Partial Minimum Satisfiability: Fine-Grained Analysis

IVAN BLIZNETS
DANIL SAGUNOV*, ITMO University, Russia
KIRILL SIMONOV, Department of Informatics, University of Bergen, Norway

There is a well-known approach to cope with NP-hard problems in practice: reduce the given problem to SAT or Max-SAT and run a SAT or a Max-SAT solver. This method is very efficient since SAT/Max-SAT solvers are extremely well-studied, as well as the complexity of these problems. At IJCAI-2011, Li et al. proposed an alternative to this approach and suggested the Partial Minimum Satisfiability problem as a reduction target for NP-hard problems. They developed the MinSatz solver and showed that reducing to Partial Min-SAT and using MinSatz is in some cases more efficient than reductions to SAT or Max-SAT. Since then many results connected to the Partial Min-SAT problem were published. However, to the best of our knowledge, the worst-case complexity of Partial Min-SAT has not been studied up until now. Our goal is to fix the issue and show a $O^*((2-\epsilon)^m)$ lower bound under the SETH assumption (here m is the total number of clauses), as well as several other lower bounds and parameterized exact algorithms with better-than-trivial running time.

JAIR Associate Editor: Kuldeep Meel

JAIR Reference Format:
Ivan Bliznets, Danil Sagunov, and Kirill Simonov. 2026. Partial Minimum Satisfiability: Fine-Grained Analysis. *Journal of Artificial Intelligence Research* 85, Article 40 (April 2026), 20 pages. DOI: 10.1613/jair.1.19378

1 Introduction

For many computationally hard problems, the best way to cope with their intractability in practice is to reduce them to SAT or Max-SAT and then run a SAT/Max-SAT solver. The main reason for this is the existence of the whole industry studying exactly these problems. There are annual conferences and competitions solely dedicated to these problems, like SAT[1], MSE (MaxSAT Evaluation[2]), and others. At IJCAI-2011, Li et al. (C. M. Li, Zhu, Manya, et al. 2011) proposed an alternative to this approach and suggested reducing NP-hard problems to the following Partial Min-SAT problem.

Partial Min-SAT	
Input:	A formula ϕ in CNF, where each clause is either hard or soft; an integer k.
Question:	Does there exist an assignment of variables of ϕ satisfying all hard clauses, and at most k soft clauses?

In the same paper, Li et al. (C. M. Li, Zhu, Manya, et al. 2011; C. M. Li, Zhu, Manyà, et al. 2012) developed the MinSatz solver and showed that reducing to Partial Min-SAT and solving with MinSatz is in some cases more

*Corresponding Author.
[1] http://satisfiability.org/SAT26/
[2] https://maxsat-evaluations.github.io/2026/

Authors' Contact Information: Ivan Bliznets, iabliznets@gmail.com; Danil Sagunov, ORCID: 0000-0003-3327-9768, danilka.pro@gmail.com, ITMO University, Saint Petersburg, Russia; Kirill Simonov, ORCID: 0000-0001-9436-7310, kirillsimonov@gmail.com, Department of Informatics, University of Bergen, Bergen, Norway.

DOI: 10.1613/jair.1.19378

SAT	
Running time	References
$O^*(1.260^m)$	(Monien and Speckenmeyer 1985)
$O^*(1.239^m)$	(Hirsch 1998)
$O^*(1.234^m)$	(Yamamoto 2005)
$O^*(1.2226^m)$	(Chu et al. 2021)

Max-SAT	
Running time	References
$O^*(1.3803^m)$	(Niedermeier and Rossmanith 1999)
$O^*(1.3412^m)$	(Bansal and Raman 1999)
$O^*(1.3248^m)$	(Chen and Kanj 2004)
$O^*(1.2989^m)$	(Xu et al. 2019)
$O^*(1.2886^m)$	(Xiao 2022)

Table 1. Progress for SAT and Max-SAT in terms of m (total number of clauses). O^* omits factors polynomial in n and m.

efficient than a reduction to SAT or Max-SAT. We also observe that the language of the Partial Min-SAT problem allows to succinctly encode a number of classical problems such as Vertex Cover, Set Cover, Independent Set, Hitting Set, and many others.

Since the introduction of the problem, Partial Min-SAT were approached with local-search algorithms (Abramé and Habet 2015), iterative methods (Hers et al. 2012), inference-based exact algorithms (C.-M. Li and Manya 2015), and generalized to a falsifiability problem (Ignatiev, Morgado, Planes, et al. 2016). The MinSatz solver was empirically evaluated on solving Independent Set (Ignatiev, Morgado, and Marques-Silva 2014) and gained performance from alternative encodings (Zhu et al. 2012). Note that some approximation results were known for Partial Min-SAT even before (C. M. Li, Zhu, Manya, et al. 2011), for example (Avidor and Zwick 2005; Escoffier and Paschos 2007; Kohli et al. 1994; Marathe and Ravi 1996). However, to the best of our knowledge, the general case of Partial Min-SAT was not studied up to these days from the worst-case complexity point of view. This is drastically different from the case of SAT and Max-SAT: for those problems there is a chain of non-trivial algorithms with improving bounds on the worst-case complexity. In Table 1 we list the algorithms whose running time depends on the overall number of clauses in the input formula. Note that the latest results were presented quite recently at IJCAI-2019 (Xu et al. 2019), AAAI-2021 (Chu et al. 2021), and IJCAI-2022 (Xiao 2022) after 15 years of no progress.

Practical relevance. While the search for the best worst-case guarantee of an algorithm for SAT and its variants poses an intriguing theoretical question, it is also well-motivated from the practical perspective.

Both Min-SAT and Max-SAT have emerged as a valuable formalism in artificial intelligence, applicable to a variety of optimization problems such as planning, diagnosis, team formation, and FPGA circuit routing (C. M. Li and Manya 2021). Min-SAT has distinct advantages, notably its meaningfulness in handling both satisfiable and unsatisfiable problem instances. Specialized Min-SAT solvers, such as MinSatz, demonstrate competitive performance in optimization tasks (C. M. Li, Zhu, Manya, et al. 2011). Furthermore, advances in logical calculi and sophisticated transformations between Min-SAT and Max-SAT allow practitioners to select the most suitable formalism based on the specific characteristics of a given problem scenario. As we demonstrate later in this paper, Partial Min-SAT has even more natural expressive power than Min-SAT, providing wider choice for practitioners.

In multi-agent systems involving large foundational models, Partial Min-SAT shows considerable promise as a method for modeling and optimizing complex agent-based decision-making processes. Each agent in these systems can be viewed as solving an instance of Partial Min-SAT, characterized by a combination of mandatory (hard) and optional (soft) constraints. Employing Partial Min-SAT in this context facilitates the optimization of collective agent behavior, balancing mandatory system requirements against the minimization of undesirable outcomes. Thus, integrating Partial Min-SAT within large-scale foundational multi-agent models can significantly enhance the accuracy, robustness, and efficiency of control algorithms, particularly within dynamic environments featuring complex structural constraints.

Tools and reduction/branching rules used in algorithms for the worst case have been consistently reused in the implementations of practical solvers. In the case of SAT, the most well-known such rules are pure literal elimination, resolution, and unit propagation. The main motivation of our study is to extend this perspective to the more expressive Partial Min-SAT problem.

Our results. In particular, we are interested in running time bounds in terms of m, where m is the overall number of clauses. One may expect that the behavior of Partial Min-SAT is similar to that of SAT and Max-SAT since the problems are tightly related, and from (Kügel 2012) we know a natural transformation from Partial Min-SAT to Partial Max-SAT and vice versa. Quite surprisingly the situation turns out to be significantly different. It is easy to see that pure literal elimination does not work for Partial Min-SAT, as assigning a pure literal to true can undesirably satisfy soft clauses. Also, Set Cover can be reduced to Partial Min-SAT and all literals will be positive. Hence, Partial Min-SAT is NP-hard even if all literals are positive. Unfortunately, the usage of the resolution rule is also restricted even if the variable appears only once positively and once negatively, since Independent Set can be reduced to this special case of Partial Min-SAT. Hence, Partial Min-SAT is NP-hard even if each variable appears once as a positive literal and once as a negative literal.

While these observations leave a possibility that a better-than-trivial algorithm for Partial Min-SAT might be obtained by using a more sophisticated approach, in Theorem 1 we show that this is most likely not the case. We prove that there is no algorithm for Partial Min-SAT with running time $O^*((2-\epsilon)^m)$ for any $\epsilon > 0$, unless the Strong Exponential Time Hypothesis (SETH) fails. Recall that SETH is a complexity hypothesis that implies that there is no $O^*((2-\epsilon)^n)$ algorithm for SAT for any $\epsilon > 0$, where n is the number of variables. Thus, Partial Min-SAT does not have algorithms with non-trivial running time both in terms of n and m if SETH is true. ($O^*(2^n)$ and $O^*(2^m)$ running time algorithms for Partial Min-SAT are straightforward.)

On the positive side, we present several non-trivial algorithms that are characterized by a broader range of natural parameters such as n (number of variables), s (number of soft clauses), h (number of hard clauses), q (size of the largest hard clause), t_s (number of variables that have positive and negative literals inside soft clauses). On instances where each clause has length at most two, Partial Min-SAT can be solved in $O^*(2^{\omega n/3})$ time (here ω is the matrix multiplication exponent). If $q \leq 2$ then Partial Min-SAT can be solved in time $O^*(c_{vc}^s)$ where c_{vc} is the lowest constant such that Vertex Cover admit $O^*(c_{vc}^{n'})$ running time algorithm on graphs with n' vertices. (The currently best bound of $c_{vc} = 1.1996$ is given by (Xiao and Nagamochi 2017)). Moreover, Partial Min-SAT can be solved in $O^*(2^{\min\{n,k,t_s\}} \cdot (2-\frac{1}{q})^{n-\min\{n,k,t_s\}})$ time and if $q \geq 3$ then there is a $O^*(1.6181^h \cdot (2-\frac{1}{q})^s)$ running time algorithm for Partial Min-SAT. Finally for the general case we present an algorithm with the running time $O^*(1.755^{\frac{m+n}{2}})$.

Organization. The rest of the paper is organized as follows. In Section 2 we give definitions that are used throughout this work and demonstrate that certain NP-hard problems can be naturally encoded as instances of Partial Min-SAT. In Section 3 we establish a connection between Partial Min-SAT and a certain set union problem that is crucial in the proof of Theorem 1. In Section 4 we show Theorem 1 and other parameterized complexity lower bounds. Finally, in Section 5 we present our positive algorithmic results.

2 Preliminaries

2.1 Standard Definitions

For an integer n, we denote by $[n]$ the set $\{1, 2, \ldots, n\}$. For a graph G, $V(G)$ and $E(G)$ denote the set of vertices and the set of edges of the graph, respectively. The $O^*(\cdot)$ notation omits polynomial factors in the size of the instance: For functions f and g, the statement that $g(n) = O^*(f(n))$ for any instance size $n > C$ for some C, is equivalent to that there exists $c > 0$ such that $g(n) = O(f(n)) \cdot n^c$.

2.1.1 Exact Exponential Algorithms. Exact exponential algorithms are commonly based on reduction rules and branching rules. A *reduction rule* is an algorithm that takes an instance of the problem and returns an equivalent but smaller instance of the same problem. A *branching rule* is an algorithm that an instance of the problem and returns several instances of the same problem, with the property that the original instance is a yes-instance if and only if at least one of the returned instances is a yes-instance. Typically, to analyze the time complexity of a branching algorithm, a *measure* of an instance is introduced, that is, a mapping that takes an instance of the problem and returns an integer. Then it is shown that each of the branches leads to an instance that has smaller measure than the original instance by at least some margin. This propery is encapsulated in the definition of a *branching vector*: A branching rule with k branches has a branching vector of $(q_1, \ldots, q_k)$, where $q_1 \leq \ldots \leq q_k$, if the first branch decreases the measure of the instance by at least q_1, the second by at least q_2, and so on. Usually the same branching rule is applied exhaustively, leading to a recursive tree where the branching rule is invoked in each node. Whenever a branching vector is known, a *branching factor* of such a recursive algorithm can be computed by solving a recursive expression: If an algorithm has a branching factor of α, then its running time is at most $O^*(\alpha^n)$, if the starting measure of the instance is defined by n, and the final measure is always nonnegative. For example, a branching rule with a branching vector of (1, 1) leads to a full binary tree of depth n, which corresponds to the running time $O^*(2^n)$ and the branching factor of 2. A recursive branching procedure with a branching vector $(q_1, \ldots, q_k)$ is also called a *branching of type* $(q_1, \ldots, q_k)$ or a $(q_1, \ldots, q_k)$*-branching*.

2.1.2 Satisfiability. The problems in our study deal with satisfiability of Boolean formulas. The basic unit of a formula is a *variable*. A *literal* is either a variable or its negation. A *clause* is a disjunction of literals. We say that a variable appears *positively* in a clause if the clause contains a literal of this variable without negation, and a variable appears *negatively* in a clause if the clause contains a literal of this varaible with negation. A *CNF formula* is a conjunction of clauses. We may also use the term *formula* instead, as we only consider CNF formulas throughout the paper.

Now, the SAT problem is defined as follows. Given a CNF formula ϕ, the task is to determine whether there is an assignment of variables to $\{\text{true}, \text{false}\}$ such that every clause of ϕ is satisfied. The Max-SAT problem is a maximization version of SAT: Given a formula ϕ and an integer k, is there an assignment that satisfies at least k clauses of ϕ? Min-SAT is defined symmetrically, minimizing the number of satisfied clauses. In the Min-Ones-SAT problem, the input is a formula ϕ and an integer k, and the task is to find a satisfying assignment that sets at most k variables to true.

Commonly, variants of SAT are studied for formulas with bounded clause length. For an integer $q \geq 2$, a q-SAT is a special case of SAT where each clause is a conjunction of at most q literals. The problems q-Max-SAT, q-Min-SAT, q-Min-Ones-SAT are defined analogously. For all variants of SAT, we denote by n the number of variables and by m the total number of clauses, unless specified otherwise.

2.1.3 Parameterized Complexity, ETH and SETH. A *parameterized problem* is a language $Q \subseteq \Sigma^* \times \mathbb{N}$ where Σ^* is the set of strings over a finite alphabet Σ. Respectively, an input of Q is a pair (I, k) where $I \subseteq \Sigma^*$ and $k \in \mathbb{N}$; k is the *parameter* of the problem. A parameterized problem Q is *fixed-parameter tractable* (FPT) if it can be decided whether $(I, k) \in Q$ in time $f(k) \cdot |I|^{O(1)}$ for some function f that depends of the parameter k only. Respectively, the parameterized complexity class FPT is composed by fixed-parameter tractable problems. The W-hierarchy is a collection of computational complexity classes: we omit the technical definitions here. The following relation is known amongst the classes in the W-hierarchy: $\mathsf{FPT} = \mathsf{W}[0] \subseteq \mathsf{W}[1] \subseteq \mathsf{W}[2] \subseteq \ldots \subseteq \mathsf{W}[P]$. It is widely believed that $\mathsf{FPT} \neq \mathsf{W}[1]$, and hence if a problem is hard for the class $\mathsf{W}[i]$ (for any $i \geq 1$) then it is considered to be fixed-parameter intractable.

We also provide conditional lower bounds by making use of stronger assumptions: the Exponential Time Hypothesis of Impagliazzo, Paturi, and Zane, and the Strong Exponential Time Hypothesis of Impagliazzo and Paturi. To formulate both, for each integer $q \geq 3$ denote by c_q the infimum of the values δ such that q-SAT can be

solved in time $O(2^{\delta n})$. ETH conjectures that $s_3 > 0$, and SETH that the limit of $s_3, s_4, \ldots$ is exactly 1. The classical implication of ETH is that 3-SAT cannot be solved in time $2^{o(n)}(n+m)^{O(1)}$, and SETH implies that SAT cannot be solved in time $O^*((2-\epsilon)^n)$ for any $\epsilon > 0$. In our hardness proofs we also use the following well-known corollary of ETH: Clique cannot be solved in time $n^{o(k)}$ where n is the number of vertices in the graph and k is the target size of the clique (Cygan, Fomin, et al. 2015). The same holds for the colored version of Clique, Multicolored Clique, where the set of vertices is partitioned into k colors, and the goal is to find a clique of size k that contains one vertex of each color.

2.2 Encodings

In order to demonstrate the expressive power of Partial Min-SAT we present encodings of a few NP-hard problems as Partial Min-SAT problem.

2.2.1 Hitting Set. In the Hitting Set one is given an universe $\mathcal{U} = \{1, 2, 3, \ldots, n\}$ and family $\mathcal{F} = \{S_1, S_2, \ldots, S_m\}$ of subsets of universe $\mathcal{U}$. The goal is to find a subset $X \subset \mathcal{U}$ of the smallest size such that for any $i \in \{1, 2, \ldots, m\}$ $S_i \cap X \neq \emptyset$. Let $S_i = \{j_{i,1}, j_{i,2}, \ldots j_{i,i_k}\}$. The equivalent instance Partial Min-SAT is the following:

- the set of soft clauses is $\mathcal{S} = \{x_1, x_2, \ldots, x_n\}$ i.e. each clause consists of exactly one variable as a positive literal.
- for each set S_i we have a hard clause $(x_{j_{i,1}} \vee x_{j_{i,2}} \vee \cdots \vee x_{j_{i,i_k}})$ i.e. the set of hard clauses $\mathcal{H}$ consist of m clauses,

2.2.2 Vertex Cover. In the Vertex Cover one is given a graph $G = (V, E)$ and the goal is to find a minimum size subset of vertices X such that each edge from E has at least one endpoint in X. Let $V = \{v_1, v_2, \ldots v_n\}$. In order to construct an equivalent instance of Partial Min-SAT we create variables x_{v,e_v} for each $v \in V$ and for each edge e_v incident to vertex v. First of all for each edge $e = uv$ we create a hard clause $(x_{v,e} \vee x_{u,e})$. Moreover for each vertex v and a set $\{e_1, \ldots e_k\}$ of all incident edges to it we create a soft clause $(x_{v,e_1} \vee x_{v,e_2} \vee \ldots x_{v,e_k})$. Taking into account that vertex cover is NP-hard even in cubic graphs we obtain that Partial Min-SAT is NP-hard even on instances where each variable appears exactly two times, the length of all hard clauses is 2 and the length of all soft clauses is 3.

2.2.3 Set Cover. In the Set Cover we are given a universe $\mathcal{U} = \{1, 2, 3, \ldots, n\}$ and family $\mathcal{F} = \{S_1, S_2, \ldots, S_m\}$ of subsets of universe $\mathcal{U}$. The goal is to find the smallest set $X = \{i_1, i_2, \ldots, i_k\}$ such that $\mathcal{U} = S_{i_1} \cup S_{i_2} \cup \cdots \cup S_{i_k}$. In order to reduce the Set Cover problem to Partial Min-SAT for each set $S_i = \{j_{i,1}, j_{i,2}, \ldots, j_{i,i_p}\}$ we create variables $x_{S_i,j_{i,1}}, x_{S_i,j_{i,2}}, \ldots, x_{S_i,j_{i,i_p}}$. Additionally we create a soft clause $(x_{S_i,j_{i,1}} \vee x_{S_i,j_{i,2}} \vee \cdots \vee x_{S_i,j_{i,i_p}})$ for each set S_i from $\{S_1, S_2, \ldots, S_m\}$. Assume that element ℓ is contained exactly in sets $S_{\ell_1}, \ldots S_{\ell_q}$ then we add a hard clause $(x_{S_{\ell_1},\ell} \vee x_{S_{\ell_2},\ell} \vee \cdots \vee x_{S_{\ell_q},\ell})$ for each $\ell \in \{1, 2, \ldots, n\}$.

2.2.4 Dominating Set. Since the Dominating Set problem is a special case of Set Cover it also has a natural encoding as a Partial Min-SAT.

2.2.5 Independent Set. In the Independent Set one is given a graph $G = (V, E)$ and the goal is to find a largest subset of vertices I such that there is no each edge from E that has both its endpoints in I. For each edge $e \in E$ we create a variable x_e. Let orient each edge arbitrarily. For each vertex we create a soft clause. Let $e_1, \ldots, e_q$ be incoming edges for vertex v and $f_1, f_2, \ldots, f_p$ be outgoing edges from vertex v then we create a soft clause $(x_{e_1} \vee \cdots \vee x_{e_q} \vee \neg x_{f_1} \vee \cdots \vee \neg x_{f_p})$. It is easy to see that number of satisfied clauses is not smaller than size of vertex cover as each edge force one of the clauses corresponding to its endpoints to be satisfied. Moreover, it is possible to satisfy at most vertex cover clauses. In this case the number of falsified clause will correspond to size of largest Independent set.

2.2.6 Maximum Clique. In the Clique one is given a graph $G = (V, E)$ and the goal is to find a largest subset of vertices C such that for each pair of verices $u, v \in C$ there is and edge uv in graph G. In order to construct an equivalent Partial Min-SAT. We create variables $x_1, \ldots, x_n$ corresponding to vertices from V add soft clauses $(\neg x_1), \ldots, (\neg x_n)$, as well for each edge ij create a hard clause $(\neg x_i \vee \neg x_j)$.

2.2.7 Minimum Union. In the Minimum Union problem one is given an integer k as well as a family $\mathcal{F} = \{S_1, S_2, \ldots, S_m\}$ of subsets from a universe $\mathcal{U} = \{1, 2, \ldots, n\}$ the goal is to find a set $X = \{j_1, j_2, \ldots, j_k\}$ such that $X \subset \{1, 2, \ldots, m\}$ and $S_{j_1} \cap S_{j_2} \cap \cdots \cap S_{j_k}$ has the smallest number of elements. Let $S_i = \{j_{i,1}, j_{i,2}, \ldots, j_{i,i_p}\}$. We create m variables $x_1, \ldots, x_m$ and n soft clauses (assign numbers to this clauses from 1 to n) such that the literal x_i is contained in soft clauses with numbers from S_i. In this case if x_i is true then clauses with numbers from S_i are satisfied. Additionally we create a CNF formula G such that G is satisfied only if $x_1 + x_2 + \cdots + x_m \geq k$. Hard clauses of our Partial Min-SAT match exactly with G (note that G mu depend on some new variables different from $x_1 + x_2 + \cdots + x_m \geq k$).

2.2.8 Partial Maximum Satisfiability. Partial Maximum Satisfiability can be easily transformed into Partial Min-SAT using Kugel's transformation (Kügel 2012): each soft clause $(x_1 \vee x_2 \vee \cdots \vee x_k)$ should be replaced with the following soft clauses: $(\neg x_1), (x_1 \vee \neg x_2), (x_1 \vee x_2 \vee \neg x_3), \ldots, (x_1 \vee x_2 \vee x_3 \vee \cdots \vee x_{k-1} \vee \neg x_k)$. If the clause $(x_1 \vee x_2 \vee \cdots \vee x_k)$ is not satisfied by some assignment then under the same assignment all clauses $(\neg x_1), (x_1 \vee \neg x_2), (x_1 \vee x_2 \vee \neg x_3), \ldots, (x_1 \vee x_2 \vee x_3 \vee \cdots \vee x_{k-1} \vee \neg x_k)$ are satisfied. Moreover, if the clause $(x_1 \vee x_2 \vee \cdots \vee x_k)$ is satisfied by some assignment then the same assignment satisfy exactly $k-1$ clauses among $(\neg x_1), (x_1 \vee \neg x_2), (x_1 \vee x_2 \vee \neg x_3), \ldots, (x_1 \vee x_2 \vee x_3 \vee \cdots \vee x_{k-1} \vee \neg x_k)$. Even though the transformation is easy, it can potentially increase the number of clauses by factor n. Note that the same transformation applied Partial Min-SAT will reduce problem into Partial Maximum Satisfiability problem.

3 Minimum Union

In this section, we show that Partial Min-SAT is closely related to a natural set problem, Minimum Union, which asks to find a collection of k sets with minimum union size. This problem was recently studied in (Agrawal and Maity 2021) from the parameterized point of view. We shall also use this connection to show parameterized lower bounds for Partial Min-SAT. Throughout the paper, we stick to the following multicolored version of Minimum Union.

Multicolored Minimum Union	
Input:	An integer n, k collections of subsets of $[n]$ $\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_k \subseteq 2^{[n]}$, and an integer ℓ.
Question:	Does there exist a choice of $S_1, S_2, \ldots, S_k$ such that $S_i \in \mathcal{F}_i$ and $\lvert \bigcup S_i \rvert \leq \ell$?

Lemma 1. *Multicolored Minimum Union is equivalent to the special case of Partial Min-SAT when all variables appear positively in the input formula. Moreover, in the corresponding Partial Min-SAT instance the number of hard clauses equals k, the number of soft clauses equals n, the number ℓ is the number of satisfied soft clauses and the variables correspond to the sets in the instance of Multicolored Minimum Union.*

Proof. To construct an instance of Partial Min-SAT from an instance of Multicolored Minimum Union, it is enough to associate a soft clause with each element of the universe $[n]$. Each of the k color groups is associated with a hard clause. With each unique universe subset in the instance, a unique variable is associated. Hence, the number of variables of the constructed instance equals the number of unique subsets, the number of soft clauses equals n and the number of hard clauses equals k. Then i^{th} hard clause is a disjunction of the variables corresponding to the subsets in $\mathcal{F}_i$. While j^{th} soft clause is a disjunction of all variables corresponding to the subsets containing element j of the universe.

It is then asked to satisfy all hard clauses but at most ℓ soft clauses simultaneously. The value of a variable in an assignment corresponds to whether we should take the corresponding subset in the solution or not. Hard clauses ensure that at least one set is picked from each $\mathcal{F}_i$, while each soft clause evaluates whether the corresponding element is covered by the chosen subsets. Thus, the constructed instance of Partial Min-SAT is equivalent to the initial instance of Multicolored Minimum Union.

To construct an instance of Multicolored Minimum Union from an instance of Partial Min-SAT with all-positive literals, one need to proceed in a similar fashion as above, but in the opposite direction: associate a universe element with each soft clause; associate a subset with each variable that consists exactly elements corresponding to soft clauses containing this variable; finally associate with each hard clause a subset family containing subsets corresponding to the variables comprising this hard clause.

Below, we provide an example of reduction from Partial Min-SAT with all-positive literals to Multicolored Minimum Union. Assume, we have the following instance of Partial Min-SAT:

hard clauses: $(x \vee y)$, $(y \vee z)$
soft clauses: $(x \vee y \vee t)$, $(t \vee z)$, $(y \vee z \vee t)$.

An equivalent instance of Multicolored Minimum Union has $n = 3$ as the number of soft clauses is three, $k = 2$ as the number of hard clauses is two. $\mathcal{F}_1 = \{\{1\}, \{1, 3\}\}$ as the literal x from the hard clause $(x \vee y)$ appears only in the first soft clause and a literal y appears in the first and the third soft clauses. Similarly, $\mathcal{F}_2 = \{\{1, 3\}, \{2, 3\}\}$ as the literal z from the hard clause $(y \vee z)$ appears in the second and the third soft clauses. □

While Multicolored Minimum Union is thus a special case of Partial Min-SAT, Partial Min-SAT still can be reduced to Multicolored Minimum Union in FPT time.

Lemma 2. *Partial Min-SAT parameterized by $k + h$ admits a parameterized reduction to Multicolored Minimum Union parameterized by $\ell + k$.*

Before we move on to the proof of Lemma 2, we show a useful result about Multicolored Minimum Union that allows to compose several instances of the problem into one.

Lemma 3. *There is a polynomial-time algorithm that, given $t > 1$ instances $I_1, I_2, \ldots, I_t$ of Multicolored Minimum Union, where $I_j = (n_j, k_j, \{F_{j,i}\}, \ell_j)$, outputs one instance $I' = (n', k', \{F'_i\}, \ell')$ of Multicolored Minimum Union, such that*

- *I' is a yes-instance if and only if there exists $j \in [t]$ such that I_j is a yes-instance;*
- *$k' = \max_{j=1}^{t} k_j$;*
- *$\ell' = (\lceil \log t \rceil + 1) \cdot (\max_{j=1}^{t} \ell_j + 1) - 1$;*
- *$n' \leq 2\ell' + \ell + \max_{j=1}^{t} n_j$.*

Proof. We present an algorithm $\mathcal{A}$. Let $k := \max_{j=1}^{t} k_j$, and $\ell := \max_{j=1}^{t} \ell_j$, and $n := \ell + \max_{j=1}^{t} n_j$. At its preliminary step, the algorithm modifies the instances $I_1, I_2, \ldots, I_t$, so (n_j, k_j, ℓ_j) are the same among $j \in [t]$. For each $j \in [t]$, the algorithm performs the following modification of I_j:

(1) For each $i \in \{k_j + 1, k_j + 2, \ldots, k\}$, add a family $\mathcal{F}_{j,i} = \{\emptyset\}$ to I_j. Increase k_j to k.
(2) For each $u \in \{n_j + 1, n_j + 2, \ldots, n_j + (\ell - \ell_j)\}$, add u to every set in $\mathcal{F}_{j,1}$. Increase ℓ_j to ℓ.
(3) Increase n_j to n.

It holds that the modified I_j is a yes-instance if and only if the initial I_j is a yes-instance. Moreover, the modification is done in polynomial time.

We denote the input families of the j^{th} instance by $\mathcal{F}_{j,1}, \mathcal{F}_{j,2}, \ldots, \mathcal{F}_{j,k}$. We assume that t is a power of two, otherwise $\mathcal{A}$ can copy the last instance as many times as needed. It holds that $t \geq 2 \log t$ since $t \geq 2$. $\mathcal{A}$ constructs the resulting instance of Multicolored Minimum Union in the following way.

The number of elements in the resulting instance equals $n' = n + 2\log t \cdot (\ell + 1)$, that is, we add $2\log t \cdot (\ell + 1)$ fresh elements to the universe. The fresh elements can be enumerated as $u_{a,b}$ for $a \in [2\log t]$ and $b \in [\ell + 1]$. We are interested in constructing t equal-sized subsets $X_1, X_2, \ldots, X_t$ of the set of the fresh elements such that $|X_i \setminus X_j| \geq (\ell + 1)$ for any distinct $i, j \in [t]$. One of the ways to do it is to first pick t distinct subsets $Y_1, Y_2, \ldots, Y_t$ of $[2\log t]$ of size $\log t$. Note that $\binom{2\log t}{\log t} \geq \frac{1}{2\log t} \cdot 2^{2\log t} \geq t$, so such sets always exist. Trivially, $|Y_i \setminus Y_j| \geq 1$. Then for each i put $X_i = \{u_{a,b} \mid a \in Y_i, b \in [\ell + 1]\}$, i.e. X_i consists of elements of Y_i each copied $\ell + 1$ times. It follows that $|X_i \setminus X_j| \geq \ell + 1$ while $|X_i| = \log t \cdot (\ell + 1)$.

Then for each $i \in [k]$, the set $\mathcal{F}_i'$ is constructed as follows. For each $j \in [t]$ and for each $S \in \mathcal{F}_{j,i}$, the set $S \cup X_j$ is added to $\mathcal{F}_i'$. This can be viewed as we append the set X_j to all sets of the j^{th} instance and then unite all instances into a single instance while preserving set colors. We finally put $\ell' = \log t \cdot (\ell + 1) + \ell$ and say that $I' = (n', k, \mathcal{F}_1', \mathcal{F}_2', \ldots, \mathcal{F}_k', \ell')$ is a yes-instance of Multicolored Minimum Union if and only if one of the $I_1, I_2, \ldots, I_t$ is a yes-instance of Multicolored Minimum Union.

For the proof in one direction, let I' be a yes-instance and $S_1', S_2', \ldots, S_k'$ be a solution to I', so $|\bigcup S_i'| \leq \ell'$. By construction of I, there are $j_1, j_2, \ldots, j_k$ such that $X_{j_p} \subseteq S_{j_p}'$ for each $p \in [k]$. If there are distinct j_p and j_q, we have that $|X_{j_p} \cup X_{j_q}| = |X_{j_p}| + |X_{j_q} \setminus X_{j_p}| = (\log t + 1) \cdot (\ell + 1) > \ell'$. Hence, $j_1 = j_2 = \ldots = j_k = j$. It follows that for each $i \in [k]$, $S_i' = S_i \cup X_j$, where $S_i \in \mathcal{F}_{j,i}$. Also $|\bigcup S_i| = |\bigcup S_i'| - |X_j| = \ell' - \log t \cdot (\ell + 1) = \ell$. Hence, I_j is a yes-instance of Multicolored Minimum Union.

Towards the other direction, let I_j be a yes-instance of Multicolored Minimum Union and $S_1, S_2, \ldots, S_k$ be a solution to I_j. Then if we pick for each $i \in [k]$ $S_i' = S_i \cup X_j$ we clearly obtain a viable solution to I'.

We obtain that $\mathcal{A}$ is a polynomial-time algorithm that satisfies all properties of the lemma. The proof is complete. □

We move on to proving Lemma 2.

Proof of Lemma 2. First we show a reduction from Partial Min-SAT to multiple instances of Multicolored Minimum Union, and then use Lemma 3 to replace them with one equivalent instance, so that our reduction is indeed a many-one parameterized reduction.

By Lemma 1, we have that a special case of Partial Min-SAT where all variables occur only positively, is equivalent to an instance of Multicolored Minimum Union. Observe that the reduction is polynomial, the original parameter k is transformed into ℓ, and the parameter h is transformed into k, thus the reduction is also parameter-preserving. It now remains to provide a parameterized reduction from Partial Min-SAT to its special case with only positive occurences. To this end, consider the following branching procedure.

If there is a variable x that occurs both positively and negatively in the instance of Partial Min-SAT, then branch on the assignment of this variable while eliminating clauses that become satisfied. Perform this repeatedly until there is no such variable, or one of the hard clauses becomes empty, or more than k soft clauses become satisfied. In the two latter cases, return a trivial no-instance of Multicolored Minimum Union. Otherwise we may assume that, in each branch, all literals in the remaining formula are positive. Return an equivalent Multicolored Minimum Union instance as discussed above.

The correctness clearly follows since on each step the branching exhausts all options for the variable assignment, while replacing the instance by an equivalent one in the leaf branches. As for the running time, on every branch at least one clause is satisfied since we only branch on variables that occur both positively and negatively. Moreover, at most k soft clauses can be satisfied on each branch as otherwise a trivial instance is returned immediately. Thus, the number of branches is at most $2^{O(k+h)}$.

Finally, since we obtain $2^{O(k+h)}$ instances of Multicolored Minimum Union each of polynomial size, applying Lemma 3 finishes the proof. □

4 Lower Bounds

4.1 Exact exponential algorithms

We start with the lower bounds for Partial Min-SAT, disproving faster-than-brute-force algorithms in terms of number of variables or number of clauses under SETH. The starting point of the reduction is the following result of (Cygan, Dell, et al. 2016) on the complexity of a variant of the Hitting Set problem.

c-Sparse-Hitting-Set	
Input:	A family $\mathcal{F} = \{S_1, S_2, \ldots, S_m\}$ of sets over $[n]$ such that $m \leq cn$, an integer ℓ.
Question:	Does there exist a set $S \subseteq [n]$ of size at most ℓ that intersects every set in $\mathcal{F}$?

Proposition 1. *Let $\delta \in (0,1)$ be an arbitrary real number. Unless SETH fails, there exists $c > 0$ such that c-Sparse-Hitting-Set cannot be solved in $O(2^{\delta n})$ running time.*

We move on to proving the lower bound.

Theorem 1. *Unless SETH fails, Partial Min-SAT cannot be solved in $O^*((2-\epsilon)^n)$ or $O^*((2-\epsilon)^m)$ running time for any $\epsilon > 0$.*

Proof. Since Partial Min-SAT with zero soft clauses is equivalent to SAT, the first part of this theorem is trivial.

To show the second part, we require sophisticated reductions. We provide a polynomial reduction from c-Sparse-Hitting Set over n elements to Partial Min-SAT where the number of clauses is bounded by $c'n$ for some constant $c' > 1$ that can be as close as needed to 1 in trade for running time. On the other hand, the number of variables in the resulting formula can be as high as $n^{f(c')}$ for a function f that grows as $c' > 1$ goes closer to 1. We then will show how from this reduction and a $O((2-\epsilon)^m)$ algorithm for Partial Min-SAT it follows that SETH fails through Proposition 1. Observe also that any $O^*((2-\epsilon)^m)$-time algorithm has a $O((2-\epsilon')^m)$ runtime bound for any $\epsilon' > \epsilon$.

Let $(n, S_1, S_2, \ldots, S_m, \ell)$ be the given instance of c-Sparse-Hitting Set. We construct an equivalent instance of Multicolored Minimum Union using a fixed integer p that is to be chosen later. The universe for instance of Multicolored Minimum Union is equivalent to the universe of the given instance. To construct the subset groups, split the m sets of the input instance into $\lceil m/p \rceil$ groups, each containing p sets, except possibly for the last one that can contain less than p sets.

Each group forms an instance of Hitting Set. To construct the family $\mathcal{F}_i$, take the i^{th} group and enumerate all inclusion-wise minimal hitting sets of the group. Clearly, the size of $\mathcal{F}_i$ is bounded by n^p, while the time required to construct it is bounded by $n^{O(p)}$. Thus, the instance of Multicolored Minimum Union consists of $\lceil m/p \rceil$ collections of sets. We finally ask to find $\lceil m/p \rceil$ sets from these collections such that their union is of size at most ℓ.

We claim that the constructed instance $(n, \lceil m/p \rceil, \mathcal{F}_1, \ldots, \mathcal{F}_{\lceil m/p \rceil}, \ell)$ is a yes-instance if and only if $(n, S_1, S_2, \ldots, S_m, \ell)$ is a yes-instance. This equivalence is clear from the fact that any optimal hitting set of $S_1, S_2, \ldots, S_m$ is a union of inclusion-wise minimal hitting sets for each of the $\lceil m/p \rceil$ groups.

Using Lemma 1, we finally construct an instance of Partial Min-SAT from the instance of Multicolored Minimum Union. The number of variables is bounded by n^p, while the total number of clauses equals $n + \lceil m/p \rceil \leq n + cn/p + 1 \leq n + cn/p + n/p$, as we reduce from c-Sparse-Hitting Set. Hence, the number of clauses m' in the constructed instance of Partial Min-SAT is bounded by $(1 + (c+1)/p)n$. Note that this number can be very small when compared to the number of variables n' in the constructed formula, that can be of order $n^{\Omega(p)}$.

Assume that the SETH holds and there exists a $O(2^{\delta' m'})$-time algorithm for Partial Min-SAT for some $\delta' < 1$, where m' is the number of clauses in the input formula. Take $\delta := \delta' + (1-\delta')/2 < 1$. By Proposition 1 there exists a constant c such that no $O(2^{\delta n})$ algorithm exists for c-Sparse-Hitting Set.

Then take $p := \lceil 2\delta'(c+1)/(1-\delta') \rceil$ and construct an algorithm for c-Sparse-Hitting Set using the reduction from Hitting Set to Partial Min-SAT and the $O(2^{\delta' m'})$-algorithm for Partial Min-SAT. The number of clauses m' is bounded by $(1+(c+1)/p)n \leq (1+(1-\delta')/2\delta')n$. The running time of the constructed algorithm is bounded by $O(2^{(\delta'+(1-\delta')/2)n})$, which is $O(2^{\delta n})$, and this contradicts Proposition 1. The proof is complete. □

4.2 Parameterized algorithms

Observe that for both SAT and Min-SAT there exists an algorithm that is single-exponential in the number of satisfied clauses i.e. in $h+k$: for SAT it is the straightforward $O^*(2^m) = O^*(2^h)$ algorithm, and for Min-SAT such an algorithm follows from the reduction to Vertex Cover (Marathe and Ravi 1996). It is thus natural to ask whether a similar algorithm exists for Partial Min-SAT, or at least one with running time $f(h+k) \cdot n^{O(1)}$ for some function f of $h+k$, the total number of satisfied clauses. The following theorem resolves the last question negatively, up to the standard parameterized complexity assumption that $\mathsf{W}[1] \neq \mathsf{FPT}$.

Theorem 2. *Partial Min-SAT is $\mathsf{W}[1]$-hard with respect to parameter $h+k$, i.e. the number of clauses to satisfy. Unless ETH fails, Partial Min-SAT does not admit an algorithm with running time $f(h+k) \cdot (n+m)^{o(\sqrt{h}+k)}$ for any computable function f.*

Proof. First, observe that Multicolored Minimum Union admits a polynomial reduction from Multicolored Clique: For an instance (G, k'), set the universe to be the set of vertices $V(G)$, set $\ell = k'$, $k = \binom{k'}{2}$, and for each pair of colors $\{c, c'\}$ introduce a collection of subsets $\mathcal{F}_{c,c'}$. For each edge $e \in E(G)$ introduce into $\mathcal{F}_{c,c'}$ a set that contains both endpoints of the edge, where c and c' are colors of the endpoints. Clearly, a union of $\binom{k'}{2}$ distinct edges from each color class contains at most k' vertices if and only if these edges form a colorful k'-clique, therefore the reduction is correct. Moreover, the reduction is parameterized with respect to $k+\ell$, thus Minimum Union parameterized by $k+\ell$ is $\mathsf{W}[1]$-hard.

Now, by Lemma 1, Multicolored Minimum Union is equivalent to a special case of Partial Min-SAT, where the parameter k of Multicolored Minimum Union is equal to the parameter h in Partial Min-SAT, and ℓ in Multicolored Minimum Union is equal to k in Partial Min-SAT. Therefore, Partial Min-SAT is $\mathsf{W}[1]$-hard when parameterized by $h+k$.

The second part of the theorem follows from the same chain of reductions. Since Multicolored Clique does not admit a $f(k') \cdot n^{o(k')}$-time algorithm assuming ETH for any compitable function f, Multicolored Minimum Union does not admit a $f(k+\ell) \cdot (n+m)^{o(\sqrt{k}+\ell)}$-time algorithm under the same assumption as $k = \binom{k'}{2}$ and $\ell = k$. From the last reduction we obtain that Partial Min-SAT cannot be solved in time $f(h+k) \cdot (n+m)^{o(\sqrt{h}+k)}$, assuming ETH. □

5 Algorithms

Having established the main intractability result for Partial Min-SAT (Theorem 1), we turn our attention to special cases where algorithms with non-trivial running times exist.

5.1 2-CNF

One notable "easy" case of SAT that is still fairly expressive is 2-SAT, which is well-known to be solvable in polynomial time. For Partial Min-SAT however, there is little hope for a polynomial algorithm in this case, as even 2-Min-SAT is known to be NP-complete (Kohli et al. 1994). On the positive side, Min-SAT with arbitrary clause length allows an efficient reduction to Vertex Cover (Marathe and Ravi 1996), where the number of

clauses in the formula is exactly transferred to the number of vertices in the graph. In particular, any algorithm that solves Vertex Cover in time $O^*(c^{n'})$, where n' is the number of vertices in the graph, immediately gives an algorithm for Min-SAT with running time $O^*(c^m)$.

A similar situation occurs with another closely related problem. In 2-Min-Ones-SAT, the input is a 2-CNF formula ϕ and an integer k, and the task is to determine whether ϕ can be satisfied by an assignment that sets at most k variables to true. (Misra et al. 2013) showed a reduction from 2-Min-Ones-SAT to Vertex Cover that transfers the number of variables in ϕ exactly to the number of vertices in the graph. Observe that Min-Ones-SAT is a special case of Partial Min-SAT where soft clauses are simply variables of the formula. Since Partial Min-SAT with hard clauses of length at most 2 generalizes both Min-SAT and 2-Min-Ones-SAT, a natural question is whether a similar reduction to Vertex Cover can be derived. In the next theorem, we show that this is indeed the case.

Theorem 3. *Partial Min-SAT with hard clause length bounded by 2 can be reduced to an instance (G, k) of Vertex Cover with $|V(G)| = s$ in polynomial time, if all hard clauses can be satisfied simultaneously. Moreover, there is a one-to-one correspondence between vertices of G and soft clauses. Any set of vertices in G is independent if and only if the corresponding set of clauses can be unsatisfied simultaneously in the initial instance. The target graph G may contain loops.*

To give an intuition, the reduction populates the target graph G with two kinds of edges: some pairs of soft clauses cannot be unsatisfied simultaneously because of a hard clause that prevents this (in fact, any clause in the transitive closure of the set of hard clauses, similar to (Misra et al. 2013)), and some because there is a variable that appears positively in one clause and negatively in the other. It is then left to show that forbidding these pairs is both necessary and sufficient for a Partial Min-SAT solution.

Proof. Denote the subformula of the Partial Min-SAT instance that corresponds to hard clauses by F, and by F^* the transitive closure of this formula. For a set of clauses F, the transitive closure is a minimal superset F^* of F such that for any two clauses $(\ell_1 \vee \ell_2)$ and $(\bar{\ell}_1 \vee \ell_3)$ in F^*, the clause $(\ell_2 \vee \ell_3)$ also belongs to F^*, where ℓ_1, ℓ_2, ℓ_3 are literals. We now define the graph G. The vertices of G are indexed by the soft clauses of the Partial Min-SAT instance. For each pair of soft clauses s_1 and s_2 (not necessarily distinct) there is an edge $s_1 s_2$ in G in one of the two cases.

- There is a literal ℓ such that $\ell \in s_1$ and $\bar{\ell} \in s_2$.
- There is a clause $f \in F^*$ of form $(\ell_1 \vee \ell_2)$, where ℓ_1 and ℓ_2 are literals, such that $\ell_1 \in s_1$ and $\ell_2 \in s_2$.

This finishes the definition of G. We now show that G satisfies the desired property.

First, assume there is an assignment t that satisfies F and unsatisfies a subset $I \subset S$ of soft clauses. We claim that I is an independent set in G. Assume the contrary, and let s_1 and s_2 be two soft clauses unsatisfied by t such that there is an edge in G between them. By construction of G, there is an edge between s_1 and s_2 in one of the two cases. In the first case, if there is a literal ℓ such that $\ell \in s_1$ and $\bar{\ell} \in s_2$, we immediately obtain a contradiction as in any assignment t either ℓ or $\bar{\ell}$ is assigned to true, which means that either s_1 or s_2 is satisfied. In the second case, there is a hard clause $f \in F$ of form $(\ell_1 \vee \ell_2)$ where $\ell_1 \in s_1$ and $\ell_2 \in s_2$. By assumption, t satisfies f, and therefore either s_1 or s_2 as well. Thus, in both cases we obtain a contradiction.

On the other hand, consider an independent set I in G. We construct an assignment t from I in the following three steps.

(1) For every literal ℓ appearing in clauses of I, set $t(\ell) = 0$.
(2) For every literal ℓ appearing in clauses of I and every clause $f \in F^*$ of form $(\ell \vee \ell')$, set $t(\ell') = 1$.
(3) Fix t^* to be an arbitrary satisfying assignment of F^*, and for each literal ℓ where t is still undefined, set $t(\ell) = t^*(\ell)$.

We now show that t is well-defined, and that t satisfies F. Observe that on step 1, no two opposite literals are set to "false" simultaneously: any two (not necessarily distinct) soft clauses s_1 and s_2 where $\ell \in s_1$ and $\bar{\ell} \in s_2$ are connected by an edge in G and thus cannot both be a part of I.

For step 2, we first show that no literal ℓ that is set to false on step 1, receives an opposite assignment on step 2. Assume the contrary, that is, there is a literal ℓ that appears in a clause s of I and also in a clause f of F^*, where $f = (\ell \vee \ell')$ and ℓ' appears in a clause s' of I. By construction of G, f induces an edge between s and s', which contradicts the assumption that both s and s' belong to I. Second, we show that no literal ℓ is set to both true and false on step 2. Starting again from the contrary, we get that there are clauses $f_1 = (\ell \vee \ell_1)$ and $f_2 = (\bar{\ell} \vee \ell_2)$ in F^* where ℓ_1 and ℓ_2 belong to clauses s_1 and s_2 in I, where both ℓ_1, ℓ_2 and s_1, s_2 are not necessarily distinct. By definition of F^*, the clauses f_1 and f_2 imply that there is also a clause $f = (\ell_1 \vee \ell_2)$ in F^*, which in turn implies an edge between s_1 and s_2 in G, a contradiction. Step 3 induces a well-defined assignment automatically: every literal that is not set on steps 1 and 2 receives a value from a fixed satisfying assignment of F. Moreover, after step 3 every literal receives a certain value in the assignment t.

We now argue that t satisfies all hard clauses. First, w.l.o.g. we assume that all the literals that receive the value "true" in t are positive (this is achieved by a simple equivalent substitution in the original formula). Denote the set of variables that are set on step i by X_i, where $i \in \{1, 2, 3\}$, that is, X_1 are varaibles appearing in I, X_2 are variables that are induced from the values of X_1 via clauses in F^*, and X_3 are the remaining variables. By construction, the sets X_i are disjoint and in total span all variables. Consider now the following list of cases for a hard clause $f \in F$.

Case 1: f contains only literals from X_1. If f contains at least one positive literal, f is automatically satisfied. If, on the other hand, f is comprised of two negative literals, then the corresponding soft clauses in I are connected by an edge of G, which contradicts the fact that I is an independent set, meaning that there no clauses f of this form.

Case 2: f contains exactly one literal from X_1. If the literal from X_1 is positive, f is automatically satisfied. Otherwise, f is of form $(\bar{x}_1 \vee x)$ where $x_1 \in X_1$; then x is necessarily in X_2 since step 2 holds for all f of this form. Since x is set to true on step 2, f is satisfied.

Case 3: f contains only literals from X_2. If f is not satisfied then f is of form $(\bar{x}_2 \vee \bar{x}'_2)$ where both x_2 and x'_2 belong to X_2. We show that this situation cannot occur. Assume the contrary, since $x_2 \in X_2$, there is a clause $f' \in F^*$ of form $(\bar{x}_1 \vee x_2)$, where $x_1 \in X_1$. By definition of transitive closure, from f and f' it follows that there is a clause of form $(\bar{x}_1 \vee \bar{x}'_2)$ in F^*. Since $x'_2 \in X_2$, there is also a clause of form $(\bar{x}'_1 \vee x'_2)$ in F^*, where $x'_1 \in X_1$. These two clauses imply a clause $(\bar{x}_1 \vee \bar{x}'_1)$ in F^*, which means that there is an edge between the corresponding soft clauses. This contradicts the starting assumption that the set I is independent.

Case 4: f contains one literal from each X_2 and X_3. If the variable from X_2 is positive in f, the clause is satisfied. We claim that the other option does not occur. Assume f is of form $(\bar{x}_2 \vee \ell_3)$, where ℓ_3 is either x_3 or $\bar{x}_3$ for some variable $x_3 \in X_3$. Since $x_2 \in X_2$, there exists a clause $f' \in F^*$ of form $(\bar{x}_1 \vee x_2)$ where $x_1 \in X_1$. By definition of transitive closure, f and f' imply that there is also a clause $f'' \in F^*$ of form $(\bar{x}_1 \vee \ell_3)$. Then, ℓ_3 is set to true on step 2, which is a contradiction to $x_3 \notin X_2$.

Case 5: f contains only literals from X_3. In this case, f is automatically satisfied since on variables of f, t coincides with t^* which is a satisfying assignment of f.

Clearly, every clause f in F falls into one of **Cases 1–5**. □

Composing our reduction with the best-known exact algorithm for Vertex Cover (Xiao and Nagamochi 2017), we obtain the following.

Corollary 1. *Partial Min-SAT with hard clause length bounded by 2 is solvable in time $O^*(1.1996^s)$.*

Restricting clauses to length 2 allows us to break the barrier of Theorem 1 in terms of the number of variables as well. The following algorithm stems from the matrix-multiplication-based Maximum Cut algorithm with the same running time due to (Williams 2007).

Theorem 4. *There is a $O^*(2^{\omega n/3})$ running-time algorithm for Partial Min-SAT restricted to formulas in 2-CNF.*

Proof. The algorithm first splits variables in three groups, each of size $\lceil n/3 \rceil$ or $\lfloor n/3 \rfloor$. Then construct a tripartite graph, where each part corresponds to a group of variables. The part corresponding to a group of size x contains exactly 2^x vertices, each corresponding to one of the 2^x assignments of variables in these group.

Then note that variables of each clause of the input formula fall either into a single group or into a pair of groups. For hard clauses that belong to a single group, we can immediately exclude assignments of this group that do not satisfy this clause. We remove the corresponding vertices from the graph. For the vertices that remain, we assign to each vertex a weight equal to the number of soft clauses that belong to this single group and are satisfied by the corresponding assignment.

To construct the edges of the graph, consider each pair (u, v) of vertices belonging to distinct parts. This pair corresponds to a specific assignment of variables of two corresponding variable groups. If there is a hard clause with all variables set by this assignment and this clause is unsatisfied by it, we do not add the edge uv to the graph.

Otherwise, the edge uv is added to the graph. The weight of this edge equals the number of soft clauses with one variable falling into the group of u and the other variable falling into the group of v and satisfied by the assignment given by (u, v).

We obtain a tripartite graph of size roughly $3 \cdot 2^{n/3}$ with weighted vertices and edges, and this graph is constructed in $O^*(2^{2n/3})$ time. It can then be shown that the optimal solution corresponds to the minimum-weight triangle of the constructed graph. Since weights of vertices and edges are polynomial in the number of soft clauses, such triangle can be found in $O^*(2^{\omega n/3})$ time using matrix multiplication. □

5.2 q-CNF and the number of variables

We now move to a more general case where the hard clause length is bounded by a parameter q.

Lemma 4. *When soft clauses contain only positive literals, Partial Min-SAT can be solved in $O^*((2 - \frac{1}{q})^n)$ time.*

Proof. To approach this case of Partial Min-SAT, we first refer to the Φ-Subset problem, which is the central problem of the work of Fomin et al (Fomin et al. 2019). Their results hold in the general model where a predefined mapping Φ constructs the instance from an encoding in the form of a binary string. However, in our case the encoding is straightforward, so we restate Φ-Subset in the following way. Here, an implicit set family $\mathcal{F}$ should be understood as defined by the algorithm that checks whether an arbitrary set belongs to $\mathcal{F}$, instead of listing the sets in the family explicitly in the input.

Φ-Subset parameterized by n

Input: An implicit subset family $\mathcal{F}$ over an n-element universe $\mathcal{U}$, such that $S \stackrel{?}{\in} \mathcal{F}$ can be checked in polynomial time for a given $S \subseteq \mathcal{U}$.

Question: Find any element of $\mathcal{F}$.

Another crucial problem in the work is the Φ-Extension problem.

Φ-Extension parameterized by k'

Input: $\mathcal{F}$ and $\mathcal{U}$ as in Φ-Subset; also a subset $S \subseteq \mathcal{U}$ and an integer k'.

Question: Find any $X \subseteq \mathcal{U}$ of size at most k' such that $S \cup X \in \mathcal{F}$.

The central result of (Fomin et al. 2019) is that a $q^{k'} \cdot n^{O(1)}$ algorithm for Φ-Extension for some family $\mathcal{F}$ yields a $(2 - \frac{1}{q})^n \cdot n^{O(1)}$ algorithm for Φ-Subset for the same family. Thus, our approach to Partial Min-SAT with all-positive soft clauses is to reduce it to Φ-Subset, where universe is a set of variables, and show a Φ-Extension algorithm running in $q^{k'} \cdot n^{O(1)}$ time.

Let (ϕ, k) be the given instance of Partial Min-SAT. We start the reduction to Φ-Subset by identifying the universe of Φ-Subset with the set of variables of ϕ. The set family $\mathcal{F}$ is then defined as follows. A set S of variables of ϕ is in $\mathcal{F}$ if and only if assigning all variables of S to one (and all other variables to zero) satisfies all hard clauses and at most k soft clauses. Clearly, $S \stackrel{?}{\in} \mathcal{F}$ can be checked in polynomial time. Moreover, solving Φ-Subset for $\mathcal{F}$ is equivalent to solving (ϕ, k).

To finish the proof, we describe the algorithm for Φ-Extension. Given a set S of variables that are assigned to one, we need to assign at most k' more variables to one so at most k soft clauses are satisfied while all hard clauses are satisfied. If the assignment of the variables of S already satisfies at least $k + 1$ soft clauses, then no X exists for Φ-Extension, since we cannot "unsatisfy" any soft clause by assigning more variables to one. The same holds if there is a hard clause with all negative literals that all evaluate to false because of S. In these two cases, the algorithm reports that no X exists and stops.

Otherwise, at most k soft clauses are satisfied. If all hard clauses are satisfied as well, the algorithm reports empty X and stops, as $S \in \mathcal{F}$ already. We consider that at least one hard clause is not satisfied by the assignment. In this case, a branching can always be performed. There are at most q literals in this clause with variables outside S. Since the clause is not satisfied, all these literals are positive (while literals with variables in S are negative). As negative literals can no more be satisfied, the only option to satisfy the clause is to satisfy at least one its positive literal. This clearly yields at most q branches, where in each branch we add a variable to S and decrease k' by one.

Clearly, the obtained algorithm for Φ-Extension runs in $q^{k'} \cdot (n + m)^{O(1)}$ time. Finally, the central result of Fomin et al. gives the running time $(2 - \frac{1}{q})^n \cdot (n + m)^{O(1)}$ for Φ-Subset, hence the algorithm for Partial Min-SAT with soft clauses containing only positive literals. □

We say that a variable is non-trivial (with respect to the set of clauses) if it appears both negatively and positively (in the clauses of this set). The following algorithm generalizes Lemma 4, as well as the best-known algorithm for Min-Ones q-SAT.

Theorem 5. *There is an algorithm for Partial Min-SAT with running time $O^*(2^{\min\{k,t_s\}} \cdot (2 - \frac{1}{q})^{n-\min\{k,t_s\}})$, where t_s is the number of non-trivial variables with respect to soft clauses.*

Proof. The approach is to reduce to the case of all-positive soft clauses using the two following rules. When neither is applicable, Lemma 4 is applied.

Reduction rule 1. If there is a variable that appears in soft clauses only as negative literals, revert all literals containing this variable in all clauses of ϕ.

Branching rule 1. If there is a variable that appears in soft clauses as both positive and negative literals, branch on this variable set to true or false.

It is trivial to see that the rules are safe. Note that Branching rule 1 increases the number of satisfied soft clauses by at least one in each branch, so the depth of the corresponding recursion tree is at most k. Since it also reduces the number of non-trivial variables with respect to soft clauses by at least one, the depth of this tree is also at most t_s. Observe that each application of Branching rule 1 also reduces the number of variables by at least one.

In each leaf of the recursion tree produced by the rules, we have an instance of Partial Min-SAT suitable for application of Lemma 4. For an instance that corresponds to a leaf of depth d, the running time of the algorithm

of Lemma 4 is at most $O^*((2-\frac{1}{q})^{n-d})$. The recursion tree is a complete binary tree, so the worst case is when all leaves have maximum depth $\min\{k, t_s\}$ and there are $2^{\min\{k,t_s\}}$ leaves in total. Thus, the running time bound of our algorithm is $O^*(2^{\min\{k,t_s\}} \cdot (2-\frac{1}{q})^{n-\min\{k,t_s\}})$. □

5.3 q-CNF and the number of clauses

The absense of non-trivial variables helps also in hard clauses.

Lemma 5. *When there are no non-trivial variables with respect to hard clauses, the* Partial Min-SAT *is solvable in* $O^*((2-\frac{1}{q})^s)$ *time.*

Proof. We say that a set of soft clauses is *non-blocking*, if no variable is non-trivial with respect to this set and assigning all literals in these clauses to false does not force any hard clause to be unsatisfied.

Claim 1. *If no variable is non-trivial with respect to hard clauses, then for any non-blocking set of soft clauses, there is an assignment satisfying all hard clauses and satisfying none of clauses in the set.*

Proof of Claim 1. To see this, assign all variables appearing in the non-blocking set so that each literal in this set evaluates to false. Since no variable is non-trivial with respect to the set, this assignment is always possible and unique. Moreover, each hard clause still contains an unassigned variable, and each such variable is not non-trivial. Assign all such unassigned variables in a way that literals of these variables in hard clauses all evaluate to true. It is left to assign each remaining variable an arbitrary value. ⌟

Now if we are given an instance (ϕ, k) of Partial Min-SAT where no variable is non-trivial w.r.t. hard clauses, it is enough to determine whether there exists a non-blocking set consisting of at least $s-k$ soft clauses. Clearly, such non-blocking set guarantees a solution for (ϕ, k), while a solution for (ϕ, k) straightforwardly yields a non-blocking set of unsatisfied soft clauses.

We focus on the problem of finding a non-blocking set for the given formula consisting of at least $s-k$ clauses. We formulate this as a Φ-Subset problem. The universe of Φ corresponds to the set of all soft clauses, while the desired subset family in Φ consists of all *complements* of non-blocking sets of soft clauses of size at most k.

Then the Φ-Extension problem can be reformulated as that we are given a set of soft clauses and we need to remove at most k clauses from it so it becomes non-blocking. This problem admits a clear q^k algorithm: if there is a hard clause that is forced to be unsatisfied, then there are at most q options of which literal should be "unblocked" by deletion of soft clauses containing it. This gives q branches, and in each branch k is decreased by at least one. If the set is blocking because some variable is non-trivial with respect to it, then branching on the value of this variable gives two branches where k is decreased.

The q^k-algorithm for Φ-Extension finally gives us the $O^*((2-\frac{1}{q})^s)$-running time algorithm for Φ-Subset. Consequently, the required algorithm for Partial Min-SAT is obtained. □

With addition of simple branching on non-trivial variables w.r.t. hard clauses gives us the following.

Corollary 2. *There is an algorithm for* Partial Min-SAT *running in* $O^*(2^{t_h} \cdot (2-\frac{1}{q})^s)$ *time, where* t_h *is the number of non-trivial variables with respect to hard clauses.*

Our final goal is to obtain a faster-than-2^m algorithm for Partial Min-SAT when length of hard clauses is bounded. We obtain this by combining simple branching rules with the algorithm above, while also using another novel technique to get rid of non-trivial variables when a suitable branching is not possible.

Theorem 6. *There is an algorithm for* Partial Min-SAT *running in time* $O^*(1.6181^h \cdot (2-\frac{1}{q})^s)$ *for* $q \geq 3$.

Proof. The algorithm starts with the following simple branching rule and simple reduction rule.

Branching rule 2. If there is a non-trivial (w.r.t. hard clauses) variable that appears at least three times (in total in ϕ), branch on the value of this variable.

Reduction rule 2. If Branching rule 2 cannot be applied and there are two non-trivial variables that appear in the same pair of hard clauses, assign these variables a value so both these clauses are satisfied.

Branching rule 2 leaves us with the case when each non-trivial variable appears exactly two times in hard clauses, while it does not appear in soft clauses at all. We still want to reduce this case to when no variable is non-trivial w.r.t. hard clauses. While there are 2^{t_h} possible assignments of non-trivial variables, none of them satisfies any soft clause. Thus, we are not interested in each assignment itself, but we are rather interested in the set of hard clauses it satisfies. This idea leaves us with the following branching rule.

Branching rule 3. If there is a hard clause containing $p \geq 1$ non-trivial variables, consider $p + 1$ branches:

- either this clause is not satisfied by any non-trivial variable; in this case assign all their literals in the clause to false;
- or this clause is satisfied by some of these variables; for each of p variables, consider a branchg where its literal is assigned to true while other $p - 1$ literals are assigned to false.

Claim 2. *Branching rule 3 is safe. It produces $p + 1$ branches. If Branching rule 2 and Reduction rule 2 cannot be applied, it reduces the number of hard clauses with a non-trivial variable by at least p in each branch, and in one branch the decrease is at least $p + 1$.*

Proof of Claim 2. To see that Branching rule 3 is safe, note that the first item in its definition corresponds to the case when the selected hard clause is not satisfied by non-trivial variables. In this case, all non-trivial variables should be assigned a determined value.

The second item corresponds to when at least one of the non-trivial variables satisfy the clause. In this case, we can always assume that *exactly* one such variable satisfy the clause. If at least two non-trivial variables satisfy the clause, we can change the value of any of them and only increase the number of satisfied hard clauses, as each non-trivial variable appears exactly two times among hard clauses. Note that soft clauses are not influenced by this change at all. Thus, a greedy strategy of picking exactly one non-trivial variable for satisfying the clause is valid.

To show the number of clauses reduced in each branch, consider the number of hard clauses sharing a non-trivial variable with the selected clause. Since Reduction rule 2 cannot be applied, each of them shares exactly one non-trivial variable with the clause. Hence, there are exactly p such clauses. When we choose to not satisfy the selected clause with non-trivial variables, we satisfy exactly all p of them. In other p branches, where we flip the value of one non-trivial variable, we satisfy all but one of them and the selected clause, so the number of newly-satisfied clauses in these branches equals p as well.

While we do not satisfy the selected clause in the first branch, in this branch all non-trivial variables of this clause are assigned a value, so the selected clause no longer contain non-trivial variables. This gives the additional $+1$ to reduced clauses in exactly one of the branches as required. ⌟

The algorithm applies Branching rule 2, Reduction rule 2 and Branching rule 3 exhaustively. When none of them can be applied, then, clearly, there are no non-trivial variables with respect to hard clauses. In this case, the algorithm uses the algorithm of Lemma 5 as a subroutine and solves the instance of the current branch.

Running time analysis. Branching rule 2 reduces the *total* number of clauses in the formula by at least one in each branch, while in one branch the number of reduced clauses is at least two. The worst branching vector for this rule is $(1, 2)$, and the derived branching factor is less than 1.6181.

Branching rule 3 gives a family of branching vectors, one for each $p \in [q]$, namely vectors $(1, 2)$, $(2, 2, 3)$, $(3, 3, 3, 4)$, …, $(q, q, \ldots, q, q+1)$. Note that the vector $(1, 2)$ can trivially be expanded (by applying itself into its branches) into a vector $(2, 3, \ldots, t-2, t-1, t-1, t)$ for arbitrary t. Hence, its branching factor is not better than the factors of other $q-1$ vectors. Then $(1, 2)$ again gives the worst branching factor that is bounded by 1.6181. However, only hard clauses are in concern of this branching rule. Note also that Branching rule 2 is never applied after Branching rule 3 was applied. So once a clause containing a non-trivial variable was reduced, it will be never considered again by these two rules.

Finally, the algorithm of Lemma 5 gives us a subroutine that depends exponentially on the number of soft clauses, and the exponent is $(2 - \frac{1}{q})$. Hence, when $q \geq 3$ the worst exponent under the number of hard clauses is 1.6181, while for soft clauses it equals $2 - \frac{1}{q}$. The upper bound of $1.6181^h \cdot (2 - \frac{1}{q})^s \cdot |\phi|^{O(1)}$ follows. □

5.4 Mixing n and m

In Theorem 1 we have shown that Partial Min-SAT is not solvable significantly faster than $O^*(2^n)$ or $O^*(2^m)$ assuming the SETH. One might start to suspect that even for all $\alpha \in [0, 1]$ there is no $\epsilon > 0$ such that Partial Min-SAT admits $O^*((2-\epsilon)^{\alpha n + (1-\alpha)m})$. The following theorem shows that this not the case. Moreover, when n is approximately equal to m the Partial Min-SAT problem can be solved faster than both $O^*(2^n)$ and $O^*(2^m)$.

Theorem 7. *There is an algorithm for Partial Min-SAT with running time* $O^*(1.755^{\frac{n+m}{2}})$.

Proof. We assume that an instance of Partial Min-SAT is given by a set of hard clauses $\mathcal{H}$ and a set of soft clauses $\mathcal{S}$.

Note that we can rename variables in such a way that for each variable the number of its positive literals in the formula is not smaller than the number of its negative literals in the formula.

First of all, we list some reduction rules whose application simplifies the problem.

Reduction rule 3. If $C = (x)$ is a hard clause containing only one variable, then assign x the value 1.

Reduction rule 4. If the literals of a variable x appear only in hard clauses, and all these literals are positive, then assign the value 1 to the variable x.

Reduction rule 5. If the literals of a variable x appear only in soft clauses and all these literals are positive, then assign the value 0 to the variable x.

The above-mentioned rules allow us to eliminate variables that appear exactly once among all clauses. In Max-SAT and SAT it is possible to eliminate in polynomial time variables that appear at most two times in the input formula. Unfortunately, this is not true for Partial Min-SAT, since Partial Min-SAT is NP-hard if each variable appears at most two times. However, we can eliminate a variable that appears once positively and once negatively, and at least one of its literals belongs to a hard clause.

Reduction rule 6. If a variable x occurs exactly twice, in hard clauses $(x \vee C_1)$ and $(\neg x \vee C_2)$, then we can replace these two clauses with one hard clause $(C_1 \vee C_2)$.

It is obvious that we can apply this rule in polynomial time. The rule is also correct since (i) if $(x \vee C_1)$, $(\neg x \vee C_2)$ are satisfied then $(C_1 \vee C_2)$ is satisfied by the same assignment as well; (ii) if $(C_1 \vee C_2)$ is satisfied by some assignment, then either assigning x to 0 or assigning x to 1 satisfies the clauses $(x \vee C_1)$, $(\neg x \vee C_2)$, and the rest of the formula is unchanged since x does not appear in other clauses.

Reduction rule 7. If some variable x appears once negatively in soft clauses, and all other literals of x are positive and are contained in hard clauses, then we can assign the value 1 to x.

Now we are ready to list branching rules that we employ. Let us call a variable that appears a times positively and b times negatively in the formula *a variable of type* (a, b).

Branching rule 4. Let x be a variable of type (a, b) such that either $a \geq 2, b \geq 1$ or $a \geq 4, b \geq 0$. We branch on the value of x, i.e. consider separately two subcases where $x = 1$ and where $x = 0$. In this situation we get a $(\frac{3}{2}, 1)$ or $(\frac{5}{2}, \frac{1}{2})$-branching, respectively.

Note that after exhaustive application of reduction rules 4, 5, 6, 7 and branching rule 4 all variables are of the following types $(3, 0), (2, 0), (1, 1)$. Moreover, any $(1, 1)$-variable appears only in soft clauses.

Branching rule 5. If $C_1, C_2 \in \mathcal{S}$ such that $x \in C_1, \neg x \in C_2$, the variable x does not appear in any other clause, and there is a positive literal $y \in C_1$ and $y \neq x$ then consider the following branching: (i) $x = 1$ and (ii) $x = y = 0$.

Claim 3. *Branching rule 5 is correct and gives a* $(\frac{3}{2}, 1)$*-branching.*

Proof of Claim 3. It is obvious that a branching on three cases (i) $x = 1$, (ii) $x = y = 0$, (iii) $x = 0, y = 1$ is correct as it exhausts all possibilities. Observe that if in third case we switch value x from 0 to 1 we do not falsify any hard clause (since no hard clauses contain x) and the number of satisfied soft clauses does not increase since the clause C_1 is satisfied by y. So we can simply omit the third branch.

In the branch $x = 1$ we eliminate the variable x as well as the clause C_1, so our measure $\frac{m+n}{2}$ decreases by 1. In the case $x = y = 0$ we eliminate the variables x, y and the clause C_2, so the measure decreases by $\frac{3}{2}$. Thus, we get a $(\frac{3}{2}, 1)$-branching. ⌟

At this point we can assume that any variable appears only positively, and in either two or three clauses.

Branching rule 6. If a soft clause C contains only one variable x then consider two cases: (i) $x = 0$ and (ii) $x = 1$.

Claim 4. *Branching rule 6 is correct and it leads to a* $(\frac{3}{2}, 1)$ *branching.*

Proof of Claim 4. The correctness of the branching rule is obvious as we exhaust all possible cases. In the branch $x = 1$ we satisfy at least 2 clauses and eliminate the variable x so the measure $\frac{m+n}{2}$ decreases by at least $\frac{3}{2}$. In the branch $x = 0$ we eliminate the variable x and the clause C since it become empty. Thus, the claim follows. ⌟

From now on we can assume that there is no soft clause containing only one literal as otherwise we apply branching rule 6.

Branching rule 7. Let x appear in exactly one soft clause of S that contains another variable y. Consider two branches $y = 1$ and $y = 0$.

Claim 5. *Branching rule 7 leads to a* $(\frac{5}{2}, \frac{1}{2})$*-branching.*

Proof of Claim 5. Assume that at least three different clauses contain variables x or y. In branch $y = 0$ we delete variable y so the measure decrease at least by $\frac{1}{2}$. In branch $y = 1$ we can assign x value 1 since the soft clause with x is already satisfied and it will not hurt us assign x value 1 as potentially it will satisfy some hard clauses. So in this branch we eliminate two variables and at least 3 clauses will be satisfied as at this moment all variables appear only positively.

If there are no three different clauses containing x or y then we have the following situation: there is a hard clause C_1 and a soft clause C_2 such that C_1, C_2 both contain variables x, y and probably some other variables, however, there is no occurrence of x, y in other clauses. In this situation there is no need to assign x a value different from y. So in the case $y = 1$ we satisfy 2 clauses and remove variables x, y and in the case $y = 0$ we also eliminate variables x, y. This leads to at least $(2, 1)$-branching which is superior than $(\frac{5}{2}, \frac{1}{2})$-branching. ⌟

At this point our instance of Partial Min-SAT is trivial (either there is an empty hard clause or there are no clauses at all), or all variables appear exactly three times and exactly once in hard clauses and twice in soft clauses. So without loss of generality we can assume that for some variable x there are soft clauses $(x \vee C_1)$, $(x \vee C_2)$ and a hard clause $(x \vee y \vee D)$ where the variable y occurs in $(x \vee y \vee D)$ and two soft clauses. The hard clause containing x has at least one more variable due to Reduction rule 3.

Branching rule 8. If $(x \vee y \vee D)$ is a hard clause and except this clause variables x, y appears only positively and in soft clauses then consider two cases: (i) $x = 0$, (ii) $x = 1, y = 0$.

Claim 6. *Branching rule 8 is correct and leads to at least* $(\frac{5}{2}, \frac{1}{2})$*-branching.*

Proof of Claim 6. In order to show correctness it is enough to show that the assignment $x = 1, y = 0$ is at least as good as the assignment $x = 1, y = 1$. Note that if we put $x = 1$ then all hard clauses containing the literal y will be satisfied and literal $\neg y$ does not appear in the formula so it is beneficial to assign $y = 0$ trying to avoid satisfying soft clauses.

In branch $x = 0$ we decrease the measure by 1 eliminating the variable x. In branch $x = 1, y = 0$ we eliminate three clauses containing the literal x, as well as the variables x, y, hence decreasing the measure by $\frac{5}{2}$. ⌟

During the algorithm we only apply reduction rules that take polynomial time to perform or branching rules with branching vectors at least $(\frac{5}{2}, \frac{1}{2}) = (\frac{3}{2}, 1)$, hence, the running time of the whole algorithm is at most $O^*(1.755^{\frac{n+m}{2}})$.

□

Acknowledgments

This work was supported by the Ministry of Economic Development of the Russian Federation (IGK 000000C313925P4C0002), agreement №139-15-2025-010.

References

A. Abramé and D. Habet. 2015. "Local search algorithm for the partial minimum satisfiability problem." In: *2015 IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 821–827.

G. Agrawal and S. Maity. Sept. 2021. "The Small Set Vertex Expansion Problem." *Theoretical Computer Science*, 886, (Sept. 2021), 84–93. doi:10.1016/j.tcs.2021.07.017.

A. Avidor and U. Zwick. 2005. "Approximating min 2-sat and min 3-sat." *Theory of Computing Systems*, 38, 3, 329–345.

N. Bansal and V. Raman. 1999. "Upper bounds for MaxSat: Further improved." In: *International symposium on algorithms and computation*. Springer, 247–258.

J. Chen and I. A. Kanj. 2004. "Improved exact algorithms for Max-Sat." *Discrete Applied Mathematics*, 142, 1-3, 17–27.

H. Chu, M. Xiao, and Z. Zhang. 2021. "An improved upper bound for SAT." *Theoretical Computer Science*, 887, 51–62.

M. Cygan, H. Dell, D. Lokshtanov, D. Marx, J. Nederlof, Y. Okamoto, R. Paturi, S. Saurabh, and M. Wahlström. June 2016. "On Problems as Hard as CNF-SAT." *ACM Transactions on Algorithms*, 12, 3, (June 2016), 1–24. doi:10.1145/2925416.

M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. 2015. *Parameterized Algorithms*. Springer. ISBN: 978-3-319-21274-6. doi:10.1007/978-3-319-21275-3.

B. Escoffier and V. T. Paschos. 2007. "Differential approximation of min sat, max sat and related problems." *European Journal of Operational Research*, 181, 2, 620–633.

F. V. Fomin, S. Gaspers, D. Lokshtanov, and S. Saurabh. Apr. 2019. "Exact Algorithms via Monotone Local Search." *Journal of the ACM*, 66, 2, (Apr. 2019), 1–23. doi:10.1145/3284176.

F. Hers, A. Morgado, J. Planes, and J. Marques-Silva. 2012. "Iterative SAT solving for minimum satisfiability." In: *2012 IEEE 24th International Conference on Tools with Artificial Intelligence*. Vol. 1. IEEE, 922–927.

E. A. Hirsch. 1998. "Two New Upper Bounds for SAT." In: *SODA*. Citeseer, 521–530.

A. Ignatiev, A. Morgado, and J. Marques-Silva. 2014. "On reducing maximum independent set to minimum satisfiability." In: *International Conference on Theory and Applications of Satisfiability Testing*. Springer, 103–120.

A. Ignatiev, A. Morgado, J. Planes, and J. Marques-Silva. 2016. "Maximal falsifiability." *AI Communications*, 29, 2, 351–370.

R. Kohli, R. Krishnamurti, and P. Mirchandani. 1994. "The minimum satisfiability problem." *SIAM Journal on Discrete Mathematics*, 7, 2, 275–283.

A. Kügel. 2012. "Natural Max-SAT encoding of Min-SAT." In: *International Conference on Learning and Intelligent Optimization*. Springer, 431–436.

C. M. Li and F. Manya. 2021. "MaxSAT, hard and soft constraints." In: *Handbook of satisfiability*. IOS Press, 903–927.

C. M. Li, Z. Zhu, F. Manya, and L. Simon. 2011. "Minimum satisfiability and its applications." In: *IJCAI*, 605–610.

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

C.-M. Li and F. Manya. 2015. "An exact inference scheme for MinSAT." In: *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

M. V. Marathe and S. Ravi. 1996. "On approximation algorithms for the minimum satisfiability problem." *Information Processing Letters*, 58, 1, 23–29.

N. Misra, N. Narayanaswamy, V. Raman, and B. S. Shankar. 2013. "Solving min ones 2-sat as fast as vertex cover." *Theoretical Computer Science*, 506, 115–121. doi:https://doi.org/10.1016/j.tcs.2013.07.019.

B. Monien and E. Speckenmeyer. 1985. "Solving satisfiability in less than 2n steps." *Discrete Applied Mathematics*, 10, 3, 287–295.

R. Niedermeier and P. Rossmanith. 1999. "New upper bounds for MaxSat." In: *International Colloquium on Automata, Languages, and Programming*. Springer, 575–584.

R. R. Williams. 2007. "Algorithms and Resource Requirements for Fundamental Problems." Ph.D. Dissertation. USA. ISBN: 9780549164777. AAI3274191.

M. Xiao. July 2022. "An Exact MaxSAT Algorithm: Further Observations and Further Improvements." In: *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*. Ed. by L. D. Raedt. Main Track. International Joint Conferences on Artificial Intelligence Organization, (July 2022), 1887–1893. doi:10.24963/ijcai.2022/262.

M. Xiao and H. Nagamochi. 2017. "Exact algorithms for maximum independent set." *Information and Computation*, 255, 126–146. doi:https://doi.org/10.1016/j.ic.2017.06.001.

C. Xu, W. Li, Y. Yang, J. Chen, and J. Wang. 2019. "Resolution and domination: an improved exact MaxSAT algorithm." In: *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. AAAI Press, 1191–1197.

M. Yamamoto. 2005. "An Improved $O^*(1.234^m)$-Time Deterministic Algorithm for SAT." In: *International Symposium on Algorithms and Computation*. Springer, 644–653.

Z. Zhu, C.-M. Li, F. Manyà, and J. Argelich. 2012. "A new encoding from MinSAT into MaxSAT." In: *International Conference on Principles and Practice of Constraint Programming*. Springer, 455–463.

Received 02 June 2025; accepted 22 January 2026