# On variable-weighted exact satisfiability problems*

Stefan Porschen

Institut für Informatik, Universität zu Köln
Pohligstr. 1, D-50969 Köln, Germany.
Email: porschen@informatik.uni-koeln.de

**Abstract**

We show that the NP-hard optimization problems minimum and maximum weight exact satisfiability (XSAT) for a CNF formula $C$ over $n$ propositional variables equipped with arbitrary real-valued weights can be solved in $O(\|C\|2^{0.2441n})$ time. To the best of our knowledge, the algorithms presented here are the first handling weighted XSAT optimization versions in non-trivial worst case time. We also investigate the corresponding weighted counting problems, namely we show that the number of all minimum, resp. maximum, weight exact satisfiability solutions of an arbitrarily weighted formula can be determined in $O(n^2 \cdot \|C\| + 2^{0.40567n})$ time. In recent years only the unweighted counterparts of these problems have been studied [8, 9, 15].

**Key Words:** weighted exact satisfiability, exact algorithm, combinatorial optimization, counting problem, NP-completeness, perfect matching, maximum weight independent set, set partition

**AMS subject classification:** 03B05, 68Q25, 05C85

## 1 Introduction

The classical propositional satisfiability problem (SAT) is a prominent problem, namely one of the first problems that have been proven to be NP-complete [7]. Till nowadays SAT plays a fundamental role in computational complexity theory and in the theory of designing exact algorithms. SAT also has a wide range of applications because many problems can be encoded as a SAT problem via reduction. This is helpful due to the fact that meanwhile several powerful solvers for SAT have been developed (cf. e.g. [13] and references therein). Weighted satisfiability problems, by which throughout we mean that the propositional variables are the weighted objects, provide natural generalizations of SAT and also have important applications, e.g. in the area of code generation [1, 12].

Besides decision and optimization problems, counting problems are interesting and important objects of computational complexity theory. For a search - or optimization problem $S$, its counting version denoted $\#S$ searches for

*Preliminary versions of different parts of this paper appeared in [16, 17].

the number of solutions of $S$ which is less than enumerating all solutions explicitly. The present paper is devoted to study a variant of weighted SAT, namely the weighted *exact satisfiability* problem. The corresponding decision problem XSAT (also called 1-in-SAT) is known to be NP-complete [19]. More concretely, we investigate the computational complexity of optimizing and counting solutions of XSAT for weighted conjunctive normal form (CNF) formulas as input. Note that the weighted variant of a search problem can increase its computational complexity considerably. For instance 2-SAT, i.e., the satisfiability problem for an input formula containing no clauses of length at least three, can be decided in linear time, and in positive case a satisfying truth assignment can also be found in linear time [5]. Whereas, finding a minimum weight truth assignment for a variable-weighted 2-CNF formula is NP-hard. This can be seen by a straightforward reduction from the vertex cover problem to 2-SAT for monotone formulas via the *variable* graph as defined in Section 2. Thus we are motivated to consider weighted exact satisfiability problems. Unweighted XSAT gets as input a CNF formula $C$ over $n$ Boolean variables and asks whether there exists a truth assignment setting exactly one literal in each clause of $C$ to true. Recently, XSAT attracted much attention regarding the unweighted decision and counting versions [6, 8, 9, 15]. However, the first breakthrough-result by Monien et al. [14] dates back to the year 1981 and provides an algorithm deciding XSAT in $O^*(2^{0.2441n})$ time. (As usual the $O^*$-notation suppresses polynomial factors in the length of the formula.) Until 2003 this bound has been the best known, then, based on the techniques in [14], it has been improved to $O^*(2^{0.2325n})$ by Byskov et al. [6]. Also Dahllöf et al. in [9] presented an XSAT decision algorithm using a different methodology but having $O^*(2^{0.2519n})$ worst case time.

In this paper we first address the optimization problems minimum, resp. maximum, weight XSAT (MINW-XSAT resp. MAXW-XSAT). Given a CNF formula whose variables carry arbitrary real-valued weights, MINW-XSAT (resp. MAXW-XSAT) searches for an XSAT solution of minimum (resp. maximum) weight (a precise definition is given in Section 2). We show that both problems can be solved in $O^*(2^{0.2441n})$ worst case time. Our algorithm essentially uses the branching strategy provided in [14], and in addition it benefits from appropriate simplification steps preserving the optimum weight XSAT status of each intermediate weighted formula.

We also deal with the counting problems #MINW-XSAT, resp. #MAXW-XSAT, that clearly are #P-complete [20]. The unweighted counting problem #XSAT has attracted some attention during the last years. Dahlöf et al. in

[8], e.g., construct an algorithm that solves #XSAT in $O^*(2^{0.81131 \cdot n})$ time based on an $O(2^{0.40567 \cdot n})$ time algorithm for solving #MAXW-IS, i.e., for counting all maximum (positive integer) weighted independent sets in a finite graph of $n$ vertices. Their bound has been improved to $O^*(2^{0.40567 \cdot n})$ in [15], and in [9] the up to now best bound of $O^*(2^{0.2857 \cdot n})$ is shown for #XSAT. None of the algorithms solving #XSAT mentioned so far is able to enumerate all XSAT models of the input formula; each of them outputs the number of solutions only. Hence, there is no evidence whether or how these algorithms can be adapted for determining the number of all solutions of MINW-XSAT (MAXW-XSAT). However, in this paper, refining the techniques in [15], we provide algorithms solving the weighted counting problems in $O(n^2 \cdot \|C\| + 2^{0.40567 \cdot n})$ time. These algorithms use monotonization procedures reducing #MINW-XSAT, resp. #MAXW-XSAT, for arbitrary formulas to the corresponding monotone counterparts taking positive monotone formulas as input only. The latter problems are attacked by appropriate adaptations of the #MAXW-IS algorithm in [8].

As a byproduct of our results we obtain solutions to the optimization and counting versions of the weighted set partition problem (for definition cf. Section 2), which is well known to be NP-complete in its decision version [10], and has many applications in combinatorial optimization. With slight modifications, these problems can analogously be solved as the corresponding weighted XSAT problems because set partition can be identified with monotone XSAT in a dual sense.

Organisation of the paper: In Section 2 basic notions are defined and notation used throughout is provided. In Section 3 we prove some useful results helping to simplify non-monotone formulas. Section 4 is devoted to solving the optimization problems, and in Section 5 the weighted counting problems are considered. Finally, in Section 6 we finish with some concluding remarks and open problems.

## 2 Basic Notions and Notation

Let us fix the basic terminology. A *literal* is a propositional variable $x \in \{0, 1\}$ (corresponding to the Boolean truth values $\{\text{false}, \text{true}\}$) or its negation $\overline{x} := \neg x$ (negated variable). The *complement* of a literal $l$ is its negation $\overline{l}$. A *clause* $c$ is the disjunction of different literals and is represented as a literal set, thus $|c|$ is the number of literals in $c$. A CNF formula $C$ is the conjunction of different clauses and is represented as a clause set. For short we throughout use the term *formula* meaning a clause set as defined. For a

formula $C$ (resp. clause $c$), we denote by $V(C)$ (resp. $V(c)$) the set of variables contained in $C$ (resp. $c$). Similarly, given a literal $l$, $V(l)$ denotes the underlying variable. For a formula $C \in \mathrm{CNF}$ and a literal $l$, we denote by $C(l) := \{c \in C : l \in c\}$ the *subformula* of all clauses in $C$ containing $l$. For a clause (or more general a literal set) $c$, $V_+(c)$ ($V_-(c)$) denotes the set of all variables occuring unnegated (negated) in $c$; observe that $V_+(c) \cap V_-(c) \neq \varnothing$ is possible. We distinguish between the length $\|C\|$ of a formula $C$ and the number $|C|$ of its clauses.

Let CNF denote the set of all formulas, and let $\mathrm{CNF}_+$ denote the set of all *positive monotone* formulas, i.e., no clause contains a negated variable. We call $C \in \mathrm{CNF}_+$ a *matching formula* if each $x \in V(C)$ occurs at most twice in $C$. For $C \in \mathrm{CNF}_+$, a clause $c \in C$ is called a *2-3-clause* if $c$ contains a variable $x$ that occurs at least three times in $C$ and all other variables in $c$ occur at least twice in $C$. We then call $x$ a *2-3-variable* of $c$. A formula $C \in \mathrm{CNF}_+$ containing a 2-3-clause is called a *2-3-formula*. $C \in \mathrm{CNF}_+$ is called a *1-3-formula* if each clause $c \in C$ that contains a variable $x$ occuring at least three times in $C$ also contains a unique variable in $C$. Such a variable $x \in C$ is called a *1-3-variable*. Observe that an arbitrary formula $C \in \mathrm{CNF}_+$ either is a matching formula, or a 2-3-formula, or a 1-3-formula.

The exact satisfiability problem (XSAT) asks in its *decision* version, whether there exists a truth assignment $t : V(C) \to \{0, 1\}$ assigning exactly one literal in each clause of $C$ to 1, such a truth assignment is called *x-model* of $C$ (here we implicitly used the truth assignment canonically induced by $t$ on the literal set of $C$). XSAT is known to be NP-complete [19]. In the *search* version one has to decide whether $C \in \mathrm{XSAT}$, and in positive case one has to find an x-model $t$ of $C$. The empty formula $\varnothing \in \mathrm{CNF}$ also is exactly satisfiable. However, a formula $C$ containing the empty clause cannot be exactly satisfiable. An optimization variant of XSAT naturally appears when weights are assigned to the variables: Given a *weighted formula* which is a pair $(C, w)$, where $C \in \mathrm{CNF}$ and $w : V(C) \to \mathbb{R}$, solving problem MINW-XSAT (MAXW-XSAT) means to decide whether $C \in \mathrm{XSAT}$, and, in positive case, to find a *minimum* (*maximum*) x-model of $C$, i.e., a model $t$ of the smallest (largest) weight among all x-models of $C$. For $X \subseteq V(C)$, we set $w(X) := \sum_{x \in X} w(x)$. The *weight* $w(t)$ *of an x-model* $t$ is defined as $w(t) := w(t^{-1}(1)) = \sum_{x \in V(C)} w(x)t(x)$. For $C \in \mathrm{CNF}$, let $\mathrm{X}(C)$ denote the set of all x-models of $C$. Similarly, given $w : V(C) \to \mathbb{R}$, let $\mathrm{X}_{\min}(C, w) \subseteq \mathrm{X}(C)$ ($\mathrm{X}_{\max}(C, w) \subseteq \mathrm{X}(C)$) denote the set of all minimum (maximum) x-models of $(C, w)$. Note that both optimization problems are NP-hard. The counting problem #MINW-XSAT (resp. #MAXW-XSAT) is

to determine the number $|\mathrm{X}_{\min}(C,w)|$ (resp. $|\mathrm{X}_{\max}(C,w)|$), for a weighted input formula $(C,w)$.

The set partition problem (SP) has the following weighted NP-hard counterparts: MINW-SP (resp. MAXW-SP) takes as input a collection $\mathcal{M}$ of subsets of a finite set $M$, where each $T \in \mathcal{M}$ is equipped with a weight $w(T) \in \mathbb{R}$. A solution of MINW-SP (resp. MAXW-SP) is a subfamily $\mathcal{T} \subseteq \mathcal{M}$ of lowest (resp. largest) total weight such that each $m \in M$ is contained in exactly one $T \in \mathcal{T}$. In other words, a solution $\mathcal{T}$, if existing, provides a partition of $M$ of least (resp. greatest) possible weight. Let #MINW-SP (resp. #MAXW-SP) denote the corresponding #P-complete weighted counting problems.

We shall make use of two simple graph concepts assigned to formulas. First, for a monotone formula $C \in \mathrm{CNF}_+$, we define its *variable graph* $G_{V(C)}$ with vertex set $V(C)$. Two vertices are joined by an edge if there is a clause containing the corresponding variables. For a weighted monotone formula $(C,w)$, $G_{V(C)}$ is regarded as a vertex-weighted graph: each vertex carries the weight of the corresponding variable. Second, we use the *intersection graph* $G_C$ of $G$. Recall that the intersection graph of a set system (in our case the sets are the clauses) has a vertex for each set and two (distinct) vertices are joined by an edge if their sets have non-empty intersection; observe that $G_C$ has no loops.

## 3 General simplification tools

In this section we prove several results that are useful for appropriately simplifying and monotonizing weighted formulas in the algorithms described below. First, we state a basic connection relating bijections between x-model spaces to bijections between optimum x-model spaces.

**Proposition 1** *For arbitrary $\mathbb{R}$-weighted formulas $(C,w),(C',w')$, assume that there exists a bijection*

$$F : \mathrm{X}(C) \ni t \mapsto t' := F(t) \in \mathrm{X}(C')$$

*between x-model spaces such that $(*)$: $w(t) = w'(t') + \alpha$, where $\alpha \in \mathbb{R}$ is a constant independent of $t$ and $t'$. Then the restricted mapping $F_\lambda := F|\mathrm{X}_\lambda(C,w)$ is a bijection between $\mathrm{X}_\lambda(C,w)$ and $\mathrm{X}_\lambda(C',w')$, and we have $|\mathrm{X}_\lambda(C,w)| = |\mathrm{X}_\lambda(C',w')|$, for $\lambda \in \{\min,\max\}$.*

Proof. First consider the minimization case. Let $t \in \mathrm{X}_{\min}(C,w)$ and assume that $t' := F_{\min}(t) \notin \mathrm{X}_{\min}(C',w')$. Then there is a x-model $t'_0 \in \mathrm{X}(C')$

with $w'(t'_0) < w'(t')$. Let $t_0 := F^{-1}(t'_0)$ be the corresponding x-model of $C$. Applying $(*)$ twice we obtain $w(t_0) = w'(t'_0) + \alpha < w'(t') + \alpha = w(t)$, contradicting the assumption that $t$ is minimum. Hence $F_{\min}(t) \in \mathrm{X}_{\min}(C', w')$ holds for each $t \in \mathrm{X}_{\min}(C, w)$.
Conversely, let $t' \in \mathrm{X}_{\min}(C', w')$ and assume $t := F^{-1}(t') \notin \mathrm{X}_{\min}(C, w)$. Then there is an x-model $t_0 \in \mathrm{X}(C)$ with $w(t_0) < w(t)$. Let $t'_0 := F(t_0)$ be the corresponding x-model of $C'$. As above, by $(*)$, we derive $w'(t'_0) = w(t_0) - \alpha < w(t) - \alpha = w'(t')$, contradicting the assumption that $t'$ is minimum. Hence $F^{-1}(t') \in \mathrm{X}_{\min}(C, w)$ holds for each $t' \in \mathrm{X}_{\min}(C', w')$. Thus, $F^{-1}$ restricted to $\mathrm{X}_{\min}(C', w')$ equals $F^{-1}_{\min}$ from which the assertion follows.
The maximization case proceeds analogously. □

For short, a transformation modifying a weighted formula $(C, w)$ into a weighted formula $(C', w')$ is called *model-preserving* if there exists a bijection having property $(*)$ of Prop. 1 between the corresponding x-model spaces.

The first simplifying transformation step considers unit clauses in weighted formulas. The next result tells us that such clauses can be removed in a model-preserving manner from the formula, more precisely:

**Lemma 1** *Let $(C, w)$ be a weighted formula containing unit clause $c_l = \{l\} \in C$. Let $(C', w')$ be the weighted formula obtained from $(C, w)$ as follows: Set $l$ to 1, and for each $c \in C(l)$, set all literals in $c - \{l\}$ to 0. Then remove $C(l)$ and also all duplicate clauses from $C$. Next, for each $c \in C(\bar{l})$, remove $\bar{l}$ from $c$. Let $w' := w|V(C')$ be the restriction of $w$ to $V(C') = V(C) - V(C(l))$. Then $(C', w')$ is a well-defined weighted formula, i.e., there cannot occur duplicate clauses or literals. Further, $|\mathrm{X}_{\min}(C, w)| = |\mathrm{X}_{\min}(C', w')|$, specifically, we have $|\mathrm{X}_{\min}(C, w)| = 0$ if there exists $x \in V_+(C(l)) \cap V_-(C(l))$ yielding a* contradiction.

Proof. By construction, $C'$ obviously contains no duplicate clauses. Because no literals are added to a clause no clause of $C'$ can have duplicate literals.

For the remaining claims, due to Prop. 1 it is sufficient to show that the stated transformation from $(C, w)$ to $(C', w')$ is model-preserving. If $C$ is not exactly satisfiable then also $C'$ is not, so we assume the contrary. Let $t \in \mathrm{X}(C)$ be arbitrary then $t$ has to set $l$ to 1 and for each $c \in C(l)$ all literals in $c - \{l\}$ must be assigned 0. Hence $V(C(l))$ is the collection of all variables that are uniquely fixed by now and let $w_l$ be the corresponding weight, i.e., the sum of weights of exactly those variables in $V(C(l))$ set to 1. Clearly, $t' := t|V(C')$ where $V(C') = V(C) - V(C(l))$ yields a model

of $C'$. Conversely, from $t' \in \mathrm{X}(C')$ one obtains a unique model $t \in \mathrm{X}(C)$ by extending to the unique values that have to be assigned to the members in $V(C(l))$ setting all literals to 0 in $c - \{l\}$, for all $c \in C(l)$. Since $w_l$ is independent of $t, t'$, above bijection has property $(*)$ via $w(t) = w'(t') + w_l$. The last claim is obvious. □

We say that a clause $c$ contains a *complemented pair (cp) of literals* if there is a variable occuring both negated and unnegated in $c$.

**Lemma 2** *For $C \in$ CNF with weight function $w : V(C) \to \mathbb{R}$, let $c \in C$ contain exactly one complemented pair: $x, \overline{x} \in c$. Let $C_c$ be the formula obtained from $C$ by removing $c$ and assigning all literals to 0 that occur in $c' := c - \{x, \overline{x}\}$ (which can be empty) and finally removing all duplicate clauses. Let $w_c$ be the restriction of $w$ to $V(C_c) = V(C) - V(c')$. Then: the following holds true:*
*(i) $|\mathrm{X}(C)| = 2|\mathrm{X}(C_c))|$ if $x \notin V(C_c)$, and $|\mathrm{X}(C)| = |\mathrm{X}(C_c)|$ if $x \in V(C_c)$,*
*(ii) $|\mathrm{X}_\lambda(C, w)| = |\mathrm{X}_\lambda(C_c, w_c)|$, $\lambda \in \{\min, \max\}$.*

Proof. Obviously $C_c \in$ CNF is a well-defined clause set and $V(C_c) = V(C) - V(c')$ holds, because by removing duplicate clauses no other variable can be removed from the formula. For proving (i) and (ii), first assume that $x \in V(C_c)$. Then a bijection $F : \mathrm{X}(C) \to \mathrm{X}(C_c)$ obviously is given by $F(t) := t|V(C_c)$ if the reverse is defined as the extension of $t' \in \mathrm{X}(C_c)$ to $V(C)$ by assigning all literals in $c'$ to 0, which, clearly, is required for every truth assignment to be an x-model of $C$. So, we have (i) in this case. Moreover, one easily gets $w(t) = w_c(F(t)) + \alpha$, where $\alpha = \sum_{y \in V(c')} w(y)t(y) = \sum_{y \in V_-(c')} w(y)$ which is a constant since each $t \in \mathrm{X}(C)$, if existing at all, assigns all literals in $c'$ to 0. Thus (ii) follows due to Prop. 1.

If $x \notin V(C_c)$, then $x$ occurs in clause $c'$ of $C$ only. Let $\mathrm{X}_i(C)$ be the set of all x-models $t$ of $C$ such that $t(x) = i$, for $i \in \{0, 1\}$. Clearly, $\mathrm{X}(C) = \mathrm{X}_0(C) \cup \mathrm{X}_1(C)$ as disjoint union. And both $\mathrm{X}_0(C)$ and $\mathrm{X}_1(C)$ are in bijection to $\mathrm{X}(C_c)$ via restricting $t \in \mathrm{X}_i(C)$ to $V(C_c)$, as above. Hence, we have (i) in this case. Obviously, $\mathrm{X}_{\min}(C, w) \subset \mathrm{X}_i(C)$, for either $i = 0$ in case $w(x) > 0$, or for $i = 1$ in case $w(x) \leq 0$. It remains to establish relation $(*)$ in Prop. 1: For $F_i : \mathrm{X}_i(C) \to \mathrm{X}(C_c)$, and $t \in \mathrm{X}_i(C)$, we have $w(t) = w(x) \cdot i + w_c(F_i(t)) + \alpha$ with $\alpha$ as given above. Defining $\alpha' := \alpha + w(x) \cdot i$ which is a constant for fixed $i \in \{0, 1\}$ we also have proven (ii) for the minimum and maximum cases. □

A formula is called *cp-free* if none of its clauses contains a cp of literals. The following lemma shows how 2-clauses can be removed model-preserving

from a weighted formula.

**Lemma 3** *For $C \in$ CNF cp-free and $w : V(C) \to \mathbb{R}$, let $c \in C$ be a 2-clause. Then there exists a weighted formula $(C', w')$, where $w' : V(C') \to \mathbb{R}$ and $C' \in$ CNF not containing $c$ with $|C'| < |C|$ such that from each $t' \in \mathrm{X}_{\min}(C', w')$ one can uniquely determine an element $t \in \mathrm{X}_{\min}(C, w)$ and vice versa.*

Proof. For $(C, w)$ with $C \in$ CNF cp-free and $w : V(C) \to \mathbb{R}$, let $c = \{l_1, l_2\} \in C$, $V(c) = \{x_1, x_2\}$, and $w_i := w(x_i)$ where $x_i = V(l_i)$, $i = 1, 2$. As $C$ is cp-free $x_1 \neq x_2$. We distinguish two cases. Case 1: $S := V(c) \cap V(C - \{c\}) = \varnothing$. Then we set $C' := C - \{c\}$, hence $V(C') = V(C) - V(c)$, and define $w'$ as the restriction $w' := w|V(C')$. Observe that $c$ can be treated independently of the remaining formula: If $t \in \mathrm{X}_{\min}(C, w)$ then clearly the restriction of $t$ to $V(C')$ is a minimum x-model of $(C', w')$ since $t$ has to minimize $c$ locally. Reversely, if $t' \in \mathrm{X}_{\min}(C', w')$ we obtain a minimum x-model of $(C, w)$ by extension to $V(c)$ again locally minimizing $c$. Finally, to minimize $c$ locally we proceed as follows: If $c$ is monotone, either positive or negative, one variable has to be set to 1 and the other to 0. Hence, we get the minimum assignment according to setting $x_1 := 1$ and $x_2 := 0$ if $w_1 \leq w_2$, and setting $x_1 := 0$ and $x_2 := 1$ if $w_1 > w_2$. If $c$ is not monotone, either we have to set both variables to 1 or both to 0. If w.l.o.g. $c = \{x_1, \overline{x}_2\}$ we obtain a minimum assignment by setting $x_1 := 1, x_2 := 1$ if $w_1 + w_2 \leq 0$, and $x_1 := 0, x_2 := 0$ if $w_1 + w_2 > 0$. Correctness and uniqueness are obvious.

Case 2: $S \neq \varnothing$, where w.l.o.g. we assume that at least $x_1$ is in $S$ (if only $x_2$ is in $S$, then simply exchange the roles of $x_1, x_2$ in what follows). Now, we define $C'$ as obtained from $C$ as follows: First, remove $c$ from the formula, i.e., $C \leftarrow C - \{c\}$, second, for each $c' \in C(l_2)$, set $c' \leftarrow c' - \{l_2\} \cup \{\overline{l}_1\}$ and for each $c'' \in C(\overline{l}_2)$, set $c'' \leftarrow c'' - \{\overline{l}_2\} \cup \{l_1\}$. Finally, remove all duplicate clauses from the resulting formula. Obviously $C' \in$ CNF is a well-defined clause set, because all clauses are duplicate-free by construction, and $C'$ does not contain $c$, hence $|C'| \leq |C| - 1$. Since $c$ forces $l_2 := \overline{l}_1$, it is easy to see that $(**)$: $C \in$ XSAT iff $C' \in$ XSAT and moreover $V(C') = V(C) - \{x_2\}$. For defining $w'$ we distinguish two cases. We either have (i) $c = \{x_1, \overline{x}_2\}$ (resp. $c = \{\overline{x}_1, x_2\}$), or (ii) $c = \{x_1, x_2\}$ (resp. $c = \{\overline{x}_1, \overline{x}_2\}$).

In subcase (i), each x-model $t$ of $C$ exactly satisfies $c$, thus $t$ necessarily fulfills (a): $t(x_1) = t(x_2)$. We define $w'$ as $w'(x_1) := w(x_1) + w(x_2)$, and $w'(x) = w(x)$, for each $x \in V(C') - \{x_1\}$. Define the map $F : \mathrm{X}(C) \to \mathrm{X}(C')$ as follows: For each $t \in \mathrm{X}(C)$, let $t' := F(t) := t|V(C')$, and for each $t' \in \mathrm{X}(C')$, we set $F^{-1}(t') := t$ uniquely defined by the extension to $V(C)$

by setting $t(x_2) := t'(x_1) = t(x_1)$. Clearly, $F$ is well defined because of $(**)$ and is a bijection. Indeed suppose there are $t, t_1 \in \mathrm{X}(C)$, $t_1 \neq t_2$ such that $F(t_1) = F(t)$. Then $t, t_1$ can be different only at $x_2$ because all other values are the same, especially $t(x_1) = t_1(x_1)$, and due to (a) we obtain $t(x_2) = t(x_1) = t_1(x_1) = t_1(x_2)$. The converse direction is obvious. Now we prove that relation $(*)$ of Prop. 1 holds true for $F$, from which the proof for subcase (i) follows immediately. So, let $t \in \mathrm{X}(C)$, then as $t(x_1) = t(x_2)$ we have (due to $x_1 \in V(C')$ by assumption, and $t'(x_1) = t(x_1)$):

$$\begin{aligned} w(t) &= [w(x_1) + w(x_2)]t(x_1) + \sum_{x \in V(C) - \{x_1, x_2\}} w(x)t(x) \\ &= w'(x_1)t'(x_1) + \sum_{x \in V(C') - \{x_1\}} w'(x)t'(x) \quad = \quad w'(t') \end{aligned}$$

In subcase (ii) we analogously have that each x-model $t$ of $C$ exactly satisfies $c$, hence it necessarily fulfills (b): $t(x_1) = 1 - t(x_2)$. We define $w'$ as $w'(x_1) := w(x_1) - w(x_2)$, and $w'(x) = w(x)$, for each $x \in V(C') - \{x_1\}$. Further, define $F : \mathrm{X}(C) \to \mathrm{X}(C')$ again via $t \mapsto F(t) =: t' := t|V(C')$, and given $t' \in \mathrm{X}(C')$, we set $F^{-1}(t') := t$ defined as the extension of $t'$ to $V(C)$ uniquely determined by setting $t(x_2) := 1 - t'(x_1) = 1 - t(x_1)$. Clearly, $F$ is well defined and is a bijection. Indeed suppose there are $t, t_1 \in \mathrm{X}(C)$, $t_1 \neq t_2$ such that $F(t_1) = F(t)$. Then $t, t_1$ can be different only at $x_2$ because all other values are the same, especially $t(x_1) = t_1(x_1)$, and because of (b) we obtain $t(x_2) = 1 - t(x_1) = 1 - t_1(x_1) = t_1(x_2)$. The converse direction is obvious. Finally, relation $(*)$ in Prop. 1 remains to be verified. So, let $t \in \mathrm{X}(C)$, then since $t(x_2) = 1 - t(x_1)$ and $t(x_1) = t'(x_1)$ we obtain

$$\begin{aligned} w(t) &= w(x_2) + [w(x_1) - w(x_2)]t(x_1) + \sum_{x \in V(C) - \{x_1, x_2\}} w(x)t(x) \\ &= w(x_2) + w'(x_1)t'(x_1) + \sum_{x \in V(C') - \{x_1\}} w'(x)t'(x) \\ &= w(x_2) + w'(t') \end{aligned}$$

where $w(x_2)$ is a constant independent of $t, t'$. In summary we have proven the Lemma using Prop. 1. □

Next we consider model-preserving elimination of pure literals in weighted formulas:

**Lemma 4** *For $C \in$ CNF cp-free with weight function $w : V(C) \to \mathbb{R}$, let $x \in V(C)$ be a variable only occuring negated in $C$. Let $C_x$ be obtained from*

*$C$ by replacing each occurence of $\overline{x}$ by $x$ and let $w_x : V(C) \to \mathbb{R}$ be defined as $w$ except for $w_x(x) := -w(x)$. Then:*
*(i)* $|\mathrm{X}(C)| = |\mathrm{X}(C_x)|$,
*(ii)* $|\mathrm{X}_\lambda(C, w)| = |\mathrm{X}_\lambda(C_x, w_x)|$, $\lambda \in \{\min, \max\}$.

Proof. (i) follows, since we have $V(C) = V(C_x)$ and obviously every $t \in \mathrm{X}(C)$ yields a $t' \in \mathrm{X}(C_x)$ defined as $t$ except for $t'(x) = 1 - t(x)$ and vice versa. This defines a mapping $F : \mathrm{X}(C) \ni t \mapsto F(t) := t' \in \mathrm{X}(C_x)$ which obviously is a bijection of x-model spaces. To prove (ii), assume that $C \in$ XSAT otherwise we are done. From $w(t) = \sum_{y \in V(C)} w(y)t(y)$, and the fact that $t$ and $t'$ as well as $w$ and $w_x$ are distinct at $x$ only, one easily obtains $w(t) = w_x(t') + w(x)$. Due to relation $(*)$ of Prop. 1 the assertion follows, for the minimum and maximum cases.. □

For obtaining a monotone weighted formula the next result is very useful which generalizes the *simple resolution concept* [14] to minimum x-model spaces of weighted formulas.

**Lemma 5** *Let $C \in$ CNF be a cp-free formula and let $w : V(C) \to \mathbb{R}$ be an arbitrary weight function. Let $c_i = \{x\} \cup u, c_j = \{\overline{x}\} \cup v \in C$ where $x \in V(C)$ and $u, v$ are literal sets (which can also be empty). Let $C_{ij}$ be the formula obtained from $C$ as follows:*
*(1) Replace every clause $c \in C(x)$ with the clause $c - \{x\} \cup v$,*
*(2) replace every clause $c \in C(\overline{x})$ with the clause $c - \{\overline{x}\} \cup u$,*
*(3) set all literals in $u \cap v$ to 0,*
*(4) remove all duplicate clauses from the current clause set.*
*Let $w_{ij} := V(C_{ij}) \to \mathbb{R}$ be the weight function defined as follows: For each $y \in V(C_{ij}) - V(u \oplus v)$, set $w_{ij}(y) := w(y)$, and moreover, only in case that $u \oplus v \neq \varnothing$, define:*
*(1') if $V_+(u \oplus v) \cap V_-(u \oplus v) = \{z\}$, then set*

$$w_{ij}(y) := \begin{cases} w(y) \quad , & \text{if } y \in V(u \oplus v) - \{z\} \\ w(z) + w(x), & \text{if } z = y \text{ and } \overline{z} \in u, z \in v \\ w(z) - w(x), & \text{if } z = y \text{ and } z \in u, \overline{z} \in v \end{cases}$$

*(2') if $V_+(u \oplus v) \cap V_-(u \oplus v) = \varnothing$, then set*

$$w_{ij}(y) := \begin{cases} w(y) \quad , & \text{if } y \in V(v - u) \\ w(y) + w(x), & \text{if } y \in V_-(u - v) \\ w(y) - w(x), & \text{if } y \in V_+(u - v) \end{cases}$$

*Then we have:*
*(i)* $V(C_{ij}) = V(C) - [\{x\} \cup V(u \cap v)]$, *and* $|C_{ij}| \leq |C| - 1$ ,
*(ii)* $|\mathrm{X}_\lambda(C, w)| = |\mathrm{X}_\lambda(C_{ij}, w_{ij})|$, $\lambda \in \{\min, \max\}$.

PROOF. Since $C$ is assumed to be cp-free and clauses are duplicate-free, neither $u$ nor $v$ can contain $x$ or $\overline{x}$. It follows that, because of (1), (2), (3), $c_i$ and $c_j$ are transformed into the same clause $u \oplus v$ (denoting the symmetric difference), hence $x$ disappears from the variable set and, because of (4), we also have $|C_{ij}| \leq |C|-1$. Hence, we obtain $V(C_{ij}) = V(C)-[\{x\} \cup V(u \cap v)]$, because no other variable can be removed during step (4). Obviously $C_{ij} \in$ CNF is a well-defined clause set, because $C_{ij}$ and also all its clauses are duplicate-free by construction. For proving (ii), first observe that also $w_{ij}$ is well defined, specifically in case $u \oplus v = \varnothing$ which holds if either $u = v$ or $u = \varnothing = v$. It is easy to verify that in both cases $C$ admits no x-model, and also that $\varnothing \in C_{ij}$, therefore $\mathrm{X}_{\min}(C, w) = \varnothing = \mathrm{X}_{\min}(C_{ij}, w_{ij})$. So, from now on we assume $u \oplus v \neq \varnothing$ and moreover that not both $u$ and $v$ are empty. Consider the map $F : \mathrm{X}(C) \ni t \mapsto F(t) := t|V(C_{ij}) \in \mathrm{X}(C_{ij})$ where $t := F^{-1}(t')$ is defined as the extension of $t'$ to $V(C)$ by setting all literals in $u \cap v$ to 0 and $t(x) = 0$, if $t'$ sets *all* literals in $v$ to 0; and $t(x) = 1$ otherwise. Both mappings $F$ and $F^{-1}$ are one-to-one and $F$, in fact, is a bijection of x-model spaces [15].

It remains to verify that $F$ as defined above satisfies relation $(*)$ of Prop. 1 w.r.t. $w$ and $w_{ij}$ implying assertion (ii), for both the minimum and maximum cases. To that end, assume $C \in$ XSAT and let $t \in \mathrm{X}(C)$, $t' := F(t)$. Because $\forall y \in V(C_{ij}) - V(u \oplus v) : w_{ij}(y) = w(y)$ and $\forall y \in V(C_{ij}) : t(y) = t'(y)$, we obtain from $w(t) = \sum_{y \in V(C)} w(y)t(y)$ that

$$(**): \quad w(t) = w(x)t(x) + \hat{w}_{ij}(t') + \alpha_1 + \sum_{y \in V(u \oplus v)} w(y)t'(y)$$

where

$$\hat{w}_{ij}(t') := \sum_{y \in V(C_{ij}) - V(u \oplus v)} w_{ij}(y)t'(y), \quad \alpha_1 := \sum_{y \in V_-(u \cap v)} w(y) \in \mathbb{R}$$

(and specifically $\alpha_1 = 0$ if $V_-(u \cap v) = \varnothing$)). Clearly, any x-model of $C$ can assign exactly one literal in $u \oplus v$ to 1, independent of the truth value of $x$. Hence, if $u \oplus v$ contains more than one complemented pair then $C$ and $C_{ij}$ cannot be exactly satisfiable. Thus, it remains to distinguish the two cases $s := |V_+(u \oplus v) \cap V_-(u \oplus v)| \in \{1, 0\}$. In case $s = 1$, let $z$ be the only variable in the intersection, then $w_{ij}$ is uniquely defined by (1'): Since $C$ is assumed to be cp-free, both $u-v$ and $v-u$ are non-empty. If (a): $z \in u$ and $\overline{z} \in v$, then the truth values of $z$ and $x$ have to be related as $t(x) = 1 - t(z) = 1 - t'(z)$ iff $C \in$ XSAT. Due to (1') and $\forall y \in V(u \oplus v) - \{z\} : w(y)t(y) = w_{ij}(y)t'(y)$

we derive from $(**)$

$$w(t) = \alpha_1 + w(x) + w_{ij}(z)t'(z) + \hat{w}_{ij}(t') + \sum_{y \in V(u \oplus v) - \{z\}} w_{ij}(y)t'(y)$$

which means $w(t) = w_{ij}(t') + \alpha$ with $\alpha := \alpha_1 + w(x) \in \mathbb{R}$, hence $(*)$. Subcase (b): $z \in v$ and $\overline{z} \in u$ is equivalent to $t(x) = t(z) = t'(z)$ and by similar calculations one obtains relation $(*)$ where now $\alpha = \alpha_1$.

In the remaining case $s = 0$, there are three subcases, namely (a): $u - v = \varnothing$ and $v - u \neq \varnothing$ which holds if either $u = \varnothing$ or $u \subset v$ and obviously forces each $t \in \mathrm{X}(C)$ to set $t(x) := 1$. Moreover, for each $y \in V(C_{ij})$: $w_{ij}(y) = w(y)$, therefore $(**)$ immediately implies $w(t) = \alpha_1 + w(x) + w_{ij}(t')$ thus relation $(*)$ of Prop. 1 holds.

In subcase (b): $u - v$ and $v - u$ both are non-empty, we set $w_{uv} := \sum_{y \in V(u \oplus v)} w(y)t'(y)$ and observe that

$$\begin{aligned} w_{uv} &= \sum_{y \in V(v-u)} w_{ij}(y)t'(y) + \sum_{y \in V_-(u-v)} w(y)t'(y) + \sum_{y \in V_+(u-v)} w(y)t'(y) \\ &= \sum_{y \in V(u \oplus v)} w_{ij}(y)t'(y) + w(x) \left[ \sum_{y \in V_+(u-v)} t'(y) - \sum_{y \in V_-(u-v)} t'(y) \right] \end{aligned}$$

thus from $(**)$ we obtain

$$\begin{aligned} (***) \quad w(t) &= \alpha_1 + w(x)t(x) + w_{ij}(t') \\ &\quad + w(x) \left[ \sum_{y \in V_+(u-v)} t'(y) - \sum_{y \in V_-(u-v)} t'(y) \right] \end{aligned}$$

Now, defining $|V_-(u - v)| =: p = const$, first assume $t(x) = 0$, then exactly one literal in $u - v$ must be set to 1 by $t$ all other literals in $u \oplus v$ are set to 0. If the literal set to 1 belongs to a variable in $V_+(u - v)$ we have $w(t) = w_{ij}(t') + \alpha_1 + w(x)[1-p]$ and if it belongs to a variable in $V_-(u-v)$ then $w(t) = w_{ij}(t') + \alpha_1 + w(x)[-(p-1)]$ holds. Finally, for $t(x) = 1$ we have that all literals in $u-v$ are set to 0, hence $w(t) = w(x) + w_{ij}(t') - w(x)p + \alpha_1$. Thus we obtain relation $(*)$ of Prop. 1 $w(t) = w_{ij}(t') + \alpha$ with $\alpha = \alpha_1 + w(x)[1-p] \in \mathbb{R}$ for each model pair $t, F(t) = t'$.

Finally, we have subcase (c): $v - u = \varnothing$ and $u - v \neq \varnothing$ which holds if either $v = \varnothing$ or $v \subset u$ forcing each $t \in \mathrm{X}(C)$ to set $t(x) := 0$, and to set exactly one literal in $u - v$ to 1. Therefore we obtain from $(***)$ as in subcase (b),

for $t(x) = 0$, relation $(*)$ as $w(t) = w_{ij}(t') + \alpha$ with $\alpha = \alpha_1 + w(x)[1-p] \in \mathbb{R}$ completing the proof of (ii). □

The next result considers weighted monotone formulas containing clauses appearing as (nearly) supersets of other clauses.

**Lemma 6** *Let $(C, w)$ with $C \in \mathrm{CNF}_+$ monotone and $w : V(C) \to \mathbb{R}$ be a weighted formula, and let $c, c' \in C$.*
*(1) For $c \subset c'$, let $C'$ be obtained from $C$ by setting all variables in $c' - c$ to 0 and then removing all duplicate clauses. Further, let $w'$ be the restriction of $w$ to $V(C') = V(C) - (c' - c)$ then $|\mathrm{X}_{\min}(C, w)| = |\mathrm{X}_{\min}(C', w')|$.*
*(2) For variable sets $u, v$, assume $c = u \cup x, c' = u \cup v$ where $x \in V(C)$ is a variable not contained in $v$. Let $C'$ be obtained from $C$ by replacing each $c \in C(x)$ with $c - \{x\} \cup v$ and then removing all duplicate clauses from the resulting formula. Moreover, let $w'$ be defined as $w$ on $V(C') - v$ and set $w'(y) := w(y) + w(x)$, for all $y \in v$, then $|\mathrm{X}_{\min}(C, w)| = |\mathrm{X}_{\min}(C', w')|$.*

Proof. First observe that $C' \in \mathrm{CNF}$ is a well-defined clause set in case of (1) and (2), because $C'$ and all its clauses are duplicate-free by construction.

Addressing (1), note that each x-model $t$ of $C$ has to set to 0 each variable in $c' - c$. Hence restricting $t$ to $V(C')$ yields a uniquely determined x-model $t'$ of $C'$ and vice versa via extension, therefore we obtain a bijection $X(C) \ni t \mapsto t' \in X(C')$ of x-model spaces. Moreover, because $C$ is monotone, we have $w(t) = \sum_{y \in c' - c} w(y)t(y) + w(t') = w(t')$ and we are done according to Prop. 1.

Addressing (2), we have $V(C') = V(C) - \{x\}$, and we obtain a bijection $F : \mathrm{X}(C) \ni t \mapsto t' \in \mathrm{X}(C')$ of x-model spaces as follows: Simply let $t' := t|V(C')$ be the restriction to $V(C')$, thus $t'$ is uniquely determined. For $t' \in \mathrm{X}(C')$ we define $t$ by extension of $t'$ to $V(C)$ as follows: If $t'$ assigns all variables in $u$ to 0 then it has to assign only one variable of $v$ to 1 in which case we set $t(x) = 1$. If $t'$ assigns all variables in $v$ to 0 then it has to assign only one variable of $u$ to 1 in which case we set $t(x) = 0$. Therefore $t$ is uniquely determined by $t'$.

Finally, we claim that relation $(*)$ of Prop. 1 holds: First observe that $u \cap v = \varnothing$ because $c' = u \cup v$ is a well defined clause. Moreover, we have

$$\begin{aligned} w(t) &= w(x)t(x) + \sum_{y \in v} w(y)t(y) + \sum_{y \in V(C')-v} w(y)t(y) \\ &= w(x)t(x) + \sum_{y \in v} w'(y)t'(y) - w(x)\sum_{y \in v} t'(y) + \sum_{y \in V(C')-v} w'(y)t'(y) \\ &= w(x)\left[t(x) - \sum_{y \in v} t'(y)\right] + \sum_{y \in v} w'(y)t'(y) + \sum_{y \in V(C')-v} w'(y)t'(y) \\ &= w(x)\left[t(x) - \sum_{y \in v} t'(y)\right] + w'(t') \\ &= w'(t') \end{aligned}$$

where for the last inequality we used that $t(x) = 0 \;\Leftrightarrow\; t'(v) = \{0\}$, and $t(x) = 1$ iff there exists exactly one variable in $v$ set to 1 iff $\sum_{y \in v} t'(y) = 1$. Hence, we derived $w(t) = w'(t')$ in either case establishing $(*)$ of Prop. 1 and completing the proof. □

For weight function $w$, let $-w$ denote the weight function obtained from $w$ by pointwise multiplying its values by $-1$.

**Lemma 7** *Let $\mathcal{A}$ be an algorithm solving* MINW-XSAT*, for arbitrary weighted input formulas $(C, w)$ with $C \in \mathrm{CNF}, w : V(C) \to \mathbb{R}$, then $\mathcal{A}$ also solves* MAXW-XSAT *for $(C, -w)$ and vice versa.*

Proof. Suppose $\mathcal{A}$ solves MINW-XSAT for arbitrary weighted formulas $(C, w)$. We claim that $\mathrm{X}_{\min}(C, w) = \mathrm{X}_{\max}(C, -w)$. From that claim the assertion follows, because given a weighted formula $(C, w)$ we perform $\mathcal{A}$ on $(C, -w)$ finding an element $t \in \mathrm{X}_{\min}(C, -w)$, if existing, therefore $t \in \mathrm{X}_{\max}(C, w)$ as required. To verify the claim let $(C, w)$ be a weighted formula. Let $t \in \mathrm{X}_{\min}(C, w)$, and assume $t \notin \mathrm{X}_{\max}(C, w')$ where $w' := -w$. Then there exists $t_0 \in \mathrm{X}(C)$ with $w'(t_0) > w'(t)$ which is equivalent to $w(t_0) < w(t)$ contradicting $t \in \mathrm{X}_{\min}(C, w)$. Therefore $\mathrm{X}_{\min}(C, w) \subseteq \mathrm{X}_{\max}(C, -w)$. Analogously, we obtain $\mathrm{X}_{\max}(C, -w) \subseteq \mathrm{X}_{\min}(C, w)$.

The vice versa assertion stating that an algorithm solving MAXW-XSAT for $(C, w)$ also solves MINW-XSAT for $(C, -w)$ also follows from the claim above restated as $\mathrm{X}_{\max}(C, w) = \mathrm{X}_{\min}(C, -w)$. □

Observe that in general $|\mathrm{X}_{\min}(C, w)| \neq |\mathrm{X}_{\max}(C, w)|$ for weighted formulas $(C, w)$. Indeed, suppose $C$ is a positive-monotone formula of strictly negative

weighted variables, and assume that $C$ has $N \geq 2$ minimum weight x-models which is not hard to ensure. Now let $x \notin V(C)$ be a new variable with strictly positive weight and let $C_x := \{c \cup \{x\} : c \in C\}$. Then $C_x$ has only one maximum x-model, namely given by setting $x$ to 1 and all other variables to 0. But setting $x$ to 0 yields formula $C$, therefore $C_x$ also has $N \geq 2$ minimum x-models. Therefore, Lemma 7 cannot be used for deriving an algorithm solving #MAXW-XSAT from one solving #MINW-XSAT for arbitrary weighted formulas $(C, w)$.

# 4 Solving the optimization problems

This section provides branching algorithms solving the optimization problems MINW-XSAT and MAXW-XSAT. We focus on the minimum case first then deriving from the result an algorithm for the maximum case too. Subsection 4.1 describes the global structure of the algorithm, followed by an explanation of the simplification steps in Subsection 4.2. In Subsection 4.3 the formulas corresponding to the leaves of the branching tree are shown to be polynomial-time solvable. Finally, in Subsection 4.4 we analyse the branching strategy and show that our algorithm works correct and runs in $O(\|C\|2^{0.2441n})$ worst case time.

## 4.1 Structure of the algorithm

The main method of the algorithm is *branching*, that means, at a given state, we take a variable $x$ of the current formula and obtain two branches by setting it to 0 resp. to 1. So, a binary search tree, the *branching tree*, is generated in a depth first manner. Its root node $R$ corresponds to the weighted input formula, and each other node corresponds to the unique weighted formula that is calculated by branching at a variable of its parent node formula via fixing it to exactly one truth value and simplifying afterwards. The leaf nodes of the branching tree correspond to weighted matching formulas. As we shall see in Section 4.3, we can compute a minimum x-model in polynomial time for a weighted matching formula. Clearly, each leaf node $L_i$ defines a unique path $P_i$ from $R$ to $L_i$ in the branching tree corresponding to a finite sequence of pairs $P_i := ((x_{i1}, \varepsilon_{i1}), \ldots, (x_{is_i}, \varepsilon_{is_i})) \in [V(C) \times \{0,1\}]^{s_i}$ meaning that starting at the root, variable $x_{i1}$ is set to fixed value $\varepsilon_{i1} \in \{0,1\}$, followed by setting $x_{i2} \leftarrow \varepsilon_{i2}$ and traversing along $P_i$ until $L_i$ is reached. As explained in Section 4.4, the number of additional variables whose values

are determined by a branching step can be estimated so that the overall running time can be derived from corresponding recurrence relations.

Suppose that each simplification step can be performed in a model-preserving manner at the current weighted formula in the sense that we can obtain a minimum weight solution for the non-simplified formula if we have a minimum x-model of the simplified one. Then, having reached a leaf node $L_i$, by (virtually) traversing the path $P_i$ back to the root thereby inverting the assignments according to variable-values of the current branching sequence and simplifications performed, we obtain at the root, restricted to the current path, a current minimum x-model replacing the up-to-now best solution if it has a higher weight value. Doing so for each leaf when expanding the tree, we obtain a global minimum x-model of the input formula. For storing the information won along traversing a root leaf path, a stack $S$ over truth value assignments seems to provide the appropriate data structure, since having found the solution of a matching formula we need the assignments in reverse order. Moreover, in order to keep space requirements polynomially bounded it is recommended to use one stack as global object that is managed accordingly by each instance of the recursive procedure, instead of copying the current stack for each new procedure instantiation which obviously required exponential space. Next we state our procedure recursively calling itself until the global solution is found, if it exists.

`Algorithm MINW-XSAT`$(C, w, sol, currsol, S)$`:`
`Input:` $C \in \mathrm{CNF}, w : V(C) \to \mathbb{R}$
`Output: Minimum x-model` $sol$ `for` $(C, w)$`, or` **nil**
**begin**
`(01)` **if** $C$ `contains the empty clause` **then return nil**
`(02)` **if** `there occurs a contradiction` **then return nil**
`(03)` `SIMPLIFY`$(C, w, currsol, S)$
`(04)` **if** $C$ `is a matching formula` **then**
`(05)` `MINPERFMATCH`$(C, w, currsol, S)$
`(06)` **if** $currsol =$ **nil then return nil**
`(07)` `COMPUTECURRSOL`$(C, w, currsol, S)$
`(08)` **if** $sol\_weight > currsol\_weight$ **then**
`(09)` $sol \leftarrow currsol$`;` $sol\_weight \leftarrow currsol\_weight$
`(10)` **end if**
`(11)` **if** $C$ `is a 2-3-Formula` **then** `BRANCHING-2-3`$(C, w, currsol, S)$
`(12)` **if** $C$ `is a 1-3-Formula` **then** `BRANCHING-1-3`$(C, w, currsol, S)$
**end**

Each internal statement in all subprocedures stated above is followed by a

recursive call of `MINW-XSAT`. In the first two lines **nil** is returned because no x-model for the current formula can exist, and the corresponding procedure instance can be stopped. By *contradiction* in line (02) we mean that during assigning truth values to variables, a single variable is required to be set to 1 and 0 at the same time.

The algorithm uses two global parameters $currsol$, $sol$ for storing the current solution, resp. the current optimal solution. Initially, i.e., before the first call of `MINW-XSAT` is performed, both parameters are assumed to be set to **nil**. Moreover the weights of the corresponding solutions, i.e., $currsol\_weight$ and $sol\_weight$ are also global parameters initially assumed to be set to $\infty$. Further, as mentioned above a global stack $S$ is used for storing the history of the path from the root to the current node in the branching tree: For a current node of the branching tree, this history consists of the assignments determined by simplifications made in each predecessor node during Procedure `SIMPLIFY` (cf. Section 4.2) and also by the variable assignments due to the branching sequence determining that path. Branching at a node in form $x \leftarrow 0$ and $x \leftarrow 1$ yields two subproblem instances by evaluating the current formula accordingly. Branching operations are executed in Procedures `BRANCHING-2-3`, line (11), and `BRANCHING-1-3` in line (12) of the algorithm as long as the current formula is no matching formula (cf. Section 4.4).

If the current formula is a matching formula, then a leaf of the tree is reached (line (05)). During Procedure `MINPERFMATCH` a minimum weight exact solution is explicitly computed , if existing (cf. Section 4.3). Afterwards, in case the current weighted matching formula has a minimum x-model $t$, in Procedure `COMPUTECURRSOL`, the operations stored in $S$ are performed on $t$ inversely and in reversed order yielding a minimum solution with respect to the current branching tree path. Otherwise the current procedure instance returns **nil** and the previous one continues, if there is any, otherwise the algorithm halts.

The next subsections are devoted to decribe all procedures in detail.

## 4.2 Simplifying transformations

Now we present the simplifying transformations performed on a current formula in Procedure `SIMPLIFY` of Algorithm `MINW-XSAT`. These transformations are invertible and preserve the minimum (maximum) weight XSAT status of the formulas in the sense that from a minimum (maximum) x-model of the image formula one can compute in polynomial time a minimum (maximum) x-model of the original formula and vice versa. This set of transformations ensures that a current formula especially satisfies the conditions

needed for `BRANCHING-2-3` as given in [14]. Without explicitly mentioning it is understood that all assignments corresponding to transformations are pushed on the global stack.
Procedure `Simplify` consists of the following Steps 1 - 8:
**Step 1**: Initially the input formula is duplicate-free. If an intermediate current formula contains a clause $c$ multiply, remove all except one occurences of $c$. (This can obviously be done independent of weights.) Finally perform Procedure `MINW-XSAT`$(C, w, sol, currsol, S)$.
**Step 2**: If $(C, w)$ contains a unit clause, then transform $(C, w)$ due to Lemma 1 and its proof. Finally perform Procedure `MINW-XSAT`$(C, w, sol, currsol, S)$.
**Step 3**: If a clause contains more than one complemented pairs, then it can never be exactly satisfiable, hence a formula containing such a clause has 0 x-models, and **nil** is returned. However, if $(C, w)$ contains exactly one complemented pair then transform $(C, w)$ according to Lemma 2. Finally perform Procedure `MINW-XSAT`$(C, w, sol, currsol, S)$.
**Step 4**: If $(C, w)$ contains a 2-clause then transform $(C, w)$ according to Lemma 3 and its proof. Finally perform `MINW-XSAT`$(C, w, sol, currsol, S)$.
**Step 5**: If $(C, w)$ contains a literal exclusively occuring negated then transform $(C, w)$ due to Lemma 4. Finally perform Procedure `MINW-XSAT`$(C, w, sol, currsol, S)$.
**Step 6**: If $(C, w)$ contains clauses $c_i = \{x\} \cup u, c_j = \{\overline{x}\} \cup v$ then return **nil** in case that $u \oplus v = \varnothing$ or $|V_+(u \oplus v) \cap V_-(u \oplus v)| > 1$. Otherwise transform $(C, w)$ according to Lemma 5. Finally perform Procedure `MINW-XSAT`$(C, w, sol, currsol, S)$.
**Step 7**: If positive monotone $(C, w)$ contains clauses $c, c'$ with $c \subset c'$ then transform $(C, w)$ according to Lemma 6 (1). Finally perform Procedure `MINW-XSAT`$(C, w, sol, currsol, S)$.
**Step 8**: If positive monotone $(C, w)$ contains clauses $c, c'$ with $c = u \cup x, c' = u \cup v$ where $x \in V(C)$ is a variable not contained in the subclause $v$, then transform $(C, w)$ according to Lemma 6 (2). Finally perform Procedure `MINW-XSAT`$(C, w, sol, currsol, S)$.

**Corollary 1** *Let $S$ be the collection of Steps 1 to 8 above, and let $(C, w)$ be a weighted formula. Then any finite sequence $f \in S^*$ whose elements are performed subsequently on $(C, w)$ yielding $(C', w')$ induces a bijection $F_f : \mathrm{X}_{\min}(C, w) \to \mathrm{X}_{\min}(C', w')$.*

Proof. Each element $f_i$ in $f = (f_1, \ldots, f_{|f|})$ yielding intermediate formula $(C_i, w_i)$, $1 \leq i \leq f_{|f|}$, where $(C_0, w_0) := (C, w)$ and $(C_{|f|}, w_{|f|}) := (C', w')$,

induces bijection $F_i : \mathrm{X}_{\min}(C_{i-1}, w_{i-1}) \to \mathrm{X}_{\min}(C_i, w_i)$ due to Lemmata 1 to 6 correspondingly. Thus $F_f := F_{|f|} \circ F_{|f|-1} \circ \cdots F_1$ is a bijection as required, via concatenating bijections $F_i$, $1 \leq i \leq f_{|f|}$. □

### 4.3 Treating matching formulas

This subsection describes how a minimum x-model is computed in polynomial time for a weighted matching formula $(C, w)$ with $C \in \mathrm{CNF}_+$ in Procedure `MINPERFMATCH` of Algorithm `MINW-XSAT`. Recall that a matching formula is monotone and contains no variable occuring more than twice in $C$. Actually, this is implemented by an appropriate reduction to the minimum weight perfect matching problem (MINW-PM) searching for a perfect matching of minimum weight in an edge weighted simple input graph $G = (V, E)$. Recall that a perfect matching is a subset $P \subseteq E$ of pairwise non-adjacent edges in $G$ such that *every* vertex of $G$ is incident to (exactly) one edge in $P$. We construct a certain graph determined by $C$, called the *matching graph* $G_M$, which is a slight modification of the intersection graph $G_C$ of $C$.

The matching graph is constructed depending on whether there are unique variables in $C$ or not:
1.) If there is no clause in $C$ containing a unique variable, then $G_M := G_C$. Label each edge of $G_C$ by $(v, w(v))$ where $v$ is a variable of smallest weight occuring in the intersection of the corresponding clauses (if variables are assumed to be enumerated this is uniquely determined).
2.) If there is $c \in C$ containing a unique variable, construct two copies of $G'_C$ where $G'_C$ is build as $G_C$ but for edge labeling only variables occuring twice have been considered. Next, join both copies of $G'_C$ by introducing an additional edge between each two vertices in either copy that contain at least one unique variable (both vertices clearly correspond to one and the same clause). Label each additional edge by $(v, w(v))$ where $v$ is a unique variable of smallest weight in that clause (if variables are assumed to be enumerated this again is uniquely determined).

**Example:** Consider formula $C = \{c_1, c_2, c_3\}$, where $c_1 = \{t, u, x, y, z\}$, $c_2 = \{r, u, v, w, x\}$, $c_3 = \{r, v, y\}$ with all weights equal to 1. The corresponding matching graph $G_M$ is shown in Figure 1, where the labels are due to the variables in the clause intersections, weights are omitted because they are irrelevant.

It is not hard to see that a minimum weight perfect matching in $G_M$ w.r.t. to the second components of the edge labels directly corresponds to a minimum

Figure 1: Matching graph $G_M$ corresponding to $C$ weights are omitted as they are all equal.

x-model of $C$, where the first components of all matching edge labels define exactly those variables that have to be assigned to 1.

Clearly, for a matching formula its matching graph can be constructed in $O(|C|^2 \cdot |V(C)|)$ time. Since each variable that is not unique, occurs at most in two clauses it can be obtained only in the intersection of two clauses. Hence, there can occur no contradiction by selecting as edge label the minimum weight variable in each intersection. Given $C \in$ CNF we have $\sum_{c \in C} |c| = \|C\| = \sum_{x \in V(C)} \omega(x)$ where $\omega(x)$ is the number of occurences of $x$ regardless whether negated or not. Clearly, for a matching formula holds $w(x) \leq 2$, for each $x$, implying $|C| \leq \|C\| \leq 2|V(C)|$. Because MINW-PM is solvable in $O(|V|^2 \cdot |E|)$ time, for $G = (V, E)$ and arbitrary real weights assigned to the edges [2], we obtain:

**Proposition 2** *For a weighted matching formula $(C, w)$, with $w : V(C) \rightarrow \mathbb{R}$, a minimum x-model can be computed in $O(|V(C)|^3)$ time.* □

In case that an x-model $t$ is found for the current matching formula as stated above, $t$ has to be transformed to yield an x-model for the input formula of Algorithm `MINW-XSAT`. This holds specifically, if the current matching formula is empty. To that end, in Procedure `COMPUTECURRSOL` simply we have to pop the global stack and apply each elementary assignment transformation inversely in the given order. Afterwards the stack has to be filled again with the same contents in the same order as before (which could be made via a second stack that is filled in reverse order during poping) until the identifier of the last call appears, then the stack is ready for the next branching step performed in the next deepest instance of the recursive call hierarchy, if there is any.

### 4.4 The branching operations

Procedure BRANCHING-2-3, enabled in Step (11) of Algorithm MINW-XSAT in case that the current formula is a 2-3-formula, is based upon the techniques provided by Monien, Speckenmeyer, and Vornberger in [14]. There has been proven that the usual XSAT decision problem for arbitrary $C \in$ CNF with $n$ variables can be solved in $O^*(2^{0.2441n})$ time, when formulas are simplified in the same manner as ensured by Procedure SIMPLIFY disregarding weight functions. Recall that iteratively branching at a variable means to search the whole space of feasible truth assignments independent of their weights. Hence, we can rely on the result by Monien et al. in [14] if and only if the current formula has the structure that is required in their branching analysis. To that end, the formula must be a 2-3-formula $C$. In the recursive algorithm proposed in [14] a leaf of the branching tree is reached when the current formula is a 1-3-formula. The decision whether such a formula is in XSAT can be made in polynomial time which then also yields the decision for the input formula, as all transformations on the path from the root are XSAT-equivalent. Unfortunately, in the weighted case 1-3-formulas cannot be treated within polynomial time, here a leaf formula must be a matching formula. So, in addition to Procedure BRANCHING-2-3 treating only 2-3-formulas, we need a branching procedure that also treats 1-3-formulas, namely Procedure BRANCHING-1-3 in Algorithm MINW-XSAT. This procedure takes a 1-3-variable $x$ occuring at least three times in the current formula and performes branching at $x$.

More precisely, given weighted formula $(C, w)$ and $x \in V(C)$, by $C[x:\varepsilon]$ we denote the formula obtained from $C$ by assigning $x \leftarrow \varepsilon \in \{0,1\}$ and fixing all subsequently determined variables according to the appropriate branching strategy, either 2-3 or 1-3, without having performed any simplification step. Let $w[x:\varepsilon]$ denote the restriction of $w$ to $V(C[x:\varepsilon])$. Assuming that the variables of $C$ are indexed a priori, the branching procedures have the following explicit form, where X$\in$ {2-3,1-3}:

Procedure BRANCHING-X$(C, w, currsol, S)$
**begin**
take the smallest X-compatible branching variable $x$ in $C$
push identifier $x:0$ on $S$
evaluate $(C_0, w_0) \leftarrow (C[x:0], w[x:0])$
push corresponding assignments on $S$
MINW-XSAT$(C_0, w_0, sol, currsol, S)$
pop $S$ until identifier $x:0$ appears, finally pop identifier $x:$

0
`push identifier` $x:1$ `on` $S$
`evaluate` $(C_1, w_1) \leftarrow C[x:1], w[x:1]$
`push corresponding assignments on` $S$
`MINW-XSAT`$(C_1, w_1, sol, currsol, S)$
`pop` $S$ `until identifier` $x:1$ `appears, finally pop identifier` $x:1$
**end**

In order to use only one global stack $S$, we have to ensure that $S$ contains the correct information when beginning a new procedure instance. To that end, identifiers must be pushed on the stack for which the assignment defining the current branch can serve: $[x:\varepsilon]$. So, when performing a new call, first the old no longer necessary information including the last identifier can be popped from the stack. Then the new one can be pushed on $S$, and the next branch can be evaluated, etc.

We claim that the worst-case recurrence relations occuring during Procedure `BRANCHING-1-3`, are already covered by those recurrence relations produced by Procedure `BRANCHING-2-3` in [14]. The latter set of critical recurrences contains, among others, the following:

$$T(n) \leq T(n-4) + T(n-5) + 1, \quad T(n) \leq T(n-6) + T(n-3) + 1$$

Here $T(n)$ denotes the number of recursive calls the underlying algorithm performs on a formula of $n$ variables until a leaf of the branching tree is reached. The two summands on each right hand side correspond to the subproblems defined by setting a variable to 0 resp. to 1, where the lower bounds for the number of variables whose values are determined in either branching step are substracted. Obviously that relation with the weekest slow-down dominates the overall running time.

So let $C \in \mathrm{CNF}_+$ be a 1-3-formula, then by Procedure `SIMPLIFY` we know that each clause has length at least three, and by definition each clause containing a variable $x$ occurring $\geq 3$ times also contains a unique variable. Clearly, the worst case is given when such a variable occurs exactly three times restricting the number of clauses on which $x$ has an impact. First assume that all clauses containing $x$ have length exactly three:

$$c_1 = x \vee e_1 \vee a_1, \quad c_2 = x \vee e_2 \vee a_2, \quad c_3 = x \vee e_3 \vee a_3$$

where $e_i$ are the pairwise distinct unique variables and $a_i$ are variables some or all of which may coincide ($i = 1, 2, 3$). The subproblem defined by $x \leftarrow 1$

forces $e_1 \leftarrow e_2 \leftarrow \cdots \leftarrow a_3 \leftarrow 0$ thus eliminating at least 5 variables. And the subproblem defined by $x \leftarrow 0$ forces $e_i \leftarrow a_i$, for $i = 1, 2, 3$, thus eliminating at least 4 variables, i.e., this case recursively leads to the recurrence $T(n) \leq T(n-4) + T(n-5) + 1$ which is covered by the set above. Now suppose there exists at least one clause containing $x$ of length four:

$$c_1 = x \vee e_1 \vee a_1 \vee b_1, \quad c_2 = x \vee e_2 \vee a_2 \vee b_2, \quad c_3 = x \vee e_3 \vee a_3 \vee b_3$$

here variables $e_i$ and $a_i$ are as above and the $b_i$ ($i = 1, 2, 3$) are additional variables some or all of which may coincide resp. at most 2 of which may equal the constant 0 (i.e. they are "empty"). Again, the subproblem defined by $x \leftarrow 1$ eliminates at least 6 variables, and the subproblem defined by $x \leftarrow 0$ eliminates at least 3 variables, which is ensured by Procedure `SIMPLIFY` (Steps 8, 9) that is performed in the next recursive call. Hence we obtain the recurrence $T(n-6) + T(n-3) + 1$ that also is covered by the set above. Clearly, all other cases perform better, hence we are justified to conclude from [14], that all the branching operations and thus Algorithm `MINW-XSAT` have a worst case time bounded by $O^*(2^{0.2441n})$, and we are ready to state the main result of this section:

**Theorem 1** *Algorithm* `MINW-XSAT` *solves minimum weight* XSAT *for* $C \in$ CNF, $w : V(C) \to \mathbb{R}$ *in* $O(\|C\|2^{0.2441n})$ *time.* □

Proof. Considering running time, first observe that for expanding the branching tree at most $2^{0.2441n}$ recursive calls of Algorithm `MINW-XSAT` are needed as argumented above. Moreover, using appropriate data structures like doubly linked lists representing clauses and variable-to-clause pointers, branching and simplification steps can each be performed in time polynomial in $n$. Moreover the polynomial factor in $\|C\|$, the length of the input formula $C$, turns out to be linear. Since stack $S$ contains information at most for one root-leaf path in the branching tree, fixing at most $n$ variables, we have $|S| \in O(n)$. Altogether, at most $O(p(n) \cdot \|C\|)$ time is needed for computations corresponding to each node of the branching tree. In summary we obtain the claimed time bound, where the polynomial in $n$, has been absorbed into the exponent which asymptotically is justified, as evaluating the recurrence relations yields an exponential factor being slightly better than 0.2441.

For proving correctness, we state the following Claim: Let $(C, w)$ be a monotone weighted X-formula, and $x \in V(C)$ be a X-variable, for X$\in \{1{-}3, 2{-}3\}$. Recall that $(C[x : \varepsilon], w[x : \varepsilon])$ denotes the monotone weighted formula resulting by X-branching at $x$ via $x \leftarrow \varepsilon$, $\varepsilon \in \{0, 1\}$, and thereby fixing all

variables determined by the rules of exact satisfiability, but assume that no simplification step has been applied. Then the following holds:
(0) If $C$ is not exactly satisfiable then neither $C[x:0]$ nor $C[x:1]$ are, and vice versa. If $C \in \mathrm{XSAT}$ then at least one of $C[x:0]$, $C[x:1]$ also is, and vice versa.
(1) If $t \in \mathrm{X}_{\min}(C,w)$ with $t(x) = \varepsilon \in \{0,1\}$ then $t_\varepsilon := t|V(C[x:\varepsilon])$ is a minimum weight x-model of $(C[x:\varepsilon], w[x:\varepsilon])$.
(2) Let $t_\varepsilon$ be a minimum x-model of $(C[x:\varepsilon], w[x:\varepsilon])$ and let $\hat{t}_\varepsilon$ denote extension of $t_\varepsilon$, $\varepsilon \in \{0,1\}$, to $V(C)$ according to the variables determined via branching. Then $\hat{t}$ with $w(\hat{t}) = \min\{w(\hat{t}_0), w(\hat{t}_1)\}$ is a minimum x-model of $(C,w)$. (Here we define $w(\hat{t}_\varepsilon) := +\infty$ if $t_\varepsilon$ does not exist.)

Proof of the Claim: First it is obvious that each x-model $t$ of $C$ yields an x-model $t_\varepsilon$ of $C[x:\varepsilon]$ via restriction if $t(x) = \varepsilon$, because they differ only in variables determined by assignment of $x \leftarrow \varepsilon$. Similarly, each x-model $t$ of $C[x:\varepsilon]$ yields via extension to $V(C)$ an x-model of $C$ when additional variables are set according to the branching step. Clearly, if $C$ is not exactly satisfiable then neither $C[x:0]$ nor $C[x:1]$ is and vice versa. Hence branching steps specifically preserve XSAT status of formulas, and we proved even more than stated in (0).

Addressing (1) we have to consider weights. So let $C$ be exactly satisfiable and let $t \in \mathrm{X}_{\min}(C,w)$ with $t(x) = \varepsilon \in \{0,1\}$, and suppose that $t_\varepsilon := t|V(C[x:\varepsilon])$ is not a minimum weight x-model of $(C[x:\varepsilon], w[x:\varepsilon])$. Then there exists an x-model $t'$ of $(C[x:\varepsilon], w[x:\varepsilon])$ such that $w[x:\varepsilon](t') < w[x:\varepsilon](t_\varepsilon)$. Then extending $t'$ to an x-model $\hat{t}'$ of $V(C)$ by setting the fixed branching variables yields a uniquely determined x-model of $C$. Since the branching variables in $t$ and $t'$ have the same values we obtain $w(\hat{t}') < w(t)$ contradicting that $t$ is minimum, so we established (1).

Now assume $C \in \mathrm{XSAT}$ and let $t_\varepsilon$ be a minimum x-model of $(C[x:\varepsilon], w[x:\varepsilon])$, $\varepsilon \in \{0,1\}$, of which at least one must exist, let $\hat{t}_\varepsilon$ denote the unique extension of $t_\varepsilon$, $\varepsilon \in \{0,1\}$, to $V(C)$ due to the variable values determined via branching. Assume $\hat{t}$ with $w(\hat{t}) = \min\{w(\hat{t}_0), w(\hat{t}_1)\}$ is not a minimum x-model of $(C,w)$. Since at least one of $C[x:0], C[x:1]$ must have an x-model, $\hat{t}$ is well defined even if the other one has no x-model whose weight is then set to $+\infty$. Therefore, there exists an x-model $t'$ of $C$ with $w(t') < w(\hat{t})$. W.l.o.g. let $\hat{t}(x) = 0$. We distinguish two cases. First assume that $t'(x) = \hat{t}(x) = 0$, then $t'_0 := t'|V(C[x:0])$ is an x-model of $C[x:0]$, and because the weight of the determined variables is fixed we must have $w[x:0](t'_0) < w[x:0](t_0)$ contradicting that $t_0$ was minimum. Second, $t'(x) = 1 - \hat{t}(x) = 1$, then $t'_1 := t'|V(C[x:1])$ is an x-model of $C[x:1]$. If

$w[x:1](t_1') < w[x:1](t_1)$ then we have a contradiction to the assumption that $t_1$ is minimum. If $w[x:1](t_1') \geq w[x:1](t_1)$ then extension to $V(C)$ yields $w(t') \geq w(\hat{t}_1)$. On the other hand we have $w(t') < w(\hat{t})$, therefore $w(\hat{t}) > w(\hat{t}_1)$ contradicting $w(\hat{t}) = \min\{w(\hat{t}_0), w(\hat{t}_1)\}$ meaning $w(\hat{t}) \leq w(\hat{t}_1)$, thus (2). This finishes the proof of the Claim.

Let $(C_I, w_I)$ be the input formula of Algorithm `MINW-XSAT`. For proving its correctness we proceed in two steps. First we show completeness: if `MINW-XSAT` returns **nil** then there is no x-model of $(C_I, w_I)$ at all. Second we show output-correctness: if `MINW-XSAT` returns $t_I$ then $t_I$ is a minimum x-model of $(C_I, w_I)$.

Completeness is not hard to see: Suppose that $(C_I, w_I)$ possesses a solution $t_I$ but Algorithm `MINW-XSAT` returns **nil**. That means no weighted matching formula obtained during the algorithm has a solution.

By the way, there cannot exist an exactly satisfiable weighted matching formula derivable from $(C_I, w_I)$ for which there exists no branch in the search tree, hence which is not checked by the algorithm. A current path in the binary search tree is cut only if the current formula $C$ has no x-model. By Claim (0) above and due to Corollary 1 the XSAT status is preserved in each step of the algorithm, therefore no subsequent formula can have an x-model. Finally, the branching strategy used rests on the fact that monotone formulas are either 1-3- or 2-3- or matching formulas. And branching is performed as long as a matching formula occurs, then the algorithm returns back to the next branch. Only variables are fixed that are forced by exact satisfiability. This is independent of the order in which 1-3- or 2-3-variables are chosen for branching as dependencies remain the same. Hence, if at all, only matching formulas are ignored that are not exactly satisfiable.

Let $(C_I', w_I')$ be the monotonized formula recursively obtained from $(C_I, w_I)$ by Procedure `Simplify` until no further modification occurs. Therefore $(C_I', w_I')$ is determined uniquely. Let $F_0$ be the x-model space bijection between $\mathrm{X}_{\min}(C_I, w_I)$ and $\mathrm{X}_{\min}(C_I', w_I')$ that exists due to Corollary 1, so $t_I' := F_I(t_I)$ is a minimum x-model of $(C_I', w_I')$. Now the algorithm branched at say variable $x \in V(C_1)$, and $t_I'$ determines the unique branch corresponding to $t_I'(x) \in \{0, 1\}$, and in the sequence all variables determined according to the branching strategy must be set via the appropriate transform of $t_I$ accordingly. In this way $t_I$ inductively selects a unique path from the root to exactly one leaf node corresponding to a weighted formula which must have a solution as inductively applying Claim (3) and Corollary 1 ensures. Thus we obtain a contradiction to the output of the algorithm because there exists a matching formula having a solution.

For proving output correctness, suppose that Algorithm `MINW-XSAT` outputs $t_I$. Let $\mathcal{M}$ denote the set of all matching formulas inspected by the algorithm and let $\mathcal{X}(\mathcal{M})$ be the set of corresponding x-models calculated by the algorithm, and finally let $\hat{\mathcal{X}}(\mathcal{M})$ be the set of transformed x-models of $(C_I, w_I)$ as calculated during `COMPUTECURRSOL`. Then, by construction, holds $w_I(t_I) = \min_{\hat{t} \in \hat{\mathcal{X}}(\mathcal{M})} w_I(\hat{t})$. Now suppose $t_I$ is not a minimum x-model of $(C_I, w_I)$. Then there exists $t' \in \mathrm{X}_{\min}(C_I, w_I)$ such that $t' \neq t_I$ and $w_I(t') < w_I(t_I)$. As in the completeness proof $t'$ selects a unique path let say to matching formula $(C_0, w_0) \in \mathcal{M}$, let $t'_0$ be the corresponding model obtained from $t'$ via trasforming due to the selected path. Inductively applying Claim (2) and Corollary 1 we obtain that $t'_0$ is a minimum x-model of $(C_0, w_0)$. Now we have two cases: (i) $t'_0 \in \mathcal{X}(\mathcal{M})$ and (ii) $t'_0 \notin \mathcal{X}(\mathcal{M})$. In case (i) inductively applying Corollary 1 and Claim (3) traversing to the root we obtain $w_I(t') \geq w_I(t_I)$, because of the minimum condition, which is a contradiction. In case (ii) let $t_{I_0}$ be the member of $\mathcal{X}(\mathcal{M})$ corresponding to $(C_0, w_0)$. In subcase (a), $w_0(t'_0) < w_0(t_{I_0})$, contradicting that by construction $t_{I_0}$ must be minimum. In subcase (b), $w_0(t'_0) \geq w_0(t_{I_0})$, again traversing back to the root formula inductively implying Claim (3) and Corollary 1 we obtain $w_I(t') \geq w_I(\hat{t}_{I_0}) \geq \min_{\hat{t} \in \hat{\mathcal{X}}(\mathcal{M})} w_I(\hat{t}) = w_I(t_I)$, where $\hat{t}_{I_0} \in \hat{\mathcal{X}}(\mathcal{M})$, yielding a contradiction and finishing output correctness proof. □

Due to Lemma 7, we obtain from the last result:

**Corollary 2** *Maximum weight* XSAT*, for* $C \in \mathrm{CNF}, w : V(C) \to \mathbb{R}$*, can be solved in* $O(\|C\| 2^{0.2441n})$ *time.* □

# 5 Algorithms for #MINW-XSAT and #MAXW-XSAT

Next we consider the #P-complete counting problems #MINW-XSAT and #MAXW-XSAT. To that end, we first restrict to monotone formulas in Subsection 5.1, followed by considering the general case in Subsection 5.2.

## 5.1 Solving the monotone cases

This section is devoted to provide algorithms solving #MINW-XSAT (resp. #MAXW-XSAT) restricted to the class $\mathrm{CNF}_+$, in $O(n^2 \cdot \|C\| + 2^{0.40567 \cdot n})$ time, for formulas $C$ containing $n$ weighted variables. By a dualization argument presented below it will turn out that these algorithms also solve

the set partition counting variants #MINW-SP (resp. #MAXW-SP). In [8], Dahlöf and Jonsson proved an upper bound of $O(2^{0.40567 \cdot n})$ for calculating the number of all maximum weight independent sets in a graph of $n$ vertices each equipped with a positive integer weight. For convenience, we refer to that algorithm as to the DJ-Algorithm. Recall that the maximum weight independent set problem (MAXW-IS) gets as input a finite (simple) graph $G = (V, E)$, and a vertex weight function $w : V \to \mathbb{N}$. It asks whether there is an independent set in $G$, i.e., a set of pairwise non-adjacent vertices, of maximal weight. Observe that MAXW-IS is NP-hard even if all weights are equal to 1 which follows from the vertex cover problem [10], because a minimum cardinality vertex cover in $G$ is the complement of a maximum cardinality independent set.

In order to attack monotone #MINW-XSAT we reduce it to a conditional variant of #MAXW-IS called #MINW-MAXW-IS. The underlying optimization problem MINW-MAXW-IS is defined as follows, for any weight functions $f_1 : V \to \mathbb{N}, f_2 : V \to \mathbb{R}$:

**Input**: $G = (V, E)$, $f : V \to \mathbb{N} \times \mathbb{R}$ with $f(x) =: (f_1(x), f_2(x)), x \in V$.
**Output**: $X \subseteq V$ such that $f_2(X) = \min\{f_2(Y) : Y \in F_1^{\max}(G)\}$, where $F_1^{\max}(G)$ is the set of all independent sets $Y$ in $G$ such that $f_1(Y)$ is maximal, with $f_i(S) := \sum_{x \in S} f_i(x)$, $i = 1, 2$, for any $S \subseteq V(G)$.

Thus, an algorithm solving #MINW-MAXW-IS, has to count all independent sets $X$ in $G$ of minimal weight w.r.t. the second component under the condition that $f_1(X)$ is maximal.

**Proposition 3** *Counting all solutions of* MINW-MAXW-IS *is possible in* $O(2^{0.40567 \cdot |V(G)|})$ *time, for input* $G, f = (f_1, f_2)$, *where* $f_1 : V(G) \to \mathbb{N}$, $f_2 : V(G) \to \mathbb{R}$.

Proof. In [8] an adaptation of the DJ-algorithm is outlined for counting among all maximum weighted independent sets in a graph only those that have minimal cardinality (cf. [8], proof of Prop. 5.2). This algorithm runs in $O(2^{0.40567 \cdot |V(G)|})$ worst case time. Clearly that variant of the maximum weight independent set problem corresponds to #MINW-MAXW-IS where $f_2(x) = 1$ for each vertex $x \in V(G)$. In this adapted version, the return function of the algorithm is employed by a separate component reserved for the cardinality of the current independent set under consideration. It is obvious that the mentioned adaptation carries over also to the generalization where $f_2$ is an arbitrary real-valued function: Simply take the value $f_2(X)$ in the corresponding component of the return function. It is not hard to see that the resulting recursive algorithm works as desired with this modification. □

Monotone #MINW-XSAT can be identified as a subproblem of #MINW-MAXW-IS in the following way. For $C \in \mathrm{CNF}_+$ with variable graph $G := G_{V(C)}$, consider variable weight function $w : V(C) \to \mathbb{R}$. Recall that each variable $x \in V(C)$ constitutes a vertex in $G$ and that two vertices are joined by an edge if the corresponding variables occur together in a clause. Since we have positive literals only, $C(x) = \{c \in C | x \in c\}$ is the subformula of all clauses in $C$ containing $x$. As vector-valued weight function $f = (f_1, f_2)$ we define $f : V(C) \to \mathbb{N} \times \mathbb{R}$, by $f_1(x) := |C(x)|$, $f_2(x) := w(x)$, for each $x \in V(C)$. Now, $t$ is a minimum x-model of $C$ if and only if vertex set $t^{-1}(1)$ is a solution of MINW-MAXW-IS for $G, f$. Indeed, let $t$ be any x-model of $C$, then each $x \in t^{-1}(1)$ is the unique variable exactly satisfying subformula $C(x)$, hence the corresponding vertex contributes first component weight $f_1(x) = |C(x)|$ in $G$. Clearly, variables in $t^{-1}(1)$ must yield a partition $C = \bigcup_{x \in t^{-1}(1)} C(x)$, thus $f_1(t^{-1}(1)) = |C|$ which is maximum, because a larger weight meant that there are clauses in which more than one variable is set to 1. Conversely, it is easy to see that each independent set $X \subset V(G)$ of weight $w(X) = f_1(X) = |C|$ defines an x-model of $C$ assigning 1 to exactly those varibles corresponding to vertices in $X$, and 0 to the remaining variables. Observe that an independent set of weight larger than $|C|$ cannot exist, because otherwise there are two variables occuring in the same clause and corresponding vertices are adjacent in $G$. Hence such an independent set indeed has maximal first component weight. Therefore, $t$ is a minimum x-model, if and only if it satisfies $w(t) = f_2(t^{-1}(1)) = \min\{f_2(\hat{t}^{-1}(1)) : \hat{t} \in \mathrm{X}(C)\}$ which is equivalent to the fact that $t^{-1}(1)$ provides a solution of MINW-MAXW-IS for input instance $G, f = (f_1, f_2)$ as defined above.

**Theorem 2** #MINW-XSAT *for positive monotone formulas $C$ of $n$ variables (resp.* #MINW-SP *for a collection $C$ of $n$ input sets), where variables (resp. input sets) are equipped with arbitrary real weights, can be solved in $O(n^2 \cdot \|C\| + 2^{0.40567 \cdot n})$ time, with $\|C\| = \sum_{c \in C} |c|$.*

Proof. First, we show that the reduction provided above from #MINW-XSAT to #MINW-MAXW-IS can be executed in $O(n^2 \cdot \|C\|)$ time, where $n := |V(C)|$ and $C \in \mathrm{CNF}_+$. This confirmes the claim of the theorem regarding #MINW-XSAT relying on Proposition 3. So, for computing the weighted variable graph $(G_{V(C)}, f)$, we first have to determine the vertex weights, for which an array $W$ is maintained. Each position of $W$ stores a variable occuring in $C$ together with both weight components. Regarding the first component $f_1$, we have to determine the number of occurences $|C(x)|$ of each variable $x$ in $C$. This can be done by running once through the formula.

Each variable $x$ found in $C$ is compared to all variables already stored in $W$, for each of which we maintain a counter corresponding to the number of its occurences in $C$. If we find a match the counter for $x$ is incremented by 1, otherwise the variable is stored in the next position of $W$, and its counter is initialized by value one, finally the second weight component $w(x)$ is assigned. Therefore a running time proportional to $n^2 \cdot \|C\|$ results. Next we have to form the edges of the variable graph. Clearly, this can be done by building a clique $K_{|c|}$ for each clause $c \in C$. As there are $|C| \leq \|C\|$ clauses and each clause contains at most $n^2$ variables, the time needed for constructing all edges of $G_C$ is upper bounded by $O(n^2 \cdot \|C\|)$.

It remains to verify the claim of the theorem regarding #MINW-SP. First observe that MINW-SP and MINW-XSAT for monotone formulas essentially are the same, as the following dualization argument shows: Let $(M, \mathcal{M}, w)$ be an input instance of MINW-SP with weight function $w : \mathcal{M} \to \mathbb{R}$. Assigning to each $T \in \mathcal{M}$ a Boolean variable $x_T \in \{0, 1\}$ equipped with weight $w(T)$, and assigning to each $m \in M$ a clause $c_m$ that contains variable $x_T$ if and only if $m \in T$ yields a variable-weighted positive monotone input formula of MINW-XSAT. It is easy to see that solving MINW-XSAT for this formula is the same as solving MINW-SP for $(M, \mathcal{M}, w)$, because exactly the variables set to 1 correspond to those sets that have to be chosen to obtain a minimum weight partition of $M$. The converse direction reducing monotone MINW-XSAT to MINW-SP proceeds analogously. For completing the proof it is left to verify that the reduction previously described can be done in at most $O(n^2 \cdot \|C\|)$ time. To that end, we hold a table $A$ of Boolean having size $|M| \cdot \|\mathcal{M}\|$ storing $A(m_i, T_j) = 1$ iff $m_i \in T_j$, where $\mathcal{M} := \{T_1, \ldots, T_{|\mathcal{M}|}\}$ and $M = \{m_1, \ldots, m_{|M|}\}$ are assumed to be indexed. After having filled this table, we row-wise assign to each clause $c_m$ all $x_{T_j}$ with $A(m_i, T_j) = 1$ needing $O(|M| \cdot |\mathcal{M}|)$ time. For filling the table, we run once through $\mathcal{M}$ starting with $T_1$ working column-wise. In column $i$, we assign value 1 to entry $A(m_j, T_i)$ if we find $m_j \in T_i$. So, filling the table needs $O(\|\mathcal{M}\|)$ time overall. Therefore, in summary, we obtain $O(\|\mathcal{M}\| + |M| \cdot |\mathcal{M}|) = O(\|C\| + |C| \cdot n) = O(n \cdot \|C\|)$, where we took into account that $M$ is in bijection to $C$, $\mathcal{M}$ is in bijection to $V(C)$, and $\|C\| = \|\mathcal{M}\| > |C| = |\mathcal{M}|$. □

Observe that monotone MINW-XSAT more directly can be identified with the *minimum weight exact hitting set problem*. Corresponding input instances consist of a base set $S$ of arbitrarily weighted elements (which are the variables of a formula $C \in \mathrm{CNF}_+$) and a collection $\mathcal{T}$ of subsets of $S$ (corresponding to the clauses in $C$). Then one searches for a minimum weight subset $X \subseteq S$ such that $X$ contains exactly one element of each $T \in \mathcal{T}$.

Clearly a minimum x-model $t$ of $C$ via $t^{-1}(1) \subseteq V(C)$ yields a minimum weight hitting set and vice versa, correspondingly. We thus obtain:

**Corollary 3** *Counting all solutions of minimum weight hitting set takes $O(n^2 \cdot \|C\| + 2^{0.40567 \cdot n})$ time, for a base set of $n$ arbitrarily weighted elements and subset collection $C$.* □

Similarly, for #MAXW-XSAT, we consider #MAXW-MAXW-IS which is defined as follows via its underlying optimization problem MAXW-MAXW-IS, for arbitrary weight functions $f_1 : V \to \mathbb{N}, f_2 : V \to \mathbb{R}$:

**Input**: $G = (V, E)$, $f : V \to \mathbb{N} \times \mathbb{R}$ with $f(x) =: (f_1(x), f_2(x)), x \in V$.
**Output**: $X \subseteq V$ such that $f_2(X) = \max\{f_2(Y) : Y \in F_1^{\max}(G)\}$, where $F_1^{\max}(G)$ is the set of all independent sets $Y$ in $G$ such that $f_1(Y)$ is maximal, with $f_i(S) := \sum_{x \in S} f_i(x)$, $i = 1, 2$, for any $S \subseteq V(G)$.

Thus, an algorithm solving #MAXW-MAXW-IS, has to count all independent sets $X$ in $G$ of maximal weight w.r.t. the second component under the condition that $f_1(X)$ is maximal. Now mimicking the argumentation above immediately yields:

**Corollary 4** #MAXW-XSAT *for positive monotone formulas $C$ of $n$ variables (resp.* #MAXW-SP *for a collection $C$ of $n$ input sets $c$), where variables (resp. input sets) are equipped with arbitrary real weights, can be solved in $O(n^2 \cdot \|C\| + 2^{0.40567 \cdot n})$ time, with $\|C\| = \sum_{c \in C} |c|$.*
*Moreover, counting all solutions of maximum weight hitting set takes $O(n^2 \cdot \|C\| + 2^{0.40567 \cdot n})$ time, for a base set of $n$ arbitrarily weighted elements and subset collection $C$.* □

## 5.2 The general non-monotone case

In this section we provide a polynomial time reduction from #MINW-XSAT (#MAXW-XSAT) for arbitrary CNF formulas to #MINW-XSAT (#MAXW-XSAT) restricted to the class $\mathrm{CNF}_+$ of monotone formulas enabling us to solve these problems in $O(n^2 \cdot \|C\| + 2^{0.40567 \cdot n})$ time, too. The main idea is to establish a sequence of polynomial time computable mappings that, iteratively, transform an arbitrary input instance $(C, w)$ into $(C', w')$ where $C'$ is positive monotone and such that the number of minimum (maximum) x-models of the original instance is preserved, i.e., equals the number of minimum (maximum) x-models of the transformed instance. Since for the empty formula $\varnothing$ holds $V(\varnothing) = \varnothing$, we have $|\mathrm{X}(\varnothing)| = 2^0 = 1$. These transformations are managed by Procedure `Monotonization` stated

below which as input gets a non-(positive-)monotone CNF formula $C$ and recursively calls itself until $C$ is positive monotone thereby it computes a multiplicator $N \in \{0,1\}$ for $C$. $N$ gets value 0 if and only if $C$ turns out not to be exactly satisfiable during the monotonization process:

**Procedure** $\texttt{Monotonization}_\lambda(C, w; N)$ $(\lambda \in \{\max, \min\})$
`Input:` $C \in \mathrm{CNF}$, $w : V(C) \to \mathbb{R}$
`Output:` $C' \in \mathrm{CNF}_+, w'$: $|\mathrm{X}_\lambda(C, w)| = |\mathrm{X}_\lambda(C', w')|$, $N \in \{0,1\}$
**begin**
`(0)` $N \leftarrow 1$
`(1)` **if** `there occurs a contradiction` **then return** $N \leftarrow 0$
`(2)` **if** $\varnothing \in C$ **then return** $N \leftarrow 0$
`(3)` **if** $\exists c \in C$ `with` $\geq 2$ `complemented pairs` **then return** $N \leftarrow 0$
`(4)` **if** $\exists c \in C$ `with 1 complemented pair` $\{x, \overline{x}\}$ **then**
$C \leftarrow C_c, w \leftarrow w_c$, $\texttt{Monotonization}_\lambda(C, w; N)$
`(5)` **if** $\exists x \in V(C)$ `occuring only negated in` $C$ **then**
$C \leftarrow C_x, w \leftarrow w_x$, $\texttt{Monotonization}_\lambda(C, w; N)$
`(6)` **if** $\exists c_i = \{x\} \cup u, c_j = \{\overline{x}\} \cup v \in C, x \in V(C)$**then**
**if** $u \oplus v = \varnothing$ **or** $|V_+(u \oplus v) \cap V_-(u \oplus v)| > 1$**then return** $N \leftarrow 0$
$C \leftarrow C_{ij}, w \leftarrow w_{ij}$, $\texttt{Monotonization}_\lambda(C, w; N)$
`(7)` **return** $C, w, N$
**end**

**Theorem 3** *For $C \in \mathrm{CNF}$, $w : V(C) \to \mathbb{R}$, $\lambda \in \{\max, \min\}$, Procedure* $\texttt{Monotonization}_\lambda$*, in $O(n^2 \cdot \|C\|)$ time, correctly computes a monotone formula $C' \in \mathrm{CNF}_+$ with $w' : V(C') \to \mathbb{R}$ such that $|\mathrm{X}_\lambda(C, w)| = |\mathrm{X}_\lambda(C', w')|$.*

Proof. Correctness of Steps (0) to (3) is obvious, where by contradiction we mean the circumstance that the same variable during a simplifying transformation is forced to be set to 1 and 0 at the same time. Correctness of Steps (4) to (6) follows by Lemmata 2, 4, 5, for $\lambda \in \{\min, \max\}$, and by the fact that the current formula is cp-free when Step (5) is executed for the first time. Thus the current weighted formula returned in Step (7) is positive monotone having the same number of minimum (maximum) x-models as the weighted input formula.

Addressing the claim for the running time we assume that we can rely on appropriate data structures, such as doubly linked lists: For each variable $x$, we maintain a list containing pointers to all clauses containing $x$, carrying additional information whether $x$ appears negative or not. Similarly, for each clause we hold a list, containing pointers to all variables contained, and assume that these lists are doubly linked. It is not hard to verify that

these data structures, for given input instance, can be filled in $O(n^2 \cdot \|C\|)$ time, and that this bound also dominates the running time of Procedure $\texttt{Monotonization}_\lambda$ relying on these data structures. □

Now we are ready for presenting the main algorithms solving the weighted XSAT counting problems, for arbitrary weighted formulas, where as abbreviation we use $(\Lambda, \lambda) \in \{(\text{MINW}, \min), (\text{MAXW}, \max)\}$:

**Algorithm** $\#\Lambda\text{-XSAT}(C, w; |\mathrm{X}_\lambda(C, w)|)$
`Input:` $C \in \text{CNF}$, $w : V(C) \to \mathbb{R}$
`Output:` $|\mathrm{X}_\lambda(C, w)|$
**begin**
`(1)` **if** $C$ `is not positive monotone` **then**
`(2)` $\texttt{Monotonization}_\lambda(C, w; N)$
`(3)` **if** $N = 0$ **then return** $|\mathrm{X}_\lambda(C, w)| \leftarrow 0$
`(4)` **if** $C = \varnothing$ **then return** $|\mathrm{X}_{\min}(C, w)| \leftarrow 1$
`(5) solve monotone` $\#\Lambda$`-XSAT (*` `let` $r$ `be its result` `*)`
`(6)` **return** $|\mathrm{X}_\lambda(C, w)| \leftarrow r$
**end**

**Theorem 4** *Algorithm* #MINW-XSAT *(*#MAXW-XSAT*) correctly calculates the number of all minimum (maximum) weight x-models of arbitrary weighted input formula* $C \in \text{CNF}$, $w : V(C) \to \mathbb{R}$, *in* $O(n^2 \cdot \|C\| + 2^{0.40567 \cdot |V(C)|})$ *time.*

Proof. First consider Algorithm #MINW-XSAT. Theorem 3 establishes the correctness of statement (2) which needs to be executed only if the input formula is not monotone. By the correctness of Prodecure $\texttt{Monotonization}_{\min}$ it is guaranteed that the multiplicator $N$ is 0 if and only if $C \notin \text{XSAT}$ in which case the number of x-models is 0, hence (3) is correct. Correctness of (4) is due to the fact mentioned above that the empty formula has only one x-model. Step (5) is correct according to Theorem 2, based on the weighted positive monotone formula as output by Procedure $\texttt{Monotonization}_{\min}$.

Addressing the running time, observe that the test in (1) needs $O(\|C\|)$ time, and Procedure $\texttt{Monotonization}_\lambda$ can be executed $O(n^2 \cdot \|C\|)$ time due to Theorem 3. Finally, Step (5) solving monotone #MINW-XSAT performs in $O(n^2 \cdot \|C\| + 2^{0.40567 \cdot |V(C)|})$ time according to Theorem 2 establishing the assertion for Algorithm #MINW-XSAT.

Argumentation for Algorithm #MAXW-SAT proceeds analogously based on Theorem 3 regarding Procedure $\texttt{Monotonization}_{\max}$ and Corollary 4 completing the proof. □

# 6 Concluding remarks and open problems

We proposed an algorithm solving the NP-hard optimization problems minimum, resp. maximum, weight XSAT in $O(\|C\|2^{0.2441n})$ time, for arbitrarily weighted $C \in$ CNF over $n$ variables. As far as we know the presented algorithm is the first breakthrough in handling the weighted XSAT optimization version in nearly the record worst case time of the decision problem. To construct a faster optimization algorithm, especially one that achieves the up to now best running time for the decision problem, namely $O^*(2^{0.2325n})$ in [6] is left as an open problem.

Recall from the proof of Theorem 2 that the set partition problem (SP) is, by dualization, can be identified with XSAT restricted to positive monotone formulas. So, we also provided algorithms for solving minimum (maximum) weight SP in $O(\|C\|2^{0.2441n})$ time, where $n$ is the number of sets in the input family, which are equipped with arbitrary real weights.

Unfortunately, the presented optimization algorithms cannot be used also to solve the weighted counting problems #MINW-XSAT resp. #MAXW-XSAT. This is due to the fact that the leaf case of the branching tree algorithm would require to compute all minimum, resp. maximum, weight perfect matchings in polynomial time which is not expected to be achievable, since it meant to solve #P-complete problems in polynomial time. However, relying on the monotonization techniques in Section 3, we were able to provide algorithms for #MINW-XSAT, resp. #MAXW-XSAT, running in $O(n^2 \cdot \|C\| + 2^{0.40567 \cdot n})$ time, for input formulas $C \in$ CNF of $n$ real weighted variables. Observe that testing all possible truth assignments in a brute-force manner needs $O(n^2 \cdot \|C\| \cdot 2^n)$ time.

Whether faster algorithms solving unweighted #XSAT like that in [9] can be adapted also to treat the weighted case without affecting the running time, is an open question. Another open problem is whether all (weighted) x-models can be enumerated explicitly with polynomial delay only, which is not provided by an algorithm merely counting (weighted) x-models. Such an enumeration algorithm running with polynomial delay only, in the number of solutions, has been provided, e.g., by Johnson et al. [11] for enumerating all maximal independent sets in a finite graph.

Of specific interest are also the weighted optimization and counting versions of XSAT restricted to 3-CNF formulas, X3SAT for short, as X3SAT remains NP-complete. The NP-hard optimization variants MINW-X3SAT, resp. MAXW-X3SAT, can be solved in $O(2^{0.16254n})$ time [3, 4] via techniques provided here and in [18]. However, it is left for future work to achieve for

the optimization variants of X3SAT the up to now best bound of $O(2^{0.1379n})$ for decision as obtained in [6]. Finally, solving the weighted counting problems #MINW-X3SAT, resp. #MAXW-X3SAT, faster than the general cases of unrestricted CNF formulas are also left as open problems. For the unweighted counterpart #X3SAT, an algorithm having worst case running time bounded by $O(2^{0.20001n})$ is presented in [9].

## References


[1] A. V. Aho, M. Ganapathi, and S. W. Tjiang, Code Generation Using Tree Matching and Dynamic Programming, ACM Trans. Programming Languages and Systems, 11 (1989) 491-516.

[2] D. Applegate, and W. Cook, Solving large-scale matching problems, in: D. S. Johnson, C. C. McGeoch (Eds.), Algorithms for Network Flows and Matching Theory, American Mathematical Society, pp. 557-576, 1993.

[3] G. Arnopolina, Über Variablen-Gewichtete X3SAT Optimierungs-Probleme, Diploma Thesis, Univ. Köln, 2006.

[4] G. Arnopolina, and S. Porschen, Solving optimum weight Exact 3-Satisfiability in $O(2^{0.16254n})$ time, Techn. Report, Univ. Köln, 2006, in preparation.

[5] B. Aspvall, M. R. Plass, and R. E. Tarjan, A linear-time algorithm for testing the truth of certain quantified Boolean formulas, Inform. Process. Lett. 8 (1979) 121-123.

[6] J. M. Byskov, B. Ammitzboll Madsen, and B. Skjernaa, New Algorithms for Exact Satisfiability, Theoretical Comp. Science 332 (2005) 515-541.

[7] S. A. Cook, The Complexity of Theorem Proving Procedures, in: Proceedings of the 3rd ACM Symposium on Theory of Computing, pp. 151-158, 1971.

[8] V. Dahllöf, and P. Jonsson, An Algorithm for Counting Maximum Weighted Independent Sets and its Applications, in: Proceedings of the 13th ACM-SIAM Symposium on Discrete Algorithms, pp. 292-298, 2002.

[9] V. Dahllöf, P. Jonsson, and R. Beigel, Algorithms for four variants of the exact satisfiability problem, Theoretical Comp. Sci. 320 (2004) 373-394.

[10] M. R. Garey and D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman and Company, San Francisco, 1979.

[11] D. S. Johnson, M. Yannakakis, and C. H. Papadimitriou, On Generating All Maximal Independent Sets, Inform. Process. Lett. 27 (1988) 119-123.

[12] S. Liao, K. Kreutzer, S. W. Tjiang, and S. Devadas, A New Viewpoint on Code Generation for Directed Acyclic Graphs, ACM Trans. Design Automation of Electronic Systems, 3 (1998) 51-75.

[13] D. Le Berre, and L. Simon, The Essentials of the SAT 2003 Competition, in: E. Giunchiglia, A. Tacchella (Eds.), Proceedings of the 6th International Conference on Theory and Applications of Satisfiability Testing (SAT'03), Lecture Notes in Computer Science, Vol. 2919, pp. 172-187, Springer-Verlag, 2004.

[14] B. Monien, E. Speckenmeyer, and O. Vornberger, Upper Bounds for Covering Problems, Methods of Operations Research 43 (1981) 419-431.

[15] S. Porschen, On Some Weighted Satisfiability and Graph Problems, in: "P. Vojtas, et al. (Eds.), Proceedings of the 31st Conference on Current Trends in Theory and Practice of Informatics (SOFSEM 2005)", Lecture Notes in Comp. Science, Vol. 3381, pp. 278-287, Springer-Verlag, 2005.

[16] S. Porschen, Solving Minimum Weight Exact Satisfiability in $O(2^{0.2441\ n})$ Time, in: "X. Deng, D. Du (Eds.), Proceedings of the 16th International Symposium on Algorithms and Computation (ISAAC 2005)", Lecture Notes in Comp. Science, Vol. 3827, pp. 654-664, Springer-Verlag, 2005.

[17] S. Porschen, Counting All Solutions of Minimum Weight Exact Satisfiability, in: "T. Calamoneri, I. Finocchi, and G. F. Italiano (Eds.), Proceedings of the 6th Italian Conference on Algorithms and Complexity (CIAC 2006)", Lecture Notes in Comp. Science, Vol. 3998, pp. 50-59, Springer-Verlag, 2006.

[18] S. Porschen, B. Randerath, and E. Speckenmeyer, Exact 3-Satisfiability is Decidable in Time $O(2^{0.16254n})$, Annals of Mathematics and Artificial Intelligence 43 (2005) 173-193.

[19] T. J. Schaefer, The complexity of satisfiability problems, in: Proceedings of the 10th ACM Symposium on Theory of Computing, pp. 216-226, 1978.

[20] L. Valiant, The complexity of enumeration and reliability problems, SIAM J. Comput. 9 (1979) 410-421.