2019 IEEE 49th International Symposium on Multiple-Valued Logic (ISMVL)

Clausal Form Transformation in MaxSAT *

Chu Min Li
MIS, Université de Picardie

Felip Manyà
IIIA-CSIC

Joan Ramon Soler
IIIA-CSIC

Abstract

Some clausal form transformation algorithms used in SAT solving cannot be used in MaxSAT solving because they preserve satisfiability but do not preserve the minimum number of unsatisfied formulas. In this paper we define three different MaxSAT clausal form transformations, inspired on the transformations applied in SAT, that derive a multiset of clauses ψ from a multiset of arbitrary propositional formulas ϕ in such a way that the minimum number of unsatisfied clauses in ψ is equal to the minimum number of unsatisfied formulas in ϕ.

1 Introduction

There has been tremendous progress in theoretical and applied aspects of the MaxSAT problem over the last decade. As a result, there are now a number of competitive solvers that are able to solve challenging optimization problems in different areas (see e.g. [1, 2, 3, 4, 5, 6, 7, 9, 10, 11, 12, 13, 14, 18, 19, 20, 21, 22, 23] and the references therein).

MaxSAT solvers require the input to be a multiset of clauses, and return an assignment that minimizes the number of unsatisfied clauses in the input multiset. However, many problems in real-world applications are encoded as a set of propositional formulas which are not necessarily clauses. Such encodings cannot be solved with modern MaxSAT solvers. Thus, some kind of clausal form transformation is needed to solve them.

In SAT, there are a number of algorithms that transform a set of arbitrary propositional formulas into a satisfiability equivalent set of clauses [24, 25]. Thus, SAT solvers requiring the input in clausal form can decide the satisfiability of a set of propositional formulas by using those algorithms. Unfortunately, some clausal form transformations used in SAT are not valid in MaxSAT. The reason is that they do not preserve the minimum number of unsatisfied formulas in the input set of formulas. It is therefore important to analyze if existing SAT clausal form transformations are valid in MaxSAT, as well as to investigate how they can be adapted to deal with MaxSAT instances when they are not valid.

In this paper we address the problem of deriving a multiset of clauses ψ from a multiset of arbitrary propositional formulas ϕ in such a way that the minimum number of unsatisfied clauses in ψ is equal to the minimum number of unsatisfied formulas in ϕ. Thus, by deriving such cost-preserving multisets, we provide a way of solving the MaxSAT problem of a multiset of arbitrary propositional formulas with MaxSAT solvers in which the input is required to be a multiset of clauses.

We define three different MaxSAT clausal form transformations. The first transformation adds an additional step to the direct SAT clausal form transformation based on applying logical equivalences. The second transformation is a variant of the first one that has the advantage of producing more compact encodings. The third transformation avoids the combinatorial explosion of the other transformations by introducing auxiliary variables to rename subformulas.

As in SAT, we can make the distinction between MaxSAT and non-clausal MaxSAT. The former is to find an assignment that minimizes the number of clauses that can be unsatisfied in a given multiset of clauses. The latter is to find an assignment that minimizes the number of formulas that can be unsatisfied in a given multiset of arbitrary propositional formulas. In this paper, it is clear from the context when we refer to MaxSAT or non-clausal MaxSAT.

MaxSAT is also defined as the problem of finding an assignment that maximizes the number of clauses that can be satisfied in a given multiset of clauses. This definition is equivalent to the one in the previous paragraph, because the assignments that maximize the number of satisfied clauses are also the assignments that minimize the number of unsatisfied clauses. We prefer the minimization version because modern MaxSAT solvers find optimal solutions by minimizing the number of unsatisfied clauses.

The paper is structured as follows. Section 2 defines basic concepts. Section 3 defines and discuss different ways of transforming multisets of propositional formulas into

*This work was supported by the MINECO-FEDER project RASO TIN2015-71799-C2-1-P.

2378-2226/19/$31.00 ©2019 IEEE
DOI 10.1109/ISMVL.2019.00031

MaxSAT clausal forms. Section 4 gives the conclusions.

2 Preliminaries

Given a set of variables $\{x_1, \ldots, x_n\}$, a literal is a variable x_i or its negation $\neg x_i$. A weighted clause is a pair (c, w), where c is a disjunction of literals and w, its weight, is a positive integer or infinity. If its weight is infinity, it is a hard clause (we omit infinity weights for simplicity); otherwise it is a soft clause. A Weighted Partial MaxSAT instance is a multiset of weighted clauses.

We represent MaxSAT instances using multisets instead of sets because repeated clauses cannot be collapsed into one of such clauses as in SAT. Clauses can remain represented by the set of its literals because repeated literals can be collapsed into one literal without affecting the preservation of the minimum number of unsatisfied clauses.

A truth assignment assigns to each variable either 0 or 1. It satisfies literal x_i ($\neg x_i$) if x_i evaluates to 1 (0), weighted clause (c, w) if it satisfies a literal of c, and a multiset of clauses if it satisfies all its clauses. The weight w is the penalty of violating clause c. When all clauses have the same weight, their weights are omitted.

The Weighted Partial MaxSAT problem, or WPMaxSAT, for an instance ϕ is to find an assignment that satisfies the hard clauses and minimizes the sum of the weights of the unsatisfied soft clauses. The most common subproblems of WPMaxSAT are the following: Weighted MaxSAT (WMaxSAT), which is WPMaxSAT without hard clauses; Partial MaxSAT (PMaxSAT), which is WPMaxSAT when all the soft clauses have the same weight, and MaxSAT, which is PMaxSAT without hard clauses.

3 Clausal form transformation

This section defines three new MaxSAT clausal form transformations. For ease of presentation, we first consider only unweighted formulas. In the last subsection, we explain how such transformations can be extended to deal with weighted, soft and hard formulas.

3.1 The direct MaxSAT clausal form transformation

For each formula in a multiset of propositional formulas ϕ, we can derive its conjunctive normal form (CNF) with the procedure below, where the uppercase letters A, B and C denote propositional formulas.

1. Remove all the occurrences of the implications using the following rules:

$$(A \rightarrow B) \rightsquigarrow (\neg A \vee B)$$

$$(A \leftrightarrow B) \rightsquigarrow (\neg A \vee B) \wedge (A \vee \neg B)$$

2. Reduce the scope of negation until negations appear only in front of literals using the following rules:

$$\neg\neg A \rightsquigarrow A$$

$$\neg(A \vee B) \rightsquigarrow (\neg A \wedge \neg B)$$

$$\neg(A \wedge B) \rightsquigarrow (\neg A \vee \neg B)$$

3. Derive a multiset of conjunctions of clauses using the following rules:

$$A \vee (B \wedge C) \rightsquigarrow (A \vee B) \wedge (A \vee C)$$

$$(A \wedge B) \vee C \rightsquigarrow (A \vee C) \wedge (B \vee C)$$

4. Remove clauses containing a literal and its complementary, because they are tautological.

In SAT, the obtained conjunction of clauses obtained in step 4 is usually represented by a set of clauses, where each clause is interchangeably represented by the disjunction of its literals and the set of its literals. In the following, we refer to such a set of clauses as *direct SAT clausal form*.

Example 1. Let $\phi = \{(\neg x_1 \leftrightarrow x_1) \wedge (\neg x_2 \leftrightarrow x_2), \neg x_1 \vee x_2\}$ be a multiset of propositional formulas, containing the formula $(\neg x_1 \leftrightarrow x_1) \wedge (\neg x_2 \leftrightarrow x_2)$ and the formula $\neg x_1 \vee x_2$, which is already in CNF. Applying the above procedure, we get that the CNF of the first formula is $x_1 \wedge \neg x_1 \wedge x_2 \wedge \neg x_2$. Thus, the SAT clausal form of ϕ is $\{\{x_1\}, \{\neg x_1\}, \{x_2\}, \{\neg x_2\}, \{\neg x_1, x_2\}\}$, which is $\{x_1, \neg x_1, x_2, \neg x_2, \neg x_1 \vee x_2\}$ when clauses are represented by disjunctions of literals instead of sets of literals.

If we look at the resulting multiset of clauses of Example 1 as a MaxSAT instance, then an exact MaxSAT solver will conclude that the minimum number of unsatisfied clauses in the clausal form of ϕ is 2, because we can derive a contradiction from $\{x_1\}$ and $\{\neg x_1\}$, and another from $\{x_2\}$ and $\{\neg x_2\}$. However, the minimum number of unsatisfied formulas in ϕ is 1. Hence, the described SAT clausal form transformation is not valid in MaxSAT, because it preserves logical equivalence between the input multiset of propositional formulas and the resulting multiset of clauses but does not preserve the minimum number of unsatisfied formulas. In other words, it is not *cost preserving*.

To overcome this drawback we add an additional step to the previous procedure. We start by considering the CNF of each input formula as a conjunction of clauses $C_1 \wedge C_2 \wedge \cdots \wedge C_m$ and then impose that exactly one contradiction can be derived when at least one of such clauses is unsatisfied. To this end, we must derive a multiset of clauses from $C_1 \wedge C_2 \wedge \cdots \wedge C_m$ by applying the following cost preserving rules:

R1: $A \wedge B \rightsquigarrow \{A, \neg A \vee B\}$.

R2: $\neg(A \vee B) \rightsquigarrow \{\neg A, A \vee \neg B\}$.

R3: $\{A, B\} \vee C \rightsquigarrow \{A \vee C, B \vee C\}$.

In the following, we refer to the multiset of clauses obtained by applying R1, R2 and R3 as *direct MaxSAT clausal form*.

The correctness of these rules is easy to prove. The next proposition proves the correctness of R1. Note that transforming $A \wedge B$ into $\{A, B\}$ is not cost preserving.

Proposition 1. *Let A and B be propositional formulas. The minimum number of formulas that can be unsatisfied in $A \wedge B$ is equal to the minimum number of formulas that can be unsatisfied in $\{A, \neg A \vee B\}$.*

Proof Assume that I is an assignment that unsatisfies $A \wedge B$. Then, I unsatisfies exactly one formula of $\{A, \neg A \vee B\}$ because A and $\neg A$ cannot be simultaneously unsatisfied and $\{A, \neg A \vee B\}$ is only satisfied when A and B evaluate to true.

Assume now that I is an assignment that unsatisfies $\{A, \neg A \vee B\}$. Then, either I unsatisfies A or I satisfies A and unsatisfies B. In both case, I unsatisfies $A \wedge B$.

Since the number of unsatisfied formulas is preserved for every assignment, it follows that the minimum number of unsatisfied formulas is also preserved. □

Example 2. Given the formula $(x_1 \vee x_2) \wedge (x_3 \vee x_4)$, which is already in CNF, we convert it to a cost preserving multiset of clauses as follows:

$$
\begin{array}{l}
(x_1 \vee x_2) \wedge (x_3 \vee x_4) =^{R1} \\
\{x_1 \vee x_2, \neg(x_1 \vee x_2) \vee (x_3 \vee x_4)\} =^{R2} \\
\{x_1 \vee x_2, \{\neg x_1, x_1 \vee \neg x_2\} \vee (x_3 \vee x_4)\} =^{R3} \\
\{x_1 \vee x_2, \neg x_1 \vee x_3 \vee x_4, x_1 \vee \neg x_2 \vee x_3 \vee x_4\}
\end{array}
$$

Note that the direct MaxSAT clausal form is larger than the direct SAT clausal form.

3.2 The improved MaxSAT clausal form transformation

A way of improving the previous transformation is by introducing auxiliary variables in the direct SAT clausal form. As a result, we obtain a more compact MaxSAT clausal form that is a Partial MaxSAT instance.

Definition 2. Let $\phi = \{A_1, \ldots, A_m, \ldots, A_n\}$ be a multiset of propositional formulas such that $A_1, \ldots, A_m$ are not clauses and $A_{m+1}, \ldots, A_n$ are clauses. Let $y_{A_1}, \ldots, y_{A_m}$ be auxiliary propositional variables. Let $CF(A_i) = \{C_1^i, \ldots, C_{r_i}^i\}$ be the multiset of clauses of the direct SAT clausal form of A_i for $i = 1, \ldots, m$. The improved MaxSAT clausal form of ϕ is the Partial MaxSAT instance that has as hard clauses the multiset

$$\bigcup_{i=1}^{m} \{C_1^i \vee \neg y_{A_i}, \ldots, C_{r_i}^i \vee \neg y_{A_i}\} \quad (1)$$

and as soft clauses the multiset

$$\bigcup_{i=1}^{m} y_{A_i} \cup \bigcup_{j=m+1}^{n} A_j \,. \quad (2)$$

Example 3. Given the multiset of propositional formulas $\{x_1 \wedge (\neg x_1 \vee x_2), (x_3 \vee x_2) \wedge (\neg x_3 \vee x_2), \neg x_1 \vee \neg x_2\}$, whose formulas are in CNF, we derive the partial MaxSAT instance that contains the following hard clauses:

$$
\begin{array}{l}
x_1 \vee \neg y_1 \\
\neg x_1 \vee x_2 \vee \neg y_1 \\
x_3 \vee x_2 \vee \neg y_2 \\
\neg x_3 \vee x_2 \vee \neg y_2
\end{array}
$$

and the following soft clauses:

$$
\begin{array}{l}
y_1 \\
y_2 \\
\neg x_1 \vee \neg x_2
\end{array}
$$

The following proposition states that the minimum number of propositional formulas that can be unsatisfied in the input multiset is equal to the the minimum number of soft clauses that can be unsatisfied in the resulting Partial MaxSAT instance.

Proposition 3. *The improved MaxSAT clausal form transformation is cost preserving.*

Proof It follows from the fact that all the occurrences of the auxiliary variables y_{A_i} in the hard part of the improved MaxSAT clausal form have negative polarity. Then, when at least one clause of the direct SAT clausal form of the propositional formula A_i is unsatisfied, y_{A_i} must be set to false to satisfy the hard part and the unit soft clause y_{A_i} becomes unsatisfied. In this way, an optimal assignment of the input multiset of propositional formulas unsatisfies A_i iff an optimal assignment of the improved MaxSAT clausal form unsatisfies the unit clause y_{A_i}, which is the single soft clauses related to A_i. □

The main problem of the two preceding transformations is that they can produce multisets of clauses whose size is exponential in the size of the corresponding input propositional formulas due to the application of the distributivity rules.

3.3 The Tseitin-style MaxSAT clausal form transformation

The way of obtaining a clausal form from a propositional formula A with the Tseitin transformation [25] in SAT relies on adding an auxiliary variable y_ρ for each subformula ρ of A that is not a literal. Each auxiliary variable y_ρ renames a subformula ρ, depending on its top-most connective, by adding one of the following equivalences:

- $y_\rho \leftrightarrow y_B \circ y_C$ if $\rho = B \circ C$ and $\circ \in \{\wedge, \vee, \leftrightarrow, \rightarrow\}$
- $y_\rho \leftrightarrow \neg y_B$ if $\rho = \neg B$

where B and C are subformulas of ρ, and y_B (y_C) is equal to B (C) if B (C) is a literal.

More precisely, given a propositional formula A that it is not a clause, the Tseitin transformation derives the following clausal form:

$$\{y_A\} \cup \left(\bigcup_{\substack{\rho \in SF(A) \\ \rho \notin Lit(A)}} Def(A, \rho) \right) \tag{3}$$

where y_A is the auxiliary variable associated to A, $SF(A)$ is the set of subformulas of A, $Lit(A)$ is the set of literals occurring in A, and $Def(A, \rho)$ is the definition of subformula ρ in A (see Definition 4).

Definition 4. Given a propositional formula A and a subformula ρ of A that is not a literal, the definition of subformula ρ in A, denoted by $Def(A, \rho)$, is defined as follows:

- If $\rho = B \wedge C$, then

  $$Def(A, \rho) = \{\neg y_\rho \vee y_B, \neg y_\rho \vee y_C, y_\rho \vee \neg y_B \vee \neg y_C\}$$

- If $\rho = B \vee C$, then

  $$Def(A, \rho) = \{\neg y_\rho \vee y_B \vee y_C, y_\rho \vee \neg y_B, y_\rho \vee \neg y_C\}$$

- If $\rho = B \rightarrow C$, then

  $$Def(A, \rho) = \{\neg y_\rho \vee \neg y_B \vee y_C, y_\rho \vee y_B, y_\rho \vee \neg y_C\}$$

- If $\rho = B \leftrightarrow C$, then

  $$\begin{aligned} Def(A, \rho) = \quad & \{y_\rho \vee y_B \vee y_C, y_\rho \vee \neg y_B \vee \neg y_C, \\ & \neg y_\rho \vee \neg y_B \vee y_C, \neg y_\rho \vee y_B \vee \neg y_C\} \end{aligned}$$

- If $\rho = \neg B$, then

  $$Def(A, \rho) = \{\neg y_\rho \vee \neg y_B, y_\rho \vee y_B\}$$

Example 4. The Tseitin-style SAT clausal form transformation of the propositional formula $(\neg x_1 \leftrightarrow x_1) \wedge (\neg x_2 \leftrightarrow x_2)$ of Example 1 is as follows:

$$\begin{aligned} & \{y_1, \\ & \neg y_1 \vee y_2, \neg y_1 \vee y_3, y_1 \vee \neg y_2 \vee \neg y_3, \\ & \neg y_2 \vee x_1, \neg y_2 \vee \neg x_1, \\ & \neg y_3 \vee x_2, \neg y_3 \vee \neg x_2\} \end{aligned}$$

where y_1 denotes $y_{(\neg x_1 \leftrightarrow x_1) \wedge (\neg x_2 \leftrightarrow x_2)}$, y_2 denotes $y_{(\neg x_1 \leftrightarrow x_1)}$ and y_3 denotes $y_{(\neg x_2 \leftrightarrow x_2)}$. Note that the second line corresponds to $y_1 \leftrightarrow y_2 \wedge y_3$, the thirth line to $y_2 \leftrightarrow (\neg x_1 \leftrightarrow x_1)$ and the fourth line to $y_3 \leftrightarrow (\neg x_2 \leftrightarrow x_2)$.

The following definition describes a Tseitin-style clausal form transformation for MaxSAT.

Definition 5. Let $\phi = \{A_1, \ldots, A_m, \ldots, A_n\}$ be a multiset of propositional formulas such that $A_1, \ldots, A_m$ are not clauses and $A_{m+1}, \ldots, A_n$ are clauses, and let $T(A_i)$ be the multiset of clauses derived by applying Equation 3 for $i = 1, \ldots, m$. The Tseitin-style MaxSAT clausal form transformation of ϕ is the Partial MaxSAT instance that has as hard clauses the multiset

$$\bigcup_{i=1}^{m} (T(A_i) \setminus \{y_{A_i}\}) \tag{4}$$

and as soft clauses the multiset

$$\bigcup_{i=1}^{m} y_{A_i} \cup \bigcup_{j=m+1}^{n} A_j \,. \tag{5}$$

Example 5. The Tseitin-style MaxSAT clausal form transformation of $\phi = \{x_1 \wedge x_2, x_3 \wedge x_4, \neg x_1 \vee \neg x_3, \neg x_2 \vee \neg x_4\}$ is the Partial MaxSAT instance that has the following hard clauses:

$$\begin{aligned} & \neg y_1 \vee x_1 \\ & \neg y_1 \vee x_2 \\ & y_1 \vee \neg x_1 \vee \neg x_2 \\ & \neg y_2 \vee x_3 \\ & \neg y_2 \vee x_4 \\ & y_2 \vee \neg x_3 \vee \neg x_4 \end{aligned}$$

and the following soft clauses:

$$\begin{aligned} & y_1 \\ & y_2 \\ & \neg x_1 \vee \neg x_3 \\ & \neg x_2 \vee \neg x_4 \end{aligned}$$

Proposition 6. *The Tseitin-style MaxSAT clausal form transformation is cost preserving.*

Proof It follows from the following fact: In the multiset of clauses $T(A_i) \setminus \{y_{A_i}\}$, for every pair of clauses of the form $y_\rho \vee D$ and $\neg y_\rho \vee D'$ in $Def(A, \rho)$, where D and D' are disjunctions of literals, it holds that there is a literal l in D such that $\neg l$ is in D', and there is a literal l' in D' such that $\neg l'$ is in D. This implies that, for each auxiliary variable y_ρ, the block of clauses of the form $y_\rho \vee D$ and the block of clauses of the form $\neg y_\rho \vee D'$ cannot be simultaneously unsatisfied. Thus, by setting adequately the auxiliary variables, we can build a satisfying assignment of $T(A_i) \setminus \{y_{A_i}\}$. When the satisfaction of $T(A_i) \setminus \{y_{A_i}\}$ forces y_{A_i} to be true, then A_i is satisfied; and when it forces y_{A_i} to be false, then A_i is unsatisfied. Therefore, the minimum number of unsatisfied soft clauses in the Tseitin-style MaxSAT clausal form is equal to the minimum number of unsatisfied formulas in the input multiset of propositional formulas. Note that the soft unit clause y_{A_i} is the single soft clause related to A_i and the variable y_{A_i} does not appear in the Tseitin transformations of the rest of formulas of the input multiset. □

Taking into account the argument in the proof of Proposition 6 that states that, for every auxiliary variable y_ρ, the block of clauses of the form $y_\rho \vee D$ and the block of clauses of the form $\neg y_\rho \vee D'$ cannot be simultaneously unsatisfied, it turns out that, an optimal assignment of the input multiset of formulas unsatisfies A_i iff an optimal assignment of the (SAT) Tseitin clausal form unsatisfies y_{A_i}. Hence, in contrast to the direct transformation, the Tseitin transformation preserves the minimum number of unsatisfied formulas.[1]

Proposition 7. *The Tseitin transformation is cost preserving.*

However, from a practical point of view, our preliminary tests indicate that using the Tseitin-style MaxSAT clausal form transformation is more efficient than using directly the Tseitin transformation.

3.4 Dealing with weights

If we associate a weight to each propositional formula, this weight can be easily incorporated into the clausal form transformations defined so far. In the case of the direct MaxSAT clausal form transformation, we associate the weight of the formula to each clause related to that formula; it works because at most one of such clauses can be unsatisfied in the derived MaxSAT instance.The same happens if we use directly the Tseitin transformation. In the case of the improved and Tseitin-style MaxSAT clausal form transformations, we associate the weight of the formula A_i to the soft unit clause y_{A_i}, which is the single soft clause related to A_i.

[1] It has not been reported before, to the best of our knowledge, that the Tseitin transformation is cost preserving.

If we consider hard formulas, we can add as hard clauses any SAT clausal form transformation of the hard formulas. We do not need to use any MaxSAT clausal form transformations because hard clauses are always satisfied in any optimal solution.

Example 6. Given the multiset of propositional formulas $\phi = \{x_1 \wedge (\neg x_1 \vee x_2), (x_3 \vee x_2) \wedge (\neg x_3 \vee x_2), \neg x_1 \vee \neg x_2\}$ of Example 3, if we assign a weight of 3 to the first formula of ϕ, a weight of 2 to the second formula and a weight of 5 to the third formula, we get the improved MaxSAT clausal form consisting of the following hard clauses:

$$
\begin{array}{l}
x_1 \vee \neg y_1 \\
\neg x_1 \vee x_2 \vee \neg y_1 \\
x_3 \vee x_2 \vee \neg y_2 \\
\neg x_3 \vee x_2 \vee \neg y_2
\end{array}
$$

and the following soft clauses:

$$
\begin{array}{l}
(y_1, 3) \\
(y_2, 2) \\
(\neg x_1 \vee \neg x_2, 5)
\end{array}
$$

Example 7. Given the multiset of propositional formulas $\phi = \{x_1 \wedge x_2, x_3 \wedge x_4, \neg x_1 \vee \neg x_3, \neg x_2 \vee \neg x_4\}$ of Example 5, if we assign a weight of 2 to the first two formulas and a weight of 5 to the last two formulas, and introduce the hard constraint $x_1 \leftrightarrow x_4$, we get the Tseitin-style MaxSAT clausal form consisting of the following hard clauses:

$$
\begin{array}{l}
\neg y_1 \vee x_1 \\
\neg y_1 \vee x_2 \\
y_1 \vee \neg x_1 \vee \neg x_2 \\
\neg y_2 \vee x_3 \\
\neg y_2 \vee x_4 \\
y_2 \vee \neg x_3 \vee \neg x_4 \\
y_3 \\
y_3 \vee x_1 \vee x_4 \\
y_3 \vee \neg x_1 \vee \neg x_4 \\
\neg y_3 \vee \neg x_1 \vee x_4 \\
\neg y_3 \vee x_1 \vee \neg x_4
\end{array}
$$

and the following soft clauses:

$$
\begin{array}{l}
(y_1, 2) \\
(y_2, 2) \\
(\neg x_1 \vee \neg x_3, 5) \\
(\neg x_2 \vee \neg x_4, 5)
\end{array}
$$

4 Concluding remarks

We have defined three MaxSAT clausal form transformations, called direct, improved and Tseitin-style transformations. The proposed transformations preserve the minimum

number of unsatisfied formulas. Moreover, we have shown that the Tseitin transformation is cost preserving. To the best of our knowledge, this is the first contribution towards non-clausal MaxSAT solving. Thanks to the presented results, non-clausal MaxSAT instances can be solved with existing clausal MaxSAT solvers.

As future work, we plan to extend the results on clause MaxSAT tableaux [15] to obtain a complete tableau calculus for non-clausal MaxSAT. Another interesting research line is to compare non-clausal MinSAT with MinSAT [16, 17]. From the multiple-valued logic perspective, it could be interesting to extend the results of this paper to derive cost-preserving signed clausal forms [8].

References

[1] A. Abramé and D. Habet. On the resiliency of unit propagation to Max-Resolution. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence, IJCAI-2015, Buenos Aires, Argentina*, pages 268–274, 2015.

[2] C. Ansótegui, J. Gabàs, and J. Levy. Exploiting subproblem optimization in SAT-based MaxSAT algorithms. *J. Heuristics*, 22(1):1–53, 2016.

[3] C. Ansótegui, I. Izquierdo, F. Manyà, and J. T. Jiménez. A Max-SAT-based approach to constructing optimal covering arrays. In *Proceedings of the 16th International Conference of the Catalan Association for Artificial Intelligence, CCIA 2013, Vic, Spain*, volume 256 of *Frontiers in Artificial Intelligence and Applications*, pages 51–59. IOS Press, 2013.

[4] J. Argelich, A. Cabiscol, I. Lynce, and F. Manyà. Efficient encodings from CSP into SAT, and from MaxCSP into MaxSAT. *Multiple-Valued Logic and Soft Computing*, 19(1-3):3–23, 2012.

[5] J. Argelich, C. M. Li, F. Manyà, and J. Planes. The first and second Max-SAT evaluations. *Journal on Satisfiability, Boolean Modeling and Computation*, 4:251–278, 2008.

[6] J. Argelich and F. Manyà. Exact Max-SAT solvers for over-constrained problems. *Journal of Heuristics*, 12(4–5):375–392, 2006.

[7] F. Bacchus, A. Hyttinen, M. Järvisalo, and P. Saikko. Reduced cost fixing for maximum satisfiability. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI, Stockholm, Sweden*, pages 5209–5213, 2018.

[8] B. Beckert, R. Hähnle, and F. Manyà. The SAT problem of signed CNF formulas. In D. Basin, M. D'Agostino, D. Gabbay, S. Matthews, and L. Viganò, editors, *Labelled Deduction*, volume 17 of *Applied Logic Series*, pages 61–82. Kluwer, Dordrecht, 2000.

[9] M. Bofill, M. Garcia, J. Suy, and M. Villaret. MaxSAT-based scheduling of B2B meetings. In *Proceedings of the12th International Conference on Integration of AI and OR Techniques in Constraint Programming, CPAIOR, Barcelona, Spain*, pages 65–73, 2015.

[10] M. L. Bonet, J. Levy, and F. Manyà. Resolution for Max-SAT. *Artificial Intelligence*, 171(8–9):240–251, 2007.

[11] F. Heras, A. Morgado, and J. Marques-Silva. MaxSAT-based encodings for Group MaxSAT. *AI Communications*, 28(2):195–214, 2015.

[12] M. Koshimura, T. Zhang, H. Fujita, and R. Hasegawa. Qmaxsat: A partial max-sat solver. *JSAT*, 8(1/2):95–100, 2012.

[13] C. M. Li and F. Manyà. MaxSAT, hard and soft constraints. In A. Biere, H. van Maaren, and T. Walsh, editors, *Handbook of Satisfiability*, pages 613–631. IOS Press, 2009.

[14] C. M. Li, F. Manyà, N. O. Mohamedou, and J. Planes. Resolution-based lower bounds in MaxSAT. *Constraints*, 15(4):456–484, 2010.

[15] C. M. Li, F. Manyà, and J. R. Soler. A clause tableaux calculus for MaxSAT. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence, IJCAI-2016, New York, USA*, pages 766–772, 2016.

[16] C. M. Li, Z. Zhu, F. Manyà, and L. Simon. Minimum satisfiability and its applications. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence, IJCAI-2011, Barcelona, Spain*, pages 605–610, 2011.

[17] C. M. Li, Z. Zhu, F. Manyà, and L. Simon. Optimizing with minimum satisfiability. *Artificial Intelligence*, 190:32–44, 2012.

[18] H. Lin and K. Su. Exploiting inference rules to compute lower bounds for MAX-SAT solving. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence, IJCAI-2007, Hyderabad, India*, pages 2334–2339, 2007.

[19] I. Lynce, V. M. Manquinho, and R. Martins. Parallel maximum satisfiability. In *Handbook of Parallel Constraint Reasoning*, pages 61–99. Springer, 2018.

[20] V. M. Manquinho, J. Marques-Silva, and J. Planes. Algorithms for weighted Boolean optimization. In *Proceedings of the 12th International Conference on Theory and Applications of Satisfiability Testing, SAT-2009, Swansea, UK*, pages 495–508. Springer LNCS 5584, 2009.

[21] F. Manyà, S. Negrete, C. Roig, and J. R. Soler. A MaxSAT-based approach to the team composition problem in a classroom. In *Autonomous Agents and Multiagent Systems - AAMAS 2017 Workshops, Visionary Papers, São Paulo, Brazil, Revised Selected Papers*, pages 164–173. Springer LNCS 10643, 2017.

[22] R. Martins, S. Joshi, V. M. Manquinho, and I. Lynce. Incremental cardinality constraints for MaxSAT. In *Principles and Practice of Constraint Programming - 20th International Conference, CP, Lyon, France*, pages 531–548, 2014.

[23] N. Narodytska and F. Bacchus. Maximum satisfiability using core-guided MaxSAT resolution. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, Québec City, Canada*, pages 2717–2723, 2014.

[24] D. A. Plaisted and S. Greenbaum. A structure-preserving clause form translation. *Journal of Symbolic Computation*, 2:293–304, 1986.

[25] G. Tseitin. *Studies in Constructive Mathematics and Mathematical Logic, Part II*, chapter On the Complexity of Derivations in the Propositional Calculus, pages 115–125. Steklov Mathematical Institute, 1968.