Impact of Extended Clauses on Local Search Solvers for Max-SAT

Federico Heras
Universitat Pompeu Fabra, Barcelona, Spain

Keywords: Max-SAT, Stochastic Local Search, Optimization, Constraint Satisfaction.

Abstract: Previous research has demonstrated that several techniques based on the *resolution rule* for *Max-SAT* are effective in improving results and boost the search, either as a preprocessing step or when embedded into specific Max-SAT solving algorithms, such as branch-and-bound and Stochastic Local Search (abbreviated *SLS*) algorithms. These techniques typically lead to a simplified and reduced Max-SAT formula, thereby enabling the algorithms to find solutions more efficiently. In this paper, we take a different approach by introducing a preprocessing step that, in contrast to prior methods, increases the size of the Max-SAT formula based on the *Extension Rule*. Our objective is to examine how this expansion of the problem instance impacts the performance of SLS algorithms. The empirical results indicate that for a subset of SLS algorithms, this approach yields improved solutions. This finding is significant as it challenges the conventional wisdom that smaller, simplified formulas are always better for all kind of solvers.

1 INTRODUCTION

(Weighted) Max-SAT is a well-known *NP-Hard* problem. The objective of the Max-SAT problem is to identify an assignment of variables that satisfies the maximum number (or weights) of the provided set of clauses. It is recognized as one of the fundamental problems in combinatorial optimization, with many significant problems naturally formulable as Max-SAT instances. Among others, they include problems like *Maximum One*, *Maximum Cut* (*Max-CUT*), *Maximum Clique* (*Max-Clique*) with many practical applications in bioinformatics, physics and electronic markets (Bansal and Bafna, 2008; Guerri and Milano, 2003; Pardalos and Rebennack, 2010; Strickland et al., 2005) and other industrial problems.

A *sound* and *complete* inference method for Max-SAT was introduced (Larrosa et al., 2008) called the resolution rule for Max-SAT. The implementation of restricted rules based on the resolution for Max-SAT has been proven advantageous across various algorithmic approaches, whether utilized as a preprocessing step or integrated directly into the algorithm. These include *branch-and-bound algorithms* (Heras et al., 2008; Larrosa et al., 2008; Li et al., 2007a; Heras and Larrosa, 2008; Heras and Bañeres, 2013; Cherif et al., 2020), *stochastic local search* (SLS) algorithms (Heras and Bañeres, 2010; Abramé and Habet, 2012), and algorithms that rely on iteratively calling to a SAT oracle (Heras and Marques-Silva, 2011; Py et al., 2022). In those works, the application of 1 step or several steps of the resolution rule resulted in a simplified and smaller Max-SAT formula.

Stochastic Local Search (*SLS*) algorithms employ heuristic approaches that begin by selecting a point within the search space and then iteratively transition from the current solution to a neighboring candidate solution. Certain methods (Anbulagan et al., 2005; Heras and Bañeres, 2010; Abramé and Habet, 2012) demonstrate that employing incomplete inference can enhance the performance of SLS algorithms, particularly those structured on the WalkSat architecture (Selman et al., 1994).

In this paper, we take a different approach by applying a transformation that, contrary to prior methods, increases the size of the Max-SAT formula. Essentially, we will apply the *Extension Rule* (*ER*) (Larrosa and Schiex, 2003; Atserias and Lauria, 2019; Rollon and Larrosa, 2022) which basically takes one clause of the original formula and creates two *extended clauses* from it, which means replacing the original clause by two copies of such clause but with an additional literal. This rule is also referred to as *split rule*. Then, we propose a novel preprocessor based on applying the Extension Rule for *all* clauses in the formula.

We assess the effect of such preprocessor with several experiments including different benchmarks

Heras, F.
Impact of Extended Clauses on Local Search Solvers for Max-SAT.
DOI: 10.5220/0013078200003890
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 17th International Conference on Agents and Artificial Intelligence (ICAART 2025) - Volume 3*, pages 78-85
ISBN: 978-989-758-737-5; ISSN: 2184-433X
Proceedings Copyright © 2025 by SCITEPRESS – Science and Technology Publications, Lda.

and SLS algorithms. The empirical investigation demonstrates that for a subset of SLS algorithms and benchmarks, this approach produces better results.

The structure of this paper is as follows. We begin by introducing some preliminary concepts. Next, we present the Extension Rule and its connection to the *Neighborhood Rule*. We then introduce a new preprocessor based on this rule. Following this, we conduct empirical experiments to evaluate the preprocessor's effectiveness. We then discuss how our work relates to previous research. Finally, we conclude with summarizing our findings and future research directions.

2 PRELIMINARY CONCEPTS

In this section, we formally introduce the Max-SAT problem and related notation.

The Max-SAT Problem. $X = \{x_1, x_2, \ldots, x_n\}$ is a set of Boolean variables. A *literal* is either a variable x_i or its negation $\bar{x}_i$. The variable to which a literal l refers is denoted by $var(l)$. Given a literal l, its negation $\bar{l}$ is $\bar{x}_i$ if l is x_i and it is x_i if l is $\bar{x}_i$.

A *clause C* is a disjunction of literals. Capital letters will represent clauses. The *size* of a clause, noted $|C|$, is the number of literals that it has. Clauses of size one and two are called *unit* and *binary* clauses, respectively. A formula in *conjunctive normal form* (CNF) is a set of clauses.

An *assignment* is a set of literals $A = \{l_1, l_2, \ldots, l_k\}$ such that for all $l_i \in A$, its variable $var(l_i) = x_i$ is assigned to value t (*true*) or f ($false$). If variable x_i is assigned to t (f), literal x_i ($\bar{x}_i$) is *satisfied* and literal $\bar{x}_i$ (x_i) is *falsified*. If all variables in X are assigned, the assignment is called *complete*, otherwise it is called *partial*. An assignment *satisfies* a literal iff the literal belongs to the assignment, the assignment satisfies a clause iff one or more of its literals are satisfied and the assignment *falsifies* a clause iff the clause contains the negation of all its literals. The *empty clause* $\square$ cannot be satisfied by definition.

A *weighted* clause is a pair (C, w), where C is a clause and the *weight w* is the cost of its falsification. A *weighted* formula in *conjunctive normal form* (WCNF) $\mathcal{F}$ is a set of weighted clauses. Many real problems contain clauses that *must* be satisfied. We denote these clauses *hard* and a special weight $\top$ is associated to them. Note that, any weight $w \geq \top$ indicates that the associated clause must be necessarily satisfied. Thus, we can replace w by $\top$ without changing the problem. Consequently, we can assume all weights are in the interval $[0, \top]$. Non-hard clauses are called *soft* clauses. We use the symbol $\equiv$ (e.g., $\mathcal{F} \equiv \mathcal{F}'$) to denote that two formulas are equivalent.

A *model* is a complete assignment that satisfies all hard clauses. The *cost of an assignment* is the sum of the weights of the falsified clauses. Given a WCNF formula, the objective of the *Weighted* Max-SAT is to find a model with minimum cost.

Example 1. *Let be $\mathcal{F}$ a weighted formula with 3 clauses $\mathcal{F} = \{(\bar{x}_1, 1), (\bar{x}_2, 1), (x_1 \vee x_2, \top)\}$. Clauses $(\bar{x}_1, 1)$ and $(\bar{x}_2, 1)$ are unit soft clauses with weight 1. Clause $(x_1 \vee x_2, \top)$ is a binary hard clause. The assignment $A_1 = \{\bar{x}_1, \bar{x}_2\}$ falsifies the hard clause $(x_1 \vee x_2, \top)$ and for this reason A_1 is* not *a model. Assignment $A_2 = \{x_1, x_2\}$ satisfies the hard clause and falsifies both soft unit clauses. Hence, assignment A_2 is a model with cost 2. Finally, the assignment $A_3 = \{x_1, \bar{x}_2\}$ satisfies the hard clause and only falsifies one soft clause. A_3 is an optimal model with cost 1.*

Let u and w be two weights. Their sum is defined as $u \oplus w = min\{u + w, \top\}$ in order to keep the result within the interval $[0, \top]$. Assuming $u \geq w$, their subtraction is defined as $u \ominus w = u - w$ if $u \neq \top$ and $u \ominus w = \top$ otherwise. The De Morgan rule is unsound for Max-SAT. Instead, the following rule should be repeatedly used until the conjunctive normal form is achieved: $(A \vee \overline{l \vee C}, w) \equiv \{(A \vee \bar{C}, w), (A \vee \bar{l} \vee C, w)\}$.

The empty clause may appear in a formula. If its weight is $\top$, i.e. $(\square, \top)$, the formula does not have any model. Following (Larrosa et al., 2008), the resolution rule for Max-SAT is $\{(x \vee A, u), (\bar{x} \vee B, w)\} \equiv \{(A \vee B, m), (x \vee A, u \ominus m), (\bar{x} \vee B, w \ominus m), (x \vee A \vee \bar{B}, m), (\bar{x} \vee \bar{A} \vee B, m)\}$, where $m = \min\{u, w\}$. $(x \vee A, u)$ and $(\bar{x} \vee B, w)$ are called *clashing* clauses; $(A \vee B, m)$ is called the *resolvent*; and $(x \vee A, u \ominus m)$ and $(\bar{x} \vee B, w \ominus m)$ are called as *posterior clashing* clauses. Finally, $(x \vee A \vee \bar{B}, m)$ and $(\bar{x} \vee \bar{A} \vee B, m)$ are called *compensation* clauses.

Note that in Max-SAT truth tables are tables with a cost associated to each truth assignments. A brute-force solving method consists in computing the truth table of the input for mula and finding the minimal cost model. For instance the cost for formula $\mathcal{F} = \{(x_1 \vee x_2, 1), (x_2, 2), (\bar{x}_1 \vee \bar{x}_2, \top)\}$ is as in Table 1:

Table 1: Truth table example.

x_1	x_2	$\mathcal{F}$
f	f	3
f	t	0
t	f	2
t	t	$\top$

First and second columns are the variables and the respective values they can take, as usual in truth tables. The third column shows the sum of weights of unsatisfied clauses in $\mathcal{F} = \{(x_1 \vee x_2, 1), (x_2, 2), (\bar{x}_1 \vee \bar{x}_2, \top)\}$ for each possible assignment. Hence, assignment $A_1 = \{x_1, \bar{x}_2\}$ is the only optimal model with cost 0.

The Max-Cut Problem. In what follows, we explain how to encode the Max-Cut problem as Max-SAT as we will use some Max-Cut instances in the experimental section. The *Max-Cut* problem consists in finding a cut of maximum size. It can be easily modeled as Max-SAT. One variable x_i is associated to each graph vertex v_i. The value $true$ ($false$) indicates that vertex v_i belongs to U (to $V - U$). For each edge (v_i, v_j), there are two clauses $(x_i \vee x_j, 1)$ and $(\bar{x}_i \vee \bar{x}_j, 1)$. Given a complete assignment, the number of violated clauses is $|E| - S$ where S is the size of the cut associated to the assignment.

3 EXTENDED CLAUSES

In this section, we introduce well-known *Neighborhood* and *Extension* rules, which are indeed deeply related one with the other. Then we propose a novel preprocessor based on the *Extension Rule*.

3.1 Neighborhood Resolution

The *Neigborhood resolution* (*NRES*) (Larrosa et al., 2008) takes two clauses containing the same literals C, with an additional literal h that appears in the positive h and in the negative $\bar{h}$ form on each clause and results in the following:

$$\{(C \vee h, w_1), (C \vee \bar{h}, w_2) \equiv$$
$$\{(C, m), (C \vee h, w_1 \ominus m), (C \vee \bar{h}, w_2 \ominus m)\}$$

where $m = min(w_1, w_2)$. Note that if weights $w_1 = w_2 = m$ we would obtain simply:

$$\{(C \vee h, m), (C \vee \bar{h}, m) \equiv \{(C, m)\}$$

The simplification capability of neighborhood resolution (NRES) is shown in the following example, where several steps of resolution are applied based on NRES, sometimes referred to as *hyper-resolution*.

Example 2. *Let $\mathcal{F}$ be a WCNF formula with clauses $\mathcal{F} = \{(x_1 \vee x_2 \vee x_3, 1), (x_1 \vee x_2 \vee \bar{x}_3, 1), (x_1 \vee \bar{x}_2 \vee x_3, 1), (x_1 \vee \bar{x}_2 \vee \bar{x}_3, 1), (\bar{x}_1, 1)\}$. The application of the Neighborhood Resolution rule to the first and second clause, and to the third and fouth clause in $\mathcal{F}$ produces the following* equivalent *formula $\mathcal{F}' = \{(x_1 \vee x_2, 1), (x_1 \vee \bar{x}_2, 1), (\bar{x}_1, 1)\}$ (i.e. $\mathcal{F} \equiv \mathcal{F}'$). Now, we can apply again the NRES rule between the first two clauses, resulting in the following* equivalent *formula $\mathcal{F}'' = \{(x_1, 1), (\bar{x}_1, 1)\}$ (i.e. $\mathcal{F} \equiv \mathcal{F}' \equiv \mathcal{F}''$). Hence, we obtained a single unit clause from 4 ternary clauses after applying the NRES rule three times, which simplified substantially the original formula $\mathcal{F}$. But we can go one step further. We can see $(x_1, 1), (\bar{x}_1, 1)$ as equivalent to $(\Box \vee x_1, 1), (\Box \vee \bar{x}_1, 1)$. Hence, by NRES rule we would obtain $(\Box, 1)$ making explicit a cost of 1 for the original formula. Some authors refers to this last transformation as* Complementary Unit clause rule *(Niedermeier and Rossmanith, 2000).*

The term Neighborhood Resolution was coined by (Cha and Iwama, 1996) in the SAT context.

3.2 Extension Rule

Traditionally research work has focused on rules to converting the input formula into a simpler form by shortening clauses, reducing the number of clauses, and generating as many unit or empty clauses as possible. In this subsection, we introduce a rule called *Extension rule* (*ER*) (Larrosa and Schiex, 2003; Rollon and Larrosa, 2022; Atserias and Lauria, 2019) and relate it with the Neigborhood resolution rule. Essentially, it takes a single clause (C, w) where C is a set of literals, and then it creates two new clauses with one additional and arbitrary literal h and $\bar{h}$ respectively.

$$\{(C, w) \equiv \{(C \vee h, w), (C \vee \bar{h}, w)\}$$

See an example below to illustrate this concept.

Example 3. *Let $\mathcal{F}$ be a WCNF formula with clauses $\mathcal{F} = \{(x_1 \vee x_2, 1)\}$. The application of the extension rule to clauses in $\mathcal{F}$ produces the following* equivalent *formula $\mathcal{F}' = \{(x_1 \vee x_2 \vee h, 1), (x_1 \vee x_2 \vee \bar{h}, 1)\}$) with arbitrary literal h.*

The equivalence of the Extension Rule becomes obvious when examining the cost distributions in a truth table. Consider the example below:

Example 4. *Assume the following formula $\mathcal{F}$ and the result of applying the Extension Rule in $\mathcal{F}'$ on variable x_2. $\mathcal{F} = \{(x_1, 3)\}$ and $\mathcal{F}' = \{(x_1 \vee x_2, 3), (x_1 \vee \bar{x}_2, 3)\}$. We can see that the distribution of costs of the truth table in Table 2 is the same. Note that it also shows the equivalence for the case of applying NRES to formula $\mathcal{F}'$ which would result back into $\mathcal{F}$.*

In the context of the *Weighted Constraint Satisfaction Problem*, the *extension operation* (Larrosa and Schiex, 2003) is applied to transform the problem and is deeply related to the Extension Rule (ER). For further details, refer to the related work section. Next, we present a formal proof of the Extension Rule's correctness.

Table 2: Truth table for $\mathcal{F} = \{(x_1, 3)\}$ and $\mathcal{F}' = \{(x_1 \vee x_2, 3), (x_1 \vee \bar{x}_2, 3)\}$.

x_1	x_2	$\{(x_1,3)\}$	$\{(x_1 \vee x_2, 3), (x_1 \vee \bar{x}_2, 3)\}$
f	f	3	3
f	t	3	3
t	f	0	0
t	t	0	0

Proof. Similar to (Rollon and Larrosa, 2022), let be the resulting clauses from Extension rule $(C \vee h, w), (C \vee \bar{h}, w)$, no matter which value is assigned variable h the resulting clause would be (C, w). If h is assigned t then clause $(C \vee h, w)$ is satisfied, and literal $\bar{h}$ in $(C \vee \bar{h}, w)$ is unsatisfied and as such, can be removed which results in (C, w). Similarly, If h is assigned f then clause $(C \vee \bar{h}, w)$ is satisfied, and literal h in $(C \vee h, w)$ is unsatisfied and as such, can be removed which results in (C, w). Alternatively, assume any literal in C is satisfied, then both $(C \vee h, w)$ and $(C \vee \bar{h}, w)$ are satisfied by the current assignment. Contrary, assume all literals in C are falsified by the current assignment, then we are left with (h, w) and $(\bar{h}, w)$. No matter which value is assigned to h as we will have to pay cost w in either case. □

Since Neighborhood Resolution is based on the Resolution Rule for Max-SAT which has been proved to be *sound* and *complete* (Larrosa et al., 2008), we can do a simpler proof based on Neighborhood Resolution:

Proof. By applying Neighborhood Resolution (NRES) to the extended clauses $(C \vee h, w), (C \vee \bar{h}, w)$, we would obtain (C, w). Hence, we obtain the original formula. □

The Neighborhood rule can be considered the *inverse transformation* of the extension rule, and vice versa.

3.3 Extension Preprocessor

Given the original Max-SAT formula see the pseudocode for the *Extension Preprocessor* in Algorithm 1.

As it can be observed, it essentially applies the Extension Rule to all clauses of the original formula. The additional literal added to each pair of extended clauses can be selected in many ways. Refer to the empirical section for the selected approach.

4 RESULTS

For our experiments, we employed the original (**OR**) Max-SAT problem alongside the transformed instance using the Extension Preprocessor (**Ext**). The preprocessor was implemented using .NET Core 7.0, and its execution time is negligible for the benchmarks considered. Hence, for clarity of presentation, we have not included them.

Algorithm 1: Extension Preprocessor.

```
Input: A Max-SAT formula $\mathcal{F}$
Output: A Max-SAT formula $\mathcal{F}'$ which is
        equivalent to $\mathcal{F}$
Function ExtensionPreprocessor($\mathcal{F}$):
    $\mathcal{F}' \leftarrow \emptyset$ ;    // Initialize new formula
    foreach clause $(C, w)$ in $\mathcal{F}$ do
        $\mathcal{F}' \leftarrow \mathcal{F}' \cup \{(C \vee l, w), (C \vee \neg l, w)\}$ ;
        // Being $l$ an arbitrary literal
    end
    return $\mathcal{F}'$
```

We investigated the impact of the preprocessor on relevant Stochastic Local Search (SLS) algorithms in the literature. The following algorithms have been evaluated which are publicly available in UBCSAT solver (Tompkins and Hoos, 2004): SAMD (Hansen and Jaumard, 1990), GSAT (Selman et al., 1992), Walksat (Selman et al., 1994), Novelty (McAllester et al., 1997), Tabu Walksat (McAllester et al., 1997), Novelty+ (Hoos, 1999), Adaptnovelty+ (Hoos, 2002), IROTS (Smyth et al., 2003), VW2 (Prestwich, 2005), and adaptg2wsat+p (Li et al., 2007b). Note that all those algorithms in UBCSAT solver can handle both unweighted and weighted Max-SAT, except adaptg2wsat+p and VW2.

The benchmarks were selected from recent Max-SAT Evaluations [1] and comprise the following:

- **M2S.** Random Max-2-SAT instances with 120 variables and 1200 to 2600 binary clauses.
- **MC7** and **MC8.** Crafted Max-CUT instances on bipartite graphs with 140 nodes and 630 edges, represented as Max-2-SAT instances.
- **MCAH.** Crafted Max-CUT instances with 140 variables and 1200 to 2600 binary clauses.
- **M3SAH.** Random Max-3-SAT instances with 110 variables and 700 to 1100 ternary clauses.
- **M3SH.** Random Max-3-SAT instances instances with 250/300 variables and 1000/1200 ternary clauses.
- **WM2S.** Random Weighted Max-2-SAT with 140 variables and 1200 to 1600 binary clauses.
- **WM3SH.** Random Weighted Max-3-SAT instances instances with 70 variables and 700 to 1000 ternary clauses.

All experiments were conducted on a PC running Windows 11 Home, equipped with an AMD Ryzen 5

[1] https://maxsat-evaluations.github.io/

4500U processor at 2.38 GHz and 8 GB of RAM. The experiments involved running each Stochastic Local Search algorithm for each instance 10 times, 10000 iterations for each run, and recording the average of the returned solutions and average execution time.

We conducted preliminary experiments to select the arbitrary h literal for each pair of extended clauses $(C \vee h, w), (C \vee \bar{h}, w)$. We tested three simple methods: always assigning the literal that appears in the most clauses, the literal that appears in the fewest clauses, and a random literal. We realized that assigning the same literal significantly impacted execution performance, as it often required traversing all clauses within each SLS solver iteration. Therefore, for the remaining experiments in this section, we assigned a random literal for each pair of extended clauses.

Find a summary of the comparison in tables 3 to 12. The values in column *Diff*, show the variance between the best solution obtained for the original problem and the one achieved for the preprocessed problem. Hence, negative values indicate a deterioration in the solution resulting from the transformation. Column *OR* shows the average execution time for the original formula, and *Ext* the average time for the preprocessed formula.

The different local search algorithms exhibit distinct behavior to the transformed instances produced by the preprocessor. Specifically, Novelty, Novelty+, AdaptNovelty+, and Tabu WalkSat exhibit notable improvements with the Extension Preprocessor *Ext*. These enhancements are observed in both unweighted and weighted Max-SAT instances. However, more substantial improvements are reported for Max-2-SAT instances compared to Max-3-SAT instances.

Conversely, the *Ext* preprocessor degrades notably the performance of Walksat, GSAT and SAMD. For IROTS, VW2, and adaptg2wsat+p, there is some degradation, but it is less significant. Notably, VW2 shows slight improvements in two benchmarks, while IROTS is the least sensitive algorithm.

Regarding execution time, for *Ext* preprocessed instances, it generally takes 1.5 to 2 times longer compared to the *OR* original instances for most Local Search algorithms. This is expected since the number of clauses is doubled. The only exceptions are IROTS and SAMD, which are even faster with the transformed instances.

5 PREVIOUS WORK

In the context of SAT several works were proposed to add additional clauses to help SLS to find solutions (Cha and Iwama, 1996; Lorenz and Wörz, 2020).

Table 3: Solutions for Novelty.

Benchmark	Diff	OR	Ext
M2S	20.87	0.60	1.23
MC7	17.27	0.39	0.79
MC8	17.41	0.39	0.79
MCAH	23.21	0.54	1.13
M3SAH	1.21	0.41	0.68
M3SH	1.50	0.24	0.43
WM2S	90.66	0.68	1.07
WM3SH	5.09	0.64	1.02

Table 4: Solutions for Novelty+.

Benchmark	Diff	OR	Ext
M2S	21.04	0.60	1.23
MC7	16.71	0.39	0.79
MC8	17.28	0.39	0.79
MCAH	23.02	0.54	1.13
M3SAH	1.42	0.41	0.70
M3SH	1.30	0.24	0.44
WM2S	90.46	0.68	1.07
WM3SH	4.66	0.64	1.01

Table 5: Solutions for AdaptNovelty+.

Benchmark	Diff	OR	Ext
M2S	8.78	0.60	1.22
MC7	6.64	0.39	0.79
MC8	6.71	0.39	0.79
MCAH	11.29	0.55	1.14
M3SAH	0.23	0.40	0.68
M3SH	-0.02	0.23	0.41
WM2S	30.44	0.65	1.04
WM3SH	0.63	0.62	0.99

Table 6: Solutions for adaptg2wsat+p.

Benchmark	Diff	OR	Ext
M2S	-4.03	0.85	2.11
MC7	-2.75	0.42	1.07
MC8	-2.75	0.42	1.07
MCAH	-6.55	0.53	1.47
M3SAH	-0.58	0.59	1.24
M3SH	-0.34	0.36	0.69

Several inference rules based on Max-SAT resolution (i.e. the *Neighborhood Resolution* rule) were introduced in (Li et al., 2007a; Larrosa et al., 2008) and subsquent works which are specially designed to simplify the formula and usually target binary and unit clauses. Most of them are based on obtaining unit clauses from binary ones and then use those to create new empty clauses making a lower bound explicit in the formula. Such rules have been shown to be efficient in different types of algorithms, both as a preprocessor or embedded in the algorithm, in-

Table 7: Solutions for Walksat.

Benchmark	Diff	OR	Ext
M2S	-2.87	0.44	0.87
MC7	-5.91	0.29	0.55
MC8	-6.32	0.29	0.56
MCAH	-4.81	0.40	0.78
M3SAH	-2.28	0.30	0.49
M3SH	-4.65	0.18	0.32
WM2S	-29.22	0.61	0.90
WM3SH	-7.53	0.49	0.75

Table 8: Solutions for Tabu Walksat.

Benchmark	Diff	OR	Ext
M2S	3.94	0.43	0.82
MC7	3.01	0.27	0.53
MC8	3.04	0.29	0.55
MCAH	5.64	0.38	0.74
M3SAH	0.06	0.29	0.47
M3SH	0.24	0.17	0.31
WM2S	14.09	0.55	0.81
WM3SH	0.78	0.47	0.73

Table 9: Solutions for SAMD.

Benchmark	Diff	OR	Ext
M2S	-5.40	0.57	0.51
MC7	-10.88	0.39	0.36
MC8	-10.95	0.39	0.36
MCAH	-13.65	0.47	0.47
M3SAH	-4.67	0.48	0.32
M3SH	-6.19	0.50	0.40
WM2S	-36.90	0.44	0.50
WM3SH	-30.60	0.54	0.48

Table 10: Solutions for VW2.

Benchmark	Diff	OR	Ext
M2S	0.16	0.46	0.90
MC7	-3.28	0.31	0.59
MC8	-3.24	0.31	0.58
MCAH	0.66	0.41	0.81
M3SAH	-2.82	0.32	0.52
M3SH	-5.77	0.20	0.35

cluding *branch-and-bound algorithms* (Heras et al., 2008; Larrosa et al., 2008; Li et al., 2007a; Heras and Larrosa, 2008; Heras and Bañeres, 2013; Cherif et al., 2020), Local Search algorithms (Heras and Bañeres, 2010; Abramé and Habet, 2012), and algorithms based on iteratively calling to a SAT oracle (Heras and Marques-Silva, 2011; Py et al., 2022).

In the context of SLS algorithms, in (Heras and Bañeres, 2010) a number of preprocessors are introduced based on the resolution rule and tested for several local search algorithms. One is based on applying the resolution rule until saturation so that a variable can be eliminated. The second one is based on hyper-resolution rules applied to problems with a specific structure mainly with binary hard clauses and unit soft clauses. The last one is based on applying unit propagation to generate new unit clauses and also new empty clauses. The latter two exhibited notable performance improvements. In (Abramé and Habet, 2012), inference rules were embedded in a local search solver, resulting in notable improved performance.

Table 11: Solutions for GSAT.

Benchmark	Diff	OR	Ext
M2S	-5.29	0.31	0.62
MC7	-4.23	0.15	0.36
MC8	-5.00	0.15	0.35
MCAH	-4.13	0.21	0.51
M3SAH	-3.08	0.19	0.34
M3SH	-6.56	0.15	0.22
WM2S	-10.76	0.35	0.51
WM3SH	-4.50	0.28	0.47

Table 12: Solutions for IROTS.

Benchmark	Diff	OR	Ext
M2S	-0.26	0.67	0.58
MC7	-1.06	0.51	0.37
MC8	-1.02	0.51	0.37
MCAH	-1.35	0.59	0.50
M3SAH	-0.16	0.56	0.44
M3SH	-3.48	0.70	0.39
WM2S	-14.65	0.63	0.49
WM3SH	-2.79	0.69	0.56

In the context of the *Weighted Constraint Satisfaction Problem* (*WCSP*), the *extension operation* is commonly used in various forms of *Soft Arc Consistency* for WCSP, such as DAC*, FDAC* and EDAC* (Larrosa and Schiex, 2003; de Givry et al., 2005) which results in an equivalent problem with an explicit lower bound. Additionally, in (de Givry et al., 2003) it is shown how to solve Max-SAT as a WCSP problem. The extension operation is commonly applied to transform unary constraints into binary ones while maintaining the equivalence, but it can also be applied to constraints of other sizes in WCSP. Hence, the extension operation in WCSP is deeply related to the Extension Rule (ER) for Max-SAT. Let us see an example.

Example 5. *Refer to Figure 1. (Weighted) Max-SAT can be interpreted as a Weighted CSP, where all variables can take only two values, and (weighted) clauses are represented as tuples of values that incur a cost (the weight of the clause) when assigned simultaneously. Case a) illustrates a WCSP problem with* *two variables, x_1 and x_2, each having two possible values (t and f), and two binary constraints: when $x_1 = t$ and $x_2 = t$, a cost of 1 is incurred, and similarly when $x_1 = t$ and $x_2 = f$. In Max-SAT, this is equivalent to the clauses $(\bar{x}_1 \vee x_2, 1)$ and $(\bar{x}_1 \vee \bar{x}_2, 1)$. Applying Soft Arc Consistency in a) results in an equivalent problem in b), which reduces two binary constraints into a unary constraint $x_1 = t$ with cost 1, that is equivalent apply NRES and obtain clause $(\bar{x}_1, 1)$ in Max-SAT. However, applying the extension operation to b) restores the original equivalent formula c), that is equivalent to applying the Extension Rule in Max-SAT context.*

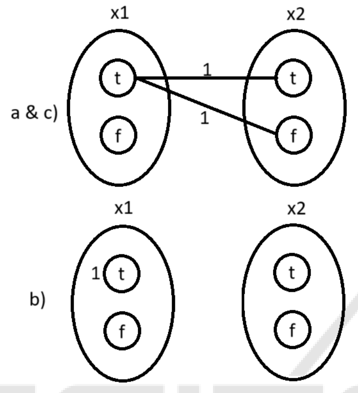

Figure 1: WCSP and Max-SAT relationship example.

In a recent work (Rollon and Larrosa, 2022), the Extension rule was explicitly reintroduced for Max-SAT and termed the *split rule* (Atserias and Lauria, 2019). In such work, the extension rule is combined with another rule that requires adding clauses with negative weight. Its theoretical potential is experimented exclusively with *Pigeon Hole Problem* (PHP) instances and it requires a handcrafted refutation based on the PHP problem structure. In our paper, we evaluated the application of the extension (i.e. split) rule in SLS solvers using a new preprocessor. Importantly, the preprocessor is not tied to any specific problem or structure. This work represents the first contribution to show that the Extension Rule can produce improved results from a practical perspective.

6 CONCLUSIONS

In this paper, we proposed a Max-SAT preprocessor based on the Extension Rule (ER) (Atserias and Lauria, 2019; Rollon and Larrosa, 2022), generating larger clauses in the resulting formula. Empirical evaluation with various SLS algorithms showed notable performance improvements, especially for Max-2-SAT instances. This finding is significant, as many problems are naturally encoded as Max-2-SAT, and some SAT problems can be reduced to Max-2-SAT (Ansótegui and Levy, 2021). Thus, achieving good results in Max-2-SAT serves as a good starting point. Previous work focused on shortening clauses and reducing their number, under the assumption that smaller formulas are better. Our study shows that larger formulas can improve outcomes for specific problems and SLS algorithms. Finally, we explored the relationships between the Neighborhood and Extension Rules, and the extension operation in WCSP.

Several potential directions of future work within this domain exist. Firstly, the development of more sophisticated heuristics for selecting the arbitrary literal employed to extend the clauses could be pursued. Additionally, incorporating a broader range of benchmarks, particularly those including hard clauses (i.e. partial Max-SAT), would be valuable to understand whether the extension rule would benefit hard clauses as well as soft ones. Furthermore, exploring the potential of hybrid approaches that combine the extension rule with other known preprocessing techniques or to consider developing heuristics to selectively apply the Extension rule, rather than applying it to all clauses as in current preprocessor. Finally, investigating the circumstances under which other algorithm families, such as branch and bound or iterative calls to SAT oracle approaches, might profit from similar strategies.

REFERENCES

Abramé, A. and Habet, D. (2012). Inference rules in local search for max-sat. In *ICTAI 2012*, pages 207–214. IEEE Computer Society.

Anbulagan, P., Pham, D. N., Slaney, J. K., and Sattar, A. (2005). Old resolution meets modern SLS. In *AAAI*, pages 354–359.

Ansótegui, C. and Levy, J. (2021). Reducing SAT to max2sat. In *IJCAI 2021*, pages 1367–1373. ijcai.org.

Atserias, A. and Lauria, M. (2019). Circular (yet sound) proofs. In *Proceedings of SAT 2019*, volume 11628 of *LNCS*, pages 1–18. Springer.

Bansal, V. and Bafna, V. (2008). Hapcut: an efficient and accurate algorithm for the haplotype assembly problem. In *Proceedings of European Conference on Computational Biology (ECCB)*, pages 153–159.

Cha, B. and Iwama, K. (1996). Adding new clauses for faster local search. In *Proceedings of the AAAI 96 and IAAI 96*, pages 332–337. AAAI Press.

Cherif, M. S., Habet, D., and Abramé, A. (2020). Understanding the power of max-sat resolution through up-resilience. *Artif. Intell.*, 289:103397.

de Givry, S., Heras, F., Zytnicki, M., and Larrosa, J. (2005). Existential arc consistency: Getting closer to full arc consistency in weighted csps. In *Proceedings of IJCAI 2005*, pages 84–89. Professional Book Center.

de Givry, S., Larrosa, J., Meseguer, P., and Schiex, T. (2003). Solving max-sat as weighted CSP. In *CP 2003*, volume 2833 of *LNCS*, pages 363–376. Springer.

Guerri, A. and Milano, M. (2003). CP-IP techniques for the bid evaluation in combinatorial auctions. In *In Proceedings of CP 2003*, LNCS, pages 863–867. Springer.

Hansen, P. and Jaumard, B. (1990). Algorithms for the maximum satisfiability problem. *Computing*, 44(4):279–303.

Heras, F. and Bañeres, D. (2010). The impact of max-sat resolution-based preprocessors on local search solvers. *J. of Satisfiability, Boolean Modeling and Computation (JSAT)*, 7(2-3):89–126.

Heras, F. and Bañeres, D. (2013). Incomplete inference for graph problems. *Optim. Lett.*, 7(4):791–805.

Heras, F. and Larrosa, J. (2008). A Max-SAT inference-based pre-processing for max-clique. In *Proceedings of SAT 2008*, pages 139–152.

Heras, F., Larrosa, J., and Oliveras, A. (2008). MiniMaxSAT: An efficient weighted Max-SAT solver. *J. Artif. Intell. Res. (JAIR)*, 31:1–32.

Heras, F. and Marques-Silva, J. (2011). Read-once resolution for unsatisfiability-based max-sat algorithms. In *In Proceedings IJCAI 2011*, pages 572–577. IJCAI/AAAI.

Hoos, H. H. (1999). On the run-time behaviour of stochastic local search algorithms for sat. In *In AAAI/IAAI 1999*, pages 661–666.

Hoos, H. H. (2002). An adaptive noise mechanism for WalkSAT. In *In AAAI/IAAI 2002*, pages 655–660.

Larrosa, J., Heras, F., and de Givry, S. (2008). A logical approach to efficient Max-SAT solving. *Artif. Intell.*, 172(2-3):204–233.

Larrosa, J. and Schiex, T. (2003). In the quest of the best form of local consistency for weighted CSP. In *Proceedings of IJCAI 2003*, pages 239–244. Morgan Kaufmann.

Li, C., Manyà, F., and Planes, J. (2007a). New inference rules for Max-SAT. *J. Artif. Intell. Res. (JAIR)*, 30:321–359.

Li, C. M., Wei, W., and Zhang, H. (2007b). Combining adaptive noise and look-ahead in local search for SAT. In *Proceedings of SAT 2007*, LNCS, pages 121–133. Springer.

Lorenz, J. and Wörz, F. (2020). On the effect of learned clauses on stochastic local search. In *In Proceedings of SAT 2020*, LNCS, pages 89–106. Springer.

McAllester, D. A., Selman, B., and Kautz, H. A. (1997). Evidence for invariants in local search. In *AAAI 97*, pages 321–326. AAAI Press / The MIT Press.

Niedermeier, R. and Rossmanith, P. (2000). New upper bounds for maximum satisfiability. *J. Algorithms*, 36(1):63–88.

Pardalos, P. M. and Rebennack, S. (2010). Computational challenges with cliques, quasi-cliques and clique partitions in graphs. In *Experimental Algorithms International Symposium (SEA)*, LNCS, pages 13–22. Springer.

Prestwich, S. D. (2005). Random walk with continuously smoothed variable weights. In *SAT 2005, St. Andrews, UK, June 19-23*, volume 3569 of *LNCS*, pages 203–215. Springer.

Py, M., Cherif, M. S., and Habet, D. (2022). Proofs and certificates for max-sat. *J. Artif. Intell. Res.*, 75:1373–1400.

Rollon, E. and Larrosa, J. (2022). Proof complexity for the maximum satisfiability problem and its use in SAT refutations. *J. Log. Comput.*, 32(7):1401–1435.

Selman, B., Kautz, H. A., and Cohen, B. (1994). Noise strategies for improving local search. In *AAAI*, pages 337–343.

Selman, B., Levesque, H. J., and Mitchell, D. G. (1992). A new method for solving hard satisfiability problems. In *AAAI-92*, pages 440–446.

Smyth, K., Hoos, H. H., and Stützle, T. (2003). Iterated robust tabu search for Max-SAT. In *Canadian Conference on AI*, pages 129–144.

Strickland, D. M., Barnes, E., and Sokol, J. S. (2005). Optimal protein structure alignment using maximum cliques. *Operations Research*, 53(3):389–402.

Tompkins, D. A. D. and Hoos, H. H. (2004). UBCSAT: an implementation and experimentation environment for SLS algorithms for SAT and MAX-SAT. In *In Proceedings of SAT 2004 Selected Papers*, volume 3542 of *LNCS*, pages 306–320. Springer.