)$ and $\mathsf{body}^-(\mathit{\Pi}') \cap \mathsf{LM}(\mathit{\Pi}'_\emptyset) = \emptyset$. It is worth noticing that a ι-stable model M of $\mathit{\Pi}$ is a stable model of $\mathit{\Pi}$ if and only if $M \models \mathit{\Pi}$, and the same applies to Max-ASP models. It is straightforward to see that a ι-stable model M for $\mathit{\Pi}$ is a Max-ASP model for $\mathit{\Pi}$. However, ι-stable models are not necessarily *optimal* Max-ASP models. For instance, $\mathit{\Pi}_1 = \{a \leftarrow {\sim}d.\ b \leftarrow {\sim}e.\ c \leftarrow a, b.\ e \leftarrow {\sim}a\}$ [9] has two ι-stable models, namely $\{e\}$, and $\{a, b, c\}$. Now, $\mathsf{SM}(\mathit{\Pi}_1) = \{\{a, b, c\}\}$ and $\{e\}$ is not optimal. On the other hand, a Max-ASP model of a NLP is not necessarily a ι-stable model. For instance, consider $\mathit{\Pi}_2 = \{a \leftarrow {\sim}c.\ \ a \leftarrow b, c.\ \ b \leftarrow a.\ \ b \leftarrow c.\ \ c \leftarrow a, b\}$ from [9, Example 6.3]. Now, $\mathsf{MaxSM}(\mathit{\Pi}_2) = \{\emptyset, \{a\}, \{a, b\}\}$ which each have cost one. However, $\{a, b\}$ is the unique ι-stable model of $\mathit{\Pi}_2$.

7 Conclusions

We study the unweighted and weighted Max-ASP problems from several different angles. Most importantly, we develop sound transformation rules for Max-ASP inference, based on which we present a complete algorithm for computing optimal weighted Max-ASP models. Translations between Max-ASP and Max-SAT are also developed, in addition to relating Max-ASP to related concepts in ASP.

Our branch-and-bound algorithm builds ground for implementations for Max-ASP search. We find that the study of inference and search methods, including study of additional transformation rules as well as solver development, for (weighted) Max-ASP could prove a fruitful direction for further study. Additionally, we aim to study translation-based approaches for solving Max-ASP as, e.g., Max-SAT, so that the already developed Max-SAT solvers can be exploited in analogy to SAT-based approaches to ASP such as [14].

References

1. Gebser, M., Pührer, J., Schaub, T., Tompits, H.: A meta-programming technique for debugging answer-set programs. In: Proc. AAAI'08, AAAI Press (2008) 448–453
2. Li, C., Manyá, F.: MaxSAT, hard and soft constraints. In: Handbook of Satisfiability. IOS Press (2009)
3. Larrosa, J., Heras, F., de Givry, S.: A logical approach to efficient Max-SAT solving. Artificial Intelligence **172**(2–3) (2008) 204–233
4. Bonet, M., Levy, J., Manyà, F.: Resolution for Max-SAT. Artificial Intelligence **171**(8–9) (2007) 606–618
5. Gebser, M., Schaub, T.: Tableau calculi for answer set programming. In: Proc. ICLP'06. Volume 4079 of LNCS., Springer (2006) 11–25
6. Buccafurri, F., Leone, N., Rullo, P.: Enhancing disjunctive datalog by constraints. IEEE Transactions on Knowledge and Data Engineering **12**(5) (2000) 845–860
7. Simons, P., Niemelä, I., Soininen, T.: Extending and implementing the stable model semantics. Artificial Intelligence **138**(1–2) (2002) 181–234
8. Brewka, G., Niemelä, I., Truszczynski, M.: Answer set optimization. In: Proc. IJCAI'03, Morgan Kaufmann (2003) 867–872
9. Gebser, M., Gharib, M., Mercer, R., Schaub, T.: Monotonic answer set programming. Journal of Logic and Computation (in press (2009)) doi:10.1093/logcom/exn040.
10. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: Proc. ICLP/SLP'88, MIT Press (1988) 1070–1080
11. Papadimitriou, C.: Computational Complexity. Addison-Wesley (1995)
12. Clark, K.: Negation as failure. In: Readings in nonmonotonic reasoning. Morgan Kaufmann Publishers (1987) 311–325
13. Fages, F.: Consistency of Clark's completion and existence of stable models. Journal of Methods of Logic in Computer Science **1** (1994) 51–60
14. Lin, F., Zhao, Y.: ASSAT: Computing answer sets of a logic program by SAT solvers. Artificial Intelligence **157**(1–2) (2004) 115–137
15. Niemelä, I.: Logic programs with stable model semantics as a constraint programming paradigm. Annals of Mathematics and Artificial Intelligence **25**(3–4) (1999) 241–273