Fine-grained Complexity of Partial Minimum Satisfiability

Ivan Bliznets[1,2], **Danil Sagunov**[2] and **Kirill Simonov**[3]
[1]HSE University, St.Petersburg, Russia
[2]St. Petersburg Department of Steklov Mathematical Institute of Russian Academy of Sciences, St.Petersburg, Russia
[3]Algorithms and Complexity Group, TU Wien, Austria
{iabliznets, danilka.pro, kirillsimonov}@gmail.com

Abstract

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 AAAI 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 $\mathcal{O}^*((2-\epsilon)^m)$ 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.

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 AAAI-2011, Li et al. [2011] proposed an alternative to this approach and suggested reducing NP-hard problems to the following PARTIAL MIN-SAT problem.

[1]http://satisfiability.org/SAT22/
[2]https://maxsat-evaluations.github.io/2021/

Running time	References
$\mathcal{O}^*(1.260^m)$	[Monien and Speckenmeyer, 1985]
$\mathcal{O}^*(1.239^m)$	[Hirsch, 1998]
$\mathcal{O}^*(1.234^m)$	[Yamamoto, 2005]
$\mathcal{O}^*(1.2226^m)$	[Chu *et al.*, 2021]

Table 1: Progress for SAT in terms of m (total number of clauses). $\mathcal{O}^*$ omits factors polynomial in n and m.

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. [2011] developed the MinSatz solver and showed that reducing to PARTIAL MIN-SAT and solving with MinSatz is in some cases more 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, many results relating to PARTIAL MIN-SAT were published [Ignatiev *et al.*, 2016; Zhu *et al.*, 2012; Li and Manya, 2015; Li *et al.*, 2012; Abramé and Habet, 2015; Hers *et al.*, 2012; Ignatiev *et al.*, 2014; Avidor and Zwick, 2005; Escoffier and Paschos, 2007]. Note that some approximation results were known for PARTIAL MIN-SAT even before [Li *et al.*, 2011], for example [Marathe and Ravi, 1996; Kohli *et al.*, 1994]. 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 Tables 1 and 2 we list the algorithms whose running time depends on the overall number of clauses in the input formula. Note that the last results were presented quite recently at AAAI-2021 and IJCAI-2019 after 15 years of no progress.

While the search for the best worst-case guarantee of an algorithm for SAT and its variants poses an intriguing the-

Running time	References
$\mathcal{O}^*(1.3803^m)$	[Niedermeier and Rossmanith, 1999]
$\mathcal{O}^*(1.3412^m)$	[Bansal and Raman, 1999]
$\mathcal{O}^*(1.3248^m)$	[Chen and Kanj, 2004]
$\mathcal{O}^*(1.2989^m)$	[Xu *et al.*, 2019]

Table 2: Progress for MAX-SAT in terms of m.

oretical question, it is also well-motivated from the practical perspective. 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.

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 $\mathcal{O}^*((2-\epsilon)^m)$ 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 $\mathcal{O}^*((2-\epsilon)^n)$ 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. ($\mathcal{O}^*(2^n)$ and $\mathcal{O}^*(2^m)$ 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 $\mathcal{O}^*(2^{\omega n/3})$ time (here ω is the matrix multiplication exponent). If $q \leq 2$ then PARTIAL MIN-SAT can be solved in time $\mathcal{O}^*(c_{vc}^s)$ where c_{vc} is the lowest constant such that VERTEX COVER admit $\mathcal{O}^*(c_{vc}^{n'})$ 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 $\mathcal{O}^*(2^{\min\{n,k,t_s\}} \cdot (2-\frac{1}{q})^{n-\min\{n,k,t_s\}})$ time and if $q \geq 3$ then there is a $\mathcal{O}^*(1.6181^h \cdot (2-\frac{1}{q})^s)$ running time algorithm for PARTIAL MIN-SAT. Finally for the general case we present an algorithm with the running time $\mathcal{O}^*(1.755^{\frac{m+n}{2}})$.

The rest of the paper is organized as follows. In Section 2 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 3 we show Theorem 1 and other parameterized complexity lower bounds. Finally, in Section 4 we present our algorithmic results. Proofs of the statements marked by $\star$ are omitted due to space constraints.

Throughout the paper we use standard notions and definitions from graph theory [Diestel, 2010], parameterized complexity [Cygan *et al.*, 2015], exact exponential algorithms [Fomin and Kratsch, 2010], and satisfiability theory [Marek, 2009]. Due to space constraints, we are not able to formally present all the definitions in the main body; we refer the reader the standard textbooks cited above.

2 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 $|\bigcup S_i| \leq \ell$?

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 a literal x from a 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 a literal z from a 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 ($\star$)**.** PARTIAL MIN-SAT *parameterized by* $k + h$ *admits a parameterized reduction to* MULTICOLORED MINIMUM UNION *parameterized by* $\ell + k$.

3 Lower Bounds

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 *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 $\mathcal{O}(2^{\delta n})$ running time.*

Theorem 1. *Unless SETH fails,* PARTIAL MIN-SAT *cannot be solved in $\mathcal{O}^*((2-\epsilon)^n)$ or $\mathcal{O}^*((2-\epsilon)^m)$ 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. We then will show how from this reduction and a $\mathcal{O}((2-\epsilon)^m)$ algorithm for PARTIAL MIN-SAT it follows that SETH fails through Proposition 1. Observe also that any $\mathcal{O}^*((2-\epsilon)^m)$-time algorithm has a $\mathcal{O}((2-\epsilon')^m)$ 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 $n^{\mathcal{O}(p)}$. 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 in the instance of PARTIAL MIN-SAT is bounded by $(1 + (c+1)/p)n$.

Assume that the SETH holds and there exists a $\mathcal{O}(2^{\delta' m'})$-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 $\mathcal{O}(2^{\delta n})$ 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 $\mathcal{O}(2^{\delta' m'})$-

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 $\mathcal{O}(2^{(\delta'+(1-\delta')/2)n})$, which is $\mathcal{O}(2^{\delta n})$, and this contradicts Proposition 1. The proof is complete. □

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 $\mathcal{O}^*(2^m) = \mathcal{O}^*(2^h)$ 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 $f(h + k) \cdot n^{\mathcal{O}(1)}$ 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*.*

4 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. 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 $\mathcal{O}^*(c^{n'})$, where n' is the number of vertices in the graph, immediately gives an algorithm for MIN-SAT with running time $\mathcal{O}^*(c^m)$. 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.

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* $\mathcal{O}^*(1.1996^s)$.

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* $\mathcal{O}^*(2^{\omega n/3})$ *running-time algorithm for* PARTIAL MIN-SAT *restricted to formulas in* 2*-CNF.*

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

Lemma 3. *When soft clauses contain only positive literals,* PARTIAL MIN-SAT *can be solved in* $\mathcal{O}^*((2 - \frac{1}{q})^n)$ *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 [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 $q^{k'} \cdot n^{\mathcal{O}(1)}$ algorithm for Φ-EXTENSION for some family $\mathcal{F}$ yields a $(2-\frac{1}{q})^n \cdot n^{\mathcal{O}(1)}$ 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 $q^{k'} \cdot n^{\mathcal{O}(1)}$ 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 $q^{k'} \cdot (n+m)^{\mathcal{O}(1)}$ time. Finally, the central result of Fomin et al. gives the running time $(2-\frac{1}{q})^n \cdot (n+m)^{\mathcal{O}(1)}$ for Φ-EXTENSION, 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 3, 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* $\mathcal{O}^*(2^{\min\{k,t_s\}} \cdot (2-\frac{1}{q})^{n-\min\{k,t_s\}})$*, 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 3 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 3. For an instance that corresponds to a leaf of depth d, the running time of the algorithm of Lemma 3 is at most $\mathcal{O}^*((2-\frac{1}{q})^{n-d})$. 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 $\mathcal{O}^*(2^{\min\{k,t_s\}} \cdot (2-\frac{1}{q})^{n-\min\{k,t_s\}})$. □

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

Lemma 4 ($\star$)**.** *When there are no non-trivial variables with respect to hard clauses, the* PARTIAL MIN-SAT *is solvable in* $\mathcal{O}^*((2-\frac{1}{q})^s)$ *time.*

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* $\mathcal{O}^*(2^{t_h} \cdot (2-\frac{1}{q})^s)$ *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* $\mathcal{O}^*(1.6181^h \cdot (2-\frac{1}{q})^s)$ *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 1. *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. 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 4 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 4 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 $1.6181^h \cdot (2 - \frac{1}{q})^s \cdot |\phi|^{\mathcal{O}(1)}$ follows. □

In Theorem 1 we have shown that PARTIAL MIN-SAT is not solvable significantly faster than $\mathcal{O}^*(2^n)$ or $\mathcal{O}^*(2^m)$ 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 $\mathcal{O}^*((2-\epsilon)^{\alpha n + (1-\alpha)m})$. 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 $\mathcal{O}^*(2^n)$ and $\mathcal{O}^*(2^m)$.

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

Acknowledgements

Research presented in Section "Algorithms" is supported by RSCF grant 18-71-10042. Research presented in Section "Minimum Union" is supported by HSE University and Leonhard Euler International Mathematical Institute in Saint Petersburg (agreement no. 075-15-2019-1620). Research presented in Section "Lower Bounds" is supported by the Austrian Science Fund (FWF) via project Y1329 (Parameterized Analysis in Artificial Intelligence).

References

[Abramé and Habet, 2015] André Abramé and Djamal Habet. Local search algorithm for the partial minimum satisfiability problem. In *2015 IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 821–827. IEEE, 2015.

[Agrawal and Maity, 2021] Garima Agrawal and Soumen Maity. The small set vertex expansion problem. *Theoretical Computer Science*, 886:84–93, September 2021.

[Avidor and Zwick, 2005] Adi Avidor and Uri Zwick. Approximating min 2-sat and min 3-sat. *Theory of Computing Systems*, 38(3):329–345, 2005.

[Bansal and Raman, 1999] Nikhil Bansal and Venkatesh Raman. Upper bounds for maxsat: Further improved. In *International symposium on algorithms and computation*, pages 247–258. Springer, 1999.

[Chen and Kanj, 2004] Jianer Chen and Iyad A Kanj. Improved exact algorithms for max-sat. *Discrete Applied Mathematics*, 142(1-3):17–27, 2004.

[Chu *et al.*, 2021] Huairui Chu, Mingyu Xiao, and Zhe Zhang. An improved upper bound for sat. *Theoretical Computer Science*, 887:51–62, 2021.

[Cygan *et al.*, 2015] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

[Cygan *et al.*, 2016] Marek Cygan, Holger Dell, Daniel Lokshtanov, Dániel Marx, Jesper Nederlof, Yoshio Okamoto, Ramamohan Paturi, Saket Saurabh, and Magnus Wahlström. On problems as hard as CNF-SAT. *ACM Transactions on Algorithms*, 12(3):1–24, June 2016.

[Diestel, 2010] R. Diestel. *Graph Theory*. Graduate Texts in Mathematics. Springer Berlin Heidelberg, 2010.

[Escoffier and Paschos, 2007] Bruno Escoffier and Vangelis Th Paschos. Differential approximation of min sat, max sat and related problems. *European Journal of Operational Research*, 181(2):620–633, 2007.

[Fomin and Kratsch, 2010] Fedor V Fomin and Dieter Kratsch. *Exact exponential algorithms*. Springer-Verlag Berlin Heidelberg, 2010.

[Fomin *et al.*, 2019] Fedor V. Fomin, Serge Gaspers, Daniel Lokshtanov, and Saket Saurabh. Exact algorithms via monotone local search. *Journal of the ACM*, 66(2):1–23, April 2019.

[Hers *et al.*, 2012] Federico Hers, Antonio Morgado, Jordi Planes, and Joao Marques-Silva. Iterative sat solving for minimum satisfiability. In *2012 IEEE 24th International Conference on Tools with Artificial Intelligence*, volume 1, pages 922–927. IEEE, 2012.

[Hirsch, 1998] Edward A Hirsch. Two new upper bounds for sat. In *SODA*, pages 521–530. Citeseer, 1998.

[Ignatiev *et al.*, 2014] Alexey Ignatiev, Antonio Morgado, and Joao Marques-Silva. On reducing maximum independent set to minimum satisfiability. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 103–120. Springer, 2014.

[Ignatiev *et al.*, 2016] Alexey Ignatiev, António Morgado, Jordi Planes, and Joao Marques-Silva. Maximal falsifiability. *AI Communications*, 29(2):351–370, 2016.

[Kohli *et al.*, 1994] Rajeev Kohli, Ramesh Krishnamurti, and Prakash Mirchandani. The minimum satisfiability problem. *SIAM Journal on Discrete Mathematics*, 7(2):275–283, 1994.

[Kügel, 2012] Adrian Kügel. Natural max-sat encoding of min-sat. In *International Conference on Learning and Intelligent Optimization*, pages 431–436. Springer, 2012.

[Li and Manya, 2015] Chu-Min Li and Felip Manya. An exact inference scheme for minsat. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[Li *et al.*, 2011] Chu-Min Li, Zhu Zhu, Felip Manya, and Laurent Simon. Minimum satisfiability and its applications. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[Li *et al.*, 2012] Chu Min Li, Zhu Zhu, Felip Manyà, and Laurent Simon. Optimizing with minimum satisfiability. *Artificial Intelligence*, 190:32–44, 2012.

[Marathe and Ravi, 1996] Madhav V Marathe and SS Ravi. On approximation algorithms for the minimum satisfiability problem. *Information Processing Letters*, 58(1):23–29, 1996.

[Marek, 2009] Victor W Marek. *Introduction to mathematics of satisfiability*. CRC Press, 2009.

[Misra *et al.*, 2013] Neeldhara Misra, N.S. Narayanaswamy, Venkatesh Raman, and Bal Sri Shankar. Solving min ones 2-sat as fast as vertex cover. *Theoretical Computer Science*, 506:115–121, 2013.

[Monien and Speckenmeyer, 1985] Burkhard Monien and Ewald Speckenmeyer. Solving satisfiability in less than 2n steps. *Discrete Applied Mathematics*, 10(3):287–295, 1985.

[Niedermeier and Rossmanith, 1999] Rolf Niedermeier and Peter Rossmanith. New upper bounds for maxsat. In *International Colloquium on Automata, Languages, and Programming*, pages 575–584. Springer, 1999.

[Williams, 2007] R. Ryan Williams. *Algorithms and Resource Requirements for Fundamental Problems*. PhD thesis, USA, 2007. AAI3274191.

[Xiao and Nagamochi, 2017] Mingyu Xiao and Hiroshi Nagamochi. Exact algorithms for maximum independent set. *Information and Computation*, 255:126–146, 2017.

[Xu *et al.*, 2019] Chao Xu, Wenjun Li, Yongjie Yang, Jianer Chen, and Jianxin Wang. Resolution and domination: an improved exact maxsat algorithm. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 1191–1197. AAAI Press, 2019.

[Yamamoto, 2005] Masaki Yamamoto. An improved $o^*(1.234^m)$-time deterministic algorithm for sat. In *International Symposium on Algorithms and Computation*, pages 644–653. Springer, 2005.

[Zhu *et al.*, 2012] Zhu Zhu, Chu-Min Li, Felip Manyà, and Josep Argelich. A new encoding from minsat into maxsat. In *International Conference on Principles and Practice of Constraint Programming*, pages 455–463. Springer, 2012.