On the Resiliency of Unit Propagation to Max-Resolution

André Abramé and Djamal Habet
Aix Marseille Université, CNRS, ENSAM, Université de Toulon, LSIS UMR 7296,
13397, Marseille, France
{andre.abrame,djamal.habet}@lsis.org

Abstract

At each node of the search tree, Branch and Bound solvers for Max-SAT compute the lower bound (LB) by estimating the number of disjoint inconsistent subsets (IS) of the formula. IS are detected thanks to unit propagation (UP) then transformed by max-resolution to ensure that they are counted only once. However, it has been observed experimentally that the max-resolution transformations impact the capability of UP to detect further IS. Consequently, few transformations are learned and the LB computation is redundant. In this paper, we study the effect of the transformations on the UP mechanism. We introduce the notion of *UP-resiliency* of a transformation, which quantifies its impact on UP. It provides, from a theoretical point of view, an explanation to the empirical efficiency of the learning scheme developed in the last ten years. The experimental results we present give evidences of UP-resiliency relevance and insights on the behavior of the learning mechanism.

1 Introduction

The Max-SAT problem consists in finding, for a given CNF formula, a Boolean assignment of the variables of this problem which maximizes the number of satisfied clauses.

Among the methods to solve the Max-SAT problem, Branch and Bound (BnB) algorithms (e.g. AKMAXSAT [Kügel, 2010], AHMAXSAT [Abramé and Habet, 2014a; 2014b], WMAXSATZ [Li *et al.*, 2010; 2007]) have shown their efficiency, especially on random and crafted instances. They explore the whole search space and compare, at each node of the search tree, the current number of falsified clauses plus an (under-)estimation of the ones which will become falsified (the lower bound, LB) to the best solution found so far (the upper bound, UB). If LB $\geq$ UB, then no better solution can be found in the current branch and a backtrack is performed. The estimation of the remaining inconsistencies is a key component of BnB solvers: on the one hand, it is one of the most time-consuming components and on the other hand the quality of the LB leads the backtracks and determines the number of explored nodes.

Efficient BnB Max-SAT solvers compute the lower bound by counting the disjoint inconsistent subsets (IS) of the formula. They use unit propagation based methods [Li *et al.*, 2005; 2006] to detect the inconsistent subsets. Each detected IS must be treated to ensure it will be counted only once. One of the existing treatments is based on the max-resolution rule [Larrosa and Heras, 2005; Heras and Larrosa, 2006; Bonet *et al.*, 2007]. It consists in applying several max-resolution steps on the clauses of the IS. The resulting formulas are equivalent to the original ones, thus the transformations can be learned to make the lower bound computation more incremental. However, it has been observed empirically that learning the max-resolution transformations may affect negatively the quality of the lower bound estimation. For this reason, existing solvers learn transformations selectively according to various criteria.

We study in this paper the impact of the max-resolution transformations on the UP mechanism. We show that in some cases the information which can be used by UP in the original formula are, after transformations, fragmented in several clauses. In such situation, UP may be less efficient in the transformed formula than in the original one. We introduce the notion of UP-resiliency to characterize the transformations which are not affected by fragmentation and more generally to measure the impact of the transformations on the UP mechanism. We show that according to this criterion, the most used learning scheme does not affect UP. It contributes to explain from a theoretical point of view the empirical results obtained in the last ten years on the development of inference rules [Li *et al.*, 2007]. The results of the experimental study we have performed confirm the relevance of the UP-resiliency criterion. They give insight on the behavior of the learning mechanism and open new perspectives of development.

This paper is organized as follows. In Section 2, we give the basic definitions and notations used in this paper. We present in Section 3 the IS learning mechanism and we motivate our contribution. We introduce UP-resiliency in Section 4 and present an experimental study on it in Section 5 before concluding in Section 6.

2 Definitions and Notations

A formula Φ in conjunctive normal form (CNF) defined on a set of propositional variables $X = \{x_1, \ldots, x_n\}$ is a con-

junction of clauses. A clause c_j is a disjunction of literals and a literal l is a variable x_i or its negation $\overline{x}_i$. Alternatively, a formula can be represented as a multiset of clauses $\Phi = \{c_1, \ldots, c_m\}$ and a clause as a set of literals $c_j = \{l_{j_1}, \ldots, l_{j_k}\}$. An assignment can be represented as a set I of literals which cannot contain both a literal and its negation. If x_i is assigned to $true$ (resp. $false$) then $x_i \in I$ (resp. $\overline{x}_i \in I$). I is a complete assignment if $|I| = n$ and it is partial otherwise. A literal l is said to be satisfied by an assignment I if $l \in I$ and falsified if $\bar{l} \in I$. A clause is satisfied by I if at least one of its literals is satisfied, and it is falsified if all its literals are falsified. By convention, an empty clause (denoted by $\square$) is always falsified. A subset ψ of Φ is inconsistent if there is no assignment which satisfies all its clauses. For a unit assignment $I = \{l\}$, we denote by $\Phi|_I$ the formula obtained by applying I on Φ. Formally: $\Phi|_I = \{c_j \mid c_j \in \Phi, \{l, \bar{l}\} \cap c_j = \emptyset\} \cup \{c_j \setminus \{\bar{l}\} \mid c_j \in \Phi, \bar{l} \in c_j\}$. This notation can be extended to any assignment $I = \{l_1, l_2, \ldots, l_k\}$ as follows: $\Phi|_I = (\ldots((\Phi|_{\{l_1\}})|_{\{l_2\}})\ldots|_{\{l_k\}})$. Eventually, solving the Max-SAT problem consists in finding a complete assignment which maximizes (minimizes) the number of satisfied (falsified) clauses of Φ. There exists other variants of Max-SAT (weighted, partial and weighted partial Max-SAT) which are not considered in this paper. Nevertheless, the presented results can be extended to these variants.

3 Preliminaries and Motivations

We present in this section the mechanisms of IS detection and transformation as well as the learning schemes used in recent solvers. Then, we show through an example that IS transformations can impact UP efficiency.

3.1 IS Detection

Unit propagation (UP) consists in iteratively satisfying the literals which appear in unit clauses until a conflict is found (an empty clause) or no more unit clause remains. When a conflict is detected, the clauses which have led to it by propagation form an inconsistent subset of the formula.

The propagation steps can be represented by an implication graph $G = (V, A)$ which is a directed acyclic graph where the nodes V are the propagated literals and each arrow of A is tagged with the clause causing the propagation [Marques-Silva and Sakallah, 1999]. The special node $\square$ is used to represent the empty clause.

In the remaining of this paper, we will use the following additional notations. Let us consider a sequence of propagation steps described by the implication graph $G = (V, A)$. For each literal $l_i \in V$, we denote $pred_G(l_i)$ the predecessors of l_i in G and $succ_G(l_i)$ its successors. We can define the neighborhood of l_i in G as $neigh_G(l_i) = pred_G(l_i) \cup succ_G(l_i)$. Note that this last definition can be extended to any subset V' of V as follows $neigh_G(V') = \bigcup_{l_i \in V'} neigh_G(l_i) \setminus V'$.

3.2 IS Transformation

Once detected, IS must be transformed to ensure that they will be counted only once. The most used transformation method is based on the max-resolution inference rule [Larrosa and Heras, 2005; Heras and Larrosa, 2006; Bonet *et al.*, 2007].

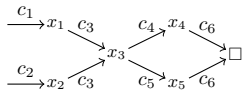
Definition 1 (Transformation by max-resolution). Let Φ be a formula and ψ an IS of Φ. For all variable x_i s.t. $\exists! c_j = \{x_i, l_{j_1}, \ldots, l_{j_s}\} \in \psi$, $x_i \in c_j$ and $\exists! c_k = \{\overline{x_i}, l_{k_1}, \ldots, l_{k_t}\} \in \psi$, $\overline{x_i} \in c_k$, we denote by $\theta(\psi, x_i)$ the set of clauses obtained from ψ after application of the max-resolution rule between c_j and c_k on x_i. Formally:

$$\theta(\psi, x_i) = (\psi \setminus \{c_j, c_k\}) \cup \{cr, cc_1, \ldots, cc_{t+s}\}$$

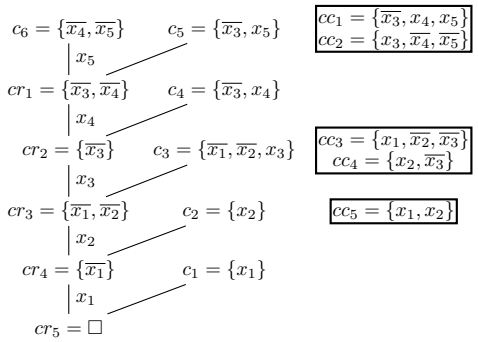
with $cr = \{l_{j_1}, \ldots, l_{j_s}, l_{k_1}, \ldots, l_{k_t}\}$ the resolvent and $cc_1 = \{x_i, l_{j_1}, \ldots, l_{j_s}, \overline{l_{k_1}}, l_{k_2}, \ldots, l_{k_t}\}$, $cc_2 = \{x_i, l_{j_1}, \ldots, l_{j_s}, \overline{l_{k_2}}, l_{k_3}, \ldots, l_{k_t}\}$, $\ldots$, $cc_t = \{x_i, l_{j_1}, \ldots, l_{j_s}, \overline{l_{k_t}}\}$, $cc_{t+1} = \{\overline{x_i}, l_{k_1}, \ldots, l_{k_t}, \overline{l_{j_1}}, l_{j_2}, \ldots, l_{j_s}\}$, $cc_{t+2} = \{\overline{x_i}, l_{k_1}, \ldots, l_{k_t}, \overline{l_{j_2}}, l_{j_3}, \ldots, l_{j_s}\}$, $\ldots$, $cc_{t+s} = \{\overline{x_i}, l_{k_1}, \ldots, l_{k_t}, \overline{l_{j_s}}\}$ the compensation clauses. For any sequence $S = (x_{i_1}, \ldots, x_{i_k})$ of variables appearing in ψ, we denote by $\Theta(\psi, S)$ the set of clauses obtained from ψ after application of the max-resolution rules on x_{i_1} then x_{i_2} and so forth. Formally $\Theta(\psi, S) = \theta(\ldots\theta(\theta(\psi, x_{i_1}), x_{i_2})\ldots, x_{i_k})$.

The following example illustrates the transformation of an IS by max-resolution.

Example 1. Let us consider a formula $\Phi_1 = \{c_1, \ldots, c_6\}$ with $c_1 = \{x_1\}$, $c_2 = \{x_2\}$, $c_3 = \{\overline{x_1}, \overline{x_2}, x_3\}$, $c_4 = \{\overline{x_3}, x_4\}$, $c_5 = \{\overline{x_3}, x_5\}$ and $c_6 = \{\overline{x_4}, \overline{x_5}\}$. The application of UP leads to the propagation sequence $(x_1@c_1, x_2@c_2, x_3@c_3, x_4@c_4, x_5@c_5)$ (meaning that x_1 is propagated by c_1, then x_2 by c_2, etc.). The clause c_6 is empty and the corresponding implication graph is shown on Fig. 1a. Hence, Φ_1 is an inconsistent subset. Its transformation by max-resolution is done as follows. Max-resolution is first applied between c_6 and c_5 on the variable x_5. The intermediary resolvent $cr_1 = \{\overline{x_3}, \overline{x_4}\}$ is produced as well as the compensation clauses $cc_1 = \{\overline{x_3}, x_4, x_5\}$ and $cc_2 = \{x_3, \overline{x_4}, \overline{x_5}\}$. The original clauses c_5 and c_6 are removed from the formula.



(a) Implication graph



(b) Max-SAT resolution steps

Figure 1: Implication graph and Max-SAT resolution steps applied on the formula Φ_1 from Example 1.

Then, max-resolution is applied between the intermediary resolvent cr_1 and the next original clause c_4 on the variable x_4 and so forth. Fig. 1b shows the max-resolution steps with in boxes the compensation clauses. After complete transformation, we obtain the formula $\Phi'_1 = \{\square, cc_1, cc_2, cc_3, cc_4, cc_5\}$ with $cc_3 = \{x_1, \overline{x_2}, \overline{x_3}\}$, $cc_4 = \{x_2, \overline{x_3}\}$ and $cc_5 = \{x_1, x_2\}$.

3.3 Existing Learning Schemes

Recent BnB solvers apply learning in the sub-part of the search tree under some conditions. Two main learning schemes are used. The first one learns a transformation only if all the intermediary resolvents are smaller than four [Heras *et al.*, 2008]. The second scheme learns transformations of the IS part matching the main following patterns [Li *et al.*, 2007]:

$$(P1)\frac{\{\{x_1,x_2\},\{x_1,\overline{x}_2\}\}}{\{\{x_1\}\}}, (P2)\frac{\{\{x_1,x_2\},\{x_1,x_3\},\{\overline{x}_2,\overline{x}_3\}\}}{\{\{x_1\},\{x_1,x_2,x_3\},\{\overline{x}_1,\overline{x}_2,\overline{x}_3\}\}},$$
$$(P3)\frac{\{\{x_1\},\{\overline{x}_1,x_2\},\{\overline{x}_2,x_3\},\ldots,\{\overline{x}_{k-1},x_k\},\{\overline{x}_k\}\}}{\{\square,\{x_1,\overline{x}_2\},\{x_2,\overline{x}_3\},\ldots,\{x_{k-1},\overline{x}_k\}\}.}$$

These patterns have been recently extended in [Abramé and Habet, 2014c]. Note that solvers do not learn only the transformations of whole IS. They can also learn the transformation of unit clause subsets (UCS) which are subsets of the formula which do not include unit clauses and which produce, after transformation by max-resolution, unit resolvent clauses.

3.4 Fragmentation Phenomenon

Two reasons are generally invoked to explain the efficiency of the learning schemes: they limit the growing of the formula size and produce small compensation clauses which are more likely to be used by UP. However, experimental analysis [Abramé and Habet, 2014c] shows that increasing learning slightly may deteriorate significantly the solver performance. The reasons cited above cannot explain this behavior.

Abramé and Habet [2014c] have also shown that the max-resolution transformations may result in a "fragmentation" of the information contained in the original clauses of the formula. Clauses which can have been used in UP are, after transformation by max-resolution, fragmented in two (or more) clauses. The original information can only be retrieved by applying max-resolution between them.

Example 2. Let us consider the formula $\Phi_2 = \{c_1, c_2, c_3, c_4, c_5, c_6, c_7\}$ with $c_6 = \{\overline{x_4}, x_6\}$ and $c_7 = \{\overline{x_5}, \overline{x_6}\}$ ($c_1, \ldots, c_5$ are the same as in Example 1). The application of UP leads to the propagation sequence $(x_1@c_1, x_2@c_2, x_3@c_3, x_4@c_4, x_5@c_5, x_6@c_6)$ and the clause c_7 is empty. The corresponding implication graph is shown in Fig. 2. If we apply max-resolution according to the sequence $(x_4, x_6, x_5, x_3, x_2, x_1)$, we obtain the formula $\Phi'_2 = \{\square, cc_3, cc_4, cc_5, cc_6, cc_7, cc_8, cc_9\}$ with $cc_6 = \{\overline{x_3}, x_4, \overline{x_6}\}$, $cc_7 = \{x_3, \overline{x_4}, x_6\}$, $cc_8 = \{\overline{x_3}, x_5, x_6\}$ and $cc_9 = \{x_3, \overline{x_5}, \overline{x_6}\}$ (cc_3, cc_4 and cc_5 are the same as in Example 1).

Let us assign x_4 and x_5 to $true$. In the original formula Φ_2, the subset $\{c_6, c_7\}$ is inconsistent and the application of UP leads to the following propagation sequence $(x_1@c_1, x_2@c_2, x_3@c_3)$. In the transformed formula Φ'_2, the clauses cc_7 and cc_9 are reduced and become respectively $\{x_3, x_6\}$ and $\{x_3, \overline{x_6}\}$ but no propagation steps can be performed. Note that if we apply max-resolution between this two clauses on x_6 we obtain the unit resolvent $\{x_3\}$ but the sole UP cannot exploit these two clauses. Thus we can say that the information which may have led to the propagation of x_3 is fragmented.

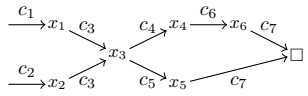


Figure 2: Implication graph of Example 2.

When such transformations are learned, it may affect the capability of UP to detect IS in the lower nodes of the search tree. Thus the LB estimation may be less accurate and the solvers may explore more nodes.

4 UP-Resiliency

We study in this section the resiliency of UP to transformations. We introduce the notion of *UP-resiliency* of a transformation to characterize the transformations which have no negative impact on UP and to quantify this impact. We discuss the UP behavior on the formula resulting from UP-resilient transformations and we show that the most used learning scheme does not affect UP.

4.1 In an Implication Graph

When fragmentation occurs, the compensation clauses which may propagate a literal l_i of the original implication graph G contain additional literals which are not in the original neighborhood of l_i in G. Thus, to detect if a transformation is not affected by the fragmentation phenomenon, we can rely on the capability of UP to propagate the literals of the original implication graph G when only their neighborhood literals in G are set to $true$. We said that such transformations are UP-resilient.

Definition 2 (UP-resiliency in an implication graph). Let Φ be a formula, ψ an IS detected by the propagation steps described by the implication graph $G = (V, A)$ and S a sequence of variables appearing in ψ. For a literal $l_i \in V$, we say that the transformation $\Theta(\psi, S)$ is UP-resilient for l_i in G iff $\square \in neigh_G(l_i)$ or l_i can be propagated in $\Theta(\psi, S)|_{neigh_G(l_i)}$.

We say that $\Theta(\psi, S)$ is UP-resilient in G for a subset V' of V iff it is UP-resilient $\forall l_i \in V'$ and that it is UP-resilient in G iff it is UP-resilient for V.

Note that the neighborhoods which include the special node $\square$ are not valid assignments. All transformations are considered UP-resilient for literals with such neighborhood.

4.2 Generalization

The definition of the UP-resiliency given in the previous section depends on the neighborhood of the literals in the implication graph. However, the same IS can be detected by sev-

eral sequences of propagation steps which can be described by distinct implication graphs.

Example 3. Let us consider the formula Φ_1 from Example 1. In addition to the one presented in the original example, two other sequences of propagation steps can lead to the detection of the inconsistent subset while changing the value given to the propagated variables: $(x_1@c_1, x_2@c_2, x_3@c_3, x_4@c_4, \overline{x_5}@c_6)$ and $(x_1@c_1, x_2@c_2, x_3@c_3, x_5@c_5, \overline{x_4}@c_6)$. Note that we ignore the propagation sequences varying only by the order of the propagation steps.

With the previous definition, the UP-resiliency of a transformation may vary depending on how the IS have been detected. To overcome this limitation, we propose to consider all the possible ways (i.e. sequences of propagation steps) to detect the IS. Let us first define the set of the possible neighborhoods of a literal appearing in an inconsistent subset.

Definition 3 (Possible neighborhoods). Let Φ be a formula and ψ an IS. For a literal l_i appearing in ψ clauses, we can define its possible neighborhoods as $pneigh_\psi(l_i) = \{neigh_G(l_i),\ \forall$ implication graph $G = (V, A)$ of ψ s.t. $l_i \in V\}$. This definition can be naturally extended to any set of literals.

Then we can give a general definition of the UP-resiliency which does not depend anymore of the propagation steps which have led to the IS discovery.

Definition 4 (UP-resiliency). Let Φ be a formula, ψ an IS and S a sequence of variables appearing in ψ. For a literal l_i appearing in ψ, we say that the transformation $\Theta(\psi, S)$ is UP-resilient for l_i iff for all $n_k \in pneigh_\psi(l_i)$: $\square \in n_k$ or l_i can be propagated in $\Theta(\psi, S)|_{n_k}$.

We say that $\Theta(\psi, S)$ is UP-resilient for a set of literals L appearing in ψ iff it is UP-resilient $\forall l_i \in L$ and that it is UP-resilient iff it is UP-resilient for all the literals appearing in ψ.

Similarly, we can quantify a transformation impact on UP by defining its percentage of UP-resiliency as the percentage of couples (l_i, n_k) s.t. $\square \in n_k$ or l_i can be propagated in $\Theta(\psi, S)|_{n_k}$.

4.3 Impact on the IS Detection

One of the most interesting properties of the UP-resilient transformations is the capability to retrieve the propagations which are not necessary anymore to an inconsistent subset. We have seen that if a transformations is UP-resilient for a literal l_i, then l_i can be propagated in the transformed formula when the literals of one of its possible neighborhoods is set to $true$. We now show that this property can be extended to sets of literals, i.e. that if a transformation is UP-resilient for a set of literals L, then the literals of L can be propagated in the transformed formula if the literals of one of L possible neighborhoods are set to $true$.

Property 1. Let Φ be a formula, ψ an IS, S a sequence of variables appearing in ψ and L a set of literals appearing in ψ. If the transformation $\Theta(\psi, S)$ is UP-resilient for L then for all $n_k \in pneigh_\psi(L)$: $\square \in n_k$ or all L literals can be propagated in $\Theta(\psi, S)|_{n_k}$.

Proof. Let ψ be an IS detected thanks to the propagation steps described by a implication graph G and $S = (x_{i_1}, \ldots, x_{i_m})$ a sequence of ψ variables. We assume that the partial transformation $\Theta(\psi, (x_{i_1}, \ldots, x_{i_h}))$ (with $h < m$) has already be applied. We denote by $G|_{(x_{i_1}, \ldots, x_{i_h})}$ the updated implication graph obtained by replacing the resolved clauses by the intermediary resolvent produced. For every literal l_{i_j} s.t. $l_{i_j}, \overline{l_{i_j}} \notin \{x_{i_1}, \ldots, x_{i_h}\}$ the compensation clauses providing UP-resiliency for l_{i_j} in G can only be obtained by (we assume that such clause has not yet been obtained and that $\square \notin neigh_G(l_{i_j})$): (1) applying max-resolution on l_{i_j} if $\square \notin succ_{G|_{(x_{i_1}, \ldots, x_{i_h})}}(l_{i_j})$ or (2) applying max-resolution on a successor l_{i_k} of l_{i_j} if $\square \in succ_G(l_{i_k})$. These two points can be easily proved by observing the compensation clauses obtained when we transform clauses containing l_{i_j} or $\overline{l_{i_j}}$. In any other cases, the compensation clauses produced do not contain l_{i_j} or include literals which are not in l_{i_j} neighborhood in G. Also note that we present here only the information which is relevant to our demonstration. These conditions are necessary but not sufficient to obtain l_i UP-resiliency.

We can now prove the property. Let us proceed by induction on the size of a set of literals L appearing in ψ.
Base: If $|L| = 1$ then obviously the property is verified.
Inductive step: We assume that the property holds for every L' s.t. $|L'| < n$ and we now consider a set of literals L s.t. $|L| = n$ and an implication graph $G = (V, A)$ of ψ s.t. $L \in V$. Let us consider the last literal l_{i_j} of L in S.

If the UP-resiliency in G for l_{i_j} has been obtained by applying max-resolution on l_{i_j} (case 1), then the compensation clauses providing UP-resiliency for l_{i_j} cannot contain literals from $L \setminus \{l_{i_j}\}$. Thus $neigh_G(L)$ is sufficient to propagate l_{i_j} in the transformed formula. By assumption we know that $neigh_G(L \setminus \{l_{i_j}\})$ propagate $L \setminus \{l_{i_j}\}$ in the transformed formula thus the property holds for L.

Otherwise, l_{i_j} UP-resiliency have been obtained thanks to case (2). We consider the first literal l_{i_k} of L which have obtained UP-resiliency by applying max-resolution on one of its successors l_{i_o} (case (2)). We can distinguish to cases:

- if $neigh_{G|_{(x_{i_1}, \ldots, x_{i_{o-1}})}}(l_{i_k}) \cap L = \emptyset$, then $neigh_G(L)$ is sufficient to propagate l_{i_k} in the transformed formula. As previously, by assumption we know that $neigh_G(L \setminus \{l_{i_k}\})$ propagates $L \setminus \{l_{i_k}\}$ in the transformed formula thus the property holds for L.
- otherwise, $\forall l_{i_p} \in neigh_{G|_{(x_{i_1}, \ldots, x_{i_{o-1}})}}(l_{i_k}) \cap L$, we know that the clauses providing UP-resiliency for l_{i_p} have not yet be obtained. After application of max-resolution on l_{i_o}, we obtain the transformed implication graph $G|_{(x_{i_1}, \ldots, x_{i_o})}$ and $\square \in neigh_{G|_{(x_{i_1}, \ldots, x_{i_o})}}(l_{i_p})$. If $\square \in neigh_G(l_{i_p})$ then $\square \in neigh_G(L)$ and the property holds. Otherwise, the transformation cannot be UP-resilient in G for l_{i_p} and it contradicts the initial assumption.

The same reasoning can be made for every implication graph G of ψ, thus the property holds for any set of literals of size n. □

When a subset ψ' of an IS ψ is not necessary anymore (for

instance when ψ is not minimal), this property ensure that UP can perform the same propagations in the transformed formula than in the original clauses of ψ'.

Example 4. Let us consider again the formula Φ_1 from Example 1 and its UP-resilient transformation in Φ'_1. If we assign x_4 and x_5 to true, the clause c_6 is falsified while c_4 and c_5 are satisfied. The application of UP leads to the following propagation steps: $(x_1@c_1, x_2@c_2, x_3@c_3)$. If we apply UP on the transformed formula Φ'_1, we obtain the propagation steps: $(x_3@cc_2, x_2@cc_4, x_1@cc_3)$. Note that the same propagations are performed in the transformed formula as in the original one. The same propagations can be done in the formula Φ_2 with the same assignments. But if we consider the formula Φ'_2 obtained by the non UP-resilient transformation presented in Example 2, no propagation can be performed.

4.4 UP-Resiliency of the Existing Patterns

Empirical results [Li *et al.*, 2007] have shown that the existing learning schemes are efficient, but until now there was no theoretical results to explain it. The following property shows that there exist transformations which are UP-resilient for the existing patterns presented in Section 3.3.

Property 2. Let Φ be a formula and ψ an IS. For any $\psi' \subset \psi$ such that ψ' matches one of the pattern P1, P2 or P3 there exists a sequence S of ψ variables s.t. $\Theta(\psi, S)$ is UP-resilient for ψ'.

Proof. (sketch) For each $l_i \in \psi'$ and each $n_k \in pneigh_\psi(l_i)$, it is simple to check that $\square \in n_k$ or l_i can be propagated in $\Theta(\psi, S)|_{n_k}$. □

5 Experimental Study

We have implemented a simple procedure to compute the percentage of UP-resiliency of the transformations. We generate all the neighborhoods of the literals and then we check if each literal can be propagated when each one of its neighborhoods is satisfied. This naive implementation is time consuming and increases the solving time of 25% in average. Its purpose is to evaluate the IS transformation and learning mechanisms, not to be competitive. The implementation is performed in the solver AHMAXSAT, which was ranked first in three out of nine categories during the last Max-SAT Evaluation[1].

We have run all the variants presented below on the unweighted and weighted random and crafted instances of the Max-SAT Evaluation 2013. We did not include partial and industrial instances. We have made 66,501 separate runs of the solver. In our opinion, it is sufficient to show the impact of the studied components on the solver behavior. Nevertheless, the presented results can be extended to these instance categories. All the experiments are performed on machines equipped with Intel Xeon 2.4 Ghz processors and 24 Gb of RAM and running under a GNU/Linux operating system. The cutoff time is fixed to 1800 seconds per instance.

[1] http://maxsat.ia.udl.cat:81/13

5.1 Impact of the Max-Resolution Application Order on the UP-resiliency

We first evaluate the impact of the order of application of the max-resolution steps on the UP-resiliency of the transformation. We compare two variants:

- AHMAXSAT-RPO applies the max-resolution steps in reverse propagation order
- AHMAXSAT-SIR uses the smallest resolvent heuristic described in [Abramé and Habet, 2014a].

The results are presented in Table 1. We can observe that the average percentage of UP-resiliency of the transformations is significantly higher with the SIR heuristic (54% vs. 60%). Consequently, less decisions are necessary to solve the instances (123,028 vs. 102,088 in average) and the solving time is reduced (135.5 vs. 108.4 seconds in average). It shows that max-resolution application order has an important impact on the UP-resiliency of the transformations, and thus on the solver performance.

Table 1: Evaluation of the impact of the max-resolution application order on the solver behavior. Columns $S(T)$, D and UPR give respectively: the number of solved instances with in bracket the average solving time, the average number of decisions and the average percentage of UP-resiliency of the transformations. Columns marked with a star consider only the instances solved by both solver variants.

Categories	#	RPO			SIR		
		S(T)	D*	UPR*	S(T)	D*	UPR*
ms_crafted	167	156(110.5)	100618	53%	156(94.3)	84148	57%
ms_random	378	299(187.3)	208420	52%	302(149.6)	171236	58%
wms_crafted	116	83(61.1)	62987	74%	83(47.4)	58110	80%
wms_random	160	160(101.1)	14409	48%	160(76)	11474	55%
Overall	821	698(135.5)	123028	54%	701(108.4)	102088	60%

5.2 Impact of the Existing Learning Schemes on the UP-resiliency

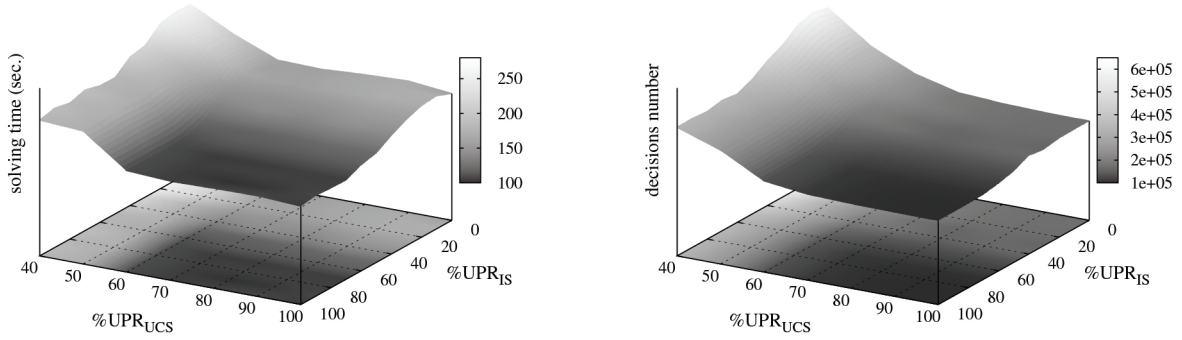
In the second set of experiments, we evaluate the impact of the existing learning schemes on the learned transformation UP-resiliency. We consider the following variants:

- AHMAXSAT-IRS uses the learning scheme of MINIMAXSAT [Heras *et al.*, 2008]. A transformation is learned if all the intermediary resolvents contain less than four literals.
- AHMAXSAT-PAT learns the transformations when IS match the patterns presented in Section 3.3.
- AHMAXSAT-PAT+ learns the transformations when IS match the patterns presented in Section 3.3 or the extended sets of patterns presented in [Abramé and Habet, 2014c].
- AHMAXSAT-UPR learns the transformations of the IS and UCS which are UP-resilient.

Table 2 shows the results obtained. We can first observe that the IRS learning scheme behaves very differently than the PAT and PAT+ ones. It learns in average 80% of the transformations (columns %L) while PAT and PAT+ learn less than 20% of them. The average percentage of UP-resiliency of the transformations learned by the IRS learning scheme (columns $\%UPR_L$) is relatively low (78%) compared to the ones of PAT

Table 2: Evaluation of the impact of the learning schemes on AHMAXSAT behavior. Columns $S(T)$, $\%L$ and $\%UPR_L$ give $\times 10^3$),

Categories	#	IRS				PAT				PAT+				UPR		
		S(T)	D	%L	$\%UPR_L$	S(T)	D	%L	$\%UPR_L$	S(T)	D	%L	$\%UPR_L$	D	%L	$\%UPR_L$
ms_crafted	167	58(160,3)	655	89%	82%	156(106,1)	111	14%	100%	156(94,3)	84	17%	99%	81	19%	100%
ms_random	378	182(378,0)	1303	80%	80%	301(188,6)	228	13%	100%	302(162,9)	220	16%	98%	220	15%	100%
wms_crafted	116	76(38,1)	67	90%	84%	83(83,8)	121	30%	100%	83(47,4)	58	40%	98%	53	46%	100%
wms_random	160	110(406,1)	113	70%	69%	157(88,2)	13	8%	100%	157(76,0)	11	10%	96%	12	9%	100%
Overall	821	426(295,0)	687	80%	78%	697(135,1)	140	14%	100%	698(114,3)	124	18%	97%	122	18%	100%



(a) Average solving time

(b) Average number of decisions

Figure 3: Impact of the percentage of non UP-resilient neighborhoods allowed in the learning of IS and UCS transformations.

and PAT+ (respectively 100% and 96%). Consequently, UP detects IS less efficiently with the IRS learning scheme than with the PAT and PAT+ ones. Thus, the solver explores four times more nodes of the search tree (columns D) than with PAT or PAT+ and it solves less instances with a higher average solving time (columns S(T)). From these results, we can conclude that the IRS learning scheme, as it is implemented in AHMAXSAT-IRS, does not control efficiently the impact of the transformations on the UP mechanism.

The comparison of the PAT and PAT+ learning schemes shows a slight increase of percentage of transformations learned for the latter (columns %L) while keeping a high percentage of UP-resiliency (96%). Consequently, the average number of decisions as well as the average solving time are slightly reduced.

It is also interesting to observe that the percentage of transformations learned with the PAT+ learning scheme is similar to the one obtained with the UPR learning scheme. The detailed results show that on the crafted instances, the PAT+ learning scheme does not detect all the UP-resilient transformations. From this observation, we can conclude that the PAT+ learning scheme can be improved.

5.3 UP-resiliency Based Learning Schemes

We have implemented a new learning scheme based on the minimum percentage of UP-resiliency allowed in the learned transformations of IS (noted $\%UPR_{IS}$) and UCS (noted $\%UPR_{UCS}$). We have tested this new learning scheme with $\%UPR_{IS}$ and $\%UPR_{UCS}$ ranging respectively from 0 to 100 and from 40 to 100. Results are presented in Fig. 3.

We can observe that the best results are obtained with $\%UPR_{IS}$ and $\%UPR_{UCS}$ ranging from 60% to 100%. On these ranges of values, the average UP-resiliency percentage of the learned transformation is always higher than 90%. With values lower than 60%, the average number of decisions increases importantly and so does the average solving time. It shows that UP-resiliency quantifies accurately the transformation impact on UP efficiency.

6 Conclusions

We have introduced in this paper the notion of UP-resiliency of a transformation which quantifies the impact of the max-resolution rule on UP. We have shown that, according to UP-resiliency criterion, the most used learning scheme based on patterns does not affect UP. It contributes to explain theoretically the learning scheme efficiency which was proved only empirically until now. The results of the experimental study we have performed provide evidences of the UP-resiliency relevance. They also show that (1) the order of application of the max-resolution steps has an impact on the UP-resiliency of the transformations and (2) the existing learning schemes do not capture all the UP-resilient transformations.

These results open several perspectives. Among them, it would be interesting to develop new max-resolution application orders to increase the percentage of UP-resiliency of the transformations. The existing learning schemes can also

be improved, either by finding a characterization of the UP-resilient transformations which can be efficiently checked or by completing the set of patterns used in the current leaning schemes. Now that we have a better understanding of the behavior of the max-resolution transformations, we can consider applying learning not only in the sub-part of the search tree but also in the upper part. The expected benefits would be an even more incremental LB computation and a better consideration of the structural properties of the instances.

As shown in the Max-SAT Evaluations, BnB solvers perform poorly on structured instances such as industrial ones. In our opinion, this is mainly due to their inability to consider the structural properties of these instances in the exploration of the search space. An extended learning mechanism as described above may allow to guide the exploration of the search tree by using the information learned as it is done in modern SAT solvers [Marques-Silva and Sakallah, 1999; Eén and Sörensson, 2003]. Thus, the UP-resiliency property presented in this paper may be a significant step towards the improvement of BnB solver performance on industrial instances.

References

[Abramé and Habet, 2014a] André Abramé and Djamal Habet. Efficient application of max-sat resolution on inconsistent subsets. In Barry O'Sullivan, editor, *Proceedings of the 20th International Conference on Principles and Practice of Constraint Programming (CP 2014)*, volume 8656 of *Lecture Notes in Computer Science*, pages 92–107. Springer International Publishing, 2014.

[Abramé and Habet, 2014b] André Abramé and Djamal Habet. Local max-resolution in branch and bound solvers for max-sat. In *Proceedings of the 26th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2014)*, pages 336–343, 2014.

[Abramé and Habet, 2014c] André Abramé and Djamal Habet. On the extension of learning for max-sat. In Ulle Endriss and João Leite, editors, *Proceedings of the 7th European Starting AI Researcher Symposium (STAIRS 2014)*, volume 241 of *Frontiers in Artificial Intelligence and Applications*, pages 1–10. IOS Press, 2014.

[Bonet *et al.*, 2007] María Luisa Bonet, Jordi Levy, and Felip Manyà. Resolution for max-sat. *Artificial Intelligence*, 171(8-9):606–618, 2007.

[Eén and Sörensson, 2003] Niklas Eén and Niklas Sörensson. An extensible sat-solver. In Enrico Giunchiglia and Armando Tacchella, editors, *Proceedings of the 6th International Conference on Theory and Applications of Satisfiability Testing (SAT 2003)*, volume 2919 of *Lecture Notes in Computer Science*, pages 502–518. Springer, 2003.

[Heras and Larrosa, 2006] Federico Heras and Javier Larrosa. New inference rules for efficient max-sat solving. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI 2006)*, pages 68–73. AAAI Press, 2006.

[Heras *et al.*, 2008] Federico Heras, Javier Larrosa, and Albert Oliveras. Minimaxsat: An efficient weighted max-sat solver. *Journal of Artificial Intelligence Research*, 31:1–32, 2008.

[Kügel, 2010] Adrian Kügel. Improved exact solver for the weighted max-sat problem. In Daniel Le Berre, editor, *Proceedings of the 1st Pragmatics of SAT Workshop (POS 2010)*, volume 8 of *EasyChair Proceedings in Computing*, pages 15–27, 2010.

[Larrosa and Heras, 2005] Javier Larrosa and Federico Heras. Resolution in max-sat and its relation to local consistency in weighted csps. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI 2005)*, pages 193–198. Professional Book Center, 2005.

[Li *et al.*, 2005] Chu Min Li, Felip Manyà, and Jordi Planes. Exploiting unit propagation to compute lower bounds in branch and bound max-sat solvers. In Peter van Beek, editor, *Principles and Practice of Constraint Programming (CP 2005)*, volume 3709 of *Lecture Notes in Computer Science*, pages 403–414. Springer Berlin / Heidelberg, 2005.

[Li *et al.*, 2006] Chu Min Li, Felip Manyà, and Jordi Planes. Detecting disjoint inconsistent subformulas for computing lower bounds for max-sat. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI 2006)*, pages 86–91. AAAI Press, 2006.

[Li *et al.*, 2007] Chu Min Li, Felip Manyà, and Jordi Planes. New inference rules for max-sat. *Journal of Artificial Intelligence Research*, 30:321–359, 2007.

[Li *et al.*, 2010] Chu Min Li, Felip Manyà, Nouredine Ould Mohamedou, and Jordi Planes. Resolution-based lower bounds in maxsat. *Constraints*, 15(4):456–484, 2010.

[Marques-Silva and Sakallah, 1999] João Marques-Silva and Karem A. Sakallah. Grasp: A search algorithm for propositional satisfiability. *IEEE Transactions on Computers*, 48(5):506–521, August 1999.