Revisiting Maximum Satisfiability and Related Problems in the Streaming Setting

Hoa T. Vu ✉
San Diego State University, San Diego, CA, USA

Abstract

We revisit the maximum satisfiability problem (`Max-SAT`) in the streaming setting. In this problem, we are given m clauses that are disjunctions of literals drawn from n Boolean variables. The objective is to find an assignment to the variables that maximizes the number of satisfied clauses. Chou et al. (FOCS 2020) showed that $\Omega(\sqrt{n})$ space is necessary to yield a $\sqrt{2}/2 + \epsilon$ approximation of the optimum value of `Max-`k`-SAT` for $k \geq 2$; they also presented an algorithm that yields a $\sqrt{2}/2 - \epsilon$ approximation of the optimum value of `Max-SAT` using $O(\log n/\epsilon^2)$ space.

In this paper, we focus not only on approximating the optimum value, but also on obtaining the corresponding Boolean assignment while using sublinear $o(mn)$ space. We present randomized single-pass algorithms that w.h.p [1] yield:

- A $1 - \epsilon$ approximation using $\tilde{O}(n/\epsilon^3)$ space [2] and exponential post-processing time;
- A $3/4 - \epsilon$ approximation using $\tilde{O}(n/\epsilon^2)$ space and polynomial post-processing time. If there are no duplicate clauses, the space can be further improved to $\tilde{O}(n/\epsilon)$.

Our ideas also extend to dynamic streams. On the other hand, we show that the streaming k`-SAT` problem that asks to decide whether one can satisfy all size-k input clauses must use $\Omega(n^k)$ space.

We also consider the related minimum satisfiability problem (`Min-SAT`), introduced by Kohli et al. (SIAM J. Discrete Math. 1994), that asks to find an assignment that minimizes the number of satisfied clauses. For this problem, we give a $\tilde{O}(n^2/\epsilon^2)$ space algorithm, which is sublinear when $m = \omega(n)$, that yields an $\alpha + \epsilon$ approximation where α is the approximation guarantee of the offline algorithm. If each variable appears in at most f clauses, we show that a $2\sqrt{fn}$ approximation using $\tilde{O}(n)$ space is possible.

Finally, for the `Max-AND-SAT` problem where clauses are conjunctions of literals, we show that any single-pass algorithm that approximates the optimum up to a factor better than $1/2$ with success probability at least $2/3$ must use $\Omega(mn)$ space.

2012 ACM Subject Classification Theory of computation → Sketching and sampling; Theory of computation → Approximation algorithms analysis

Keywords and phrases data streams, algorithms, maximum satisfiability, minimum satisfiability, satisfiability

[1] W.h.p. denotes "with high probability". Here, we consider $1 - 1/\operatorname{poly}(n)$ or $1 - 1/\operatorname{poly}(m)$ as high probability.

[2] $\tilde{O}$ hides polylog factors.

1 Introduction

Problems overview. The Boolean satisfiability problem (SAT) is one of the most famous problems in computer science. A satisfiability instance is a conjunction of m clauses $C_1, C_2, \ldots, C_m$ where each clause C_j is a disjunction of literals drawn from a set of n Boolean variables $x_1, \ldots, x_n$ (a literal is either a variable or its negation). Deciding whether such an expression is satisfiable is NP-Complete [15, 40]. When each clause has size exactly k, this is known as the k-SAT problem.

In the optimization version, one aims to find an assignment to the variables that maximizes the number of satisfied clauses. This is known as the maximum satisfiability (Max-SAT) problem. This problem is still NP-Hard even when each clause has at most two literals [17]. However, Max-SAT can be approximated up to a factor 3/4 using linear programming (LP) [18], network flow [43], or a careful greedy approach [38]. In polynomial time, one can also obtain an approximation slightly better than 3/4 using semidefinite programming (SDP) [7, 19]. Hastad showed the inapproximability result that unless P = NP, there is no polynomial-time algorithm that yields an approximation better than 21/22 to Max-2-SAT [21].

A related problem is the minimum satisfiability problem (Min-SAT) which was introduced by Kohli et al. [28]. In this problem, the goal is to minimize the number of satisfied clauses. They showed that this problem is NP-Hard and gave a simple randomized 2-approximation. Marathe and Ravi [33] showed that Min-SAT is equivalent to the minimum vertex cover problem and therefore an approximation factor better than 2 in polynomial time is unlikely. Better approximations for Min-k-SAT, for small values of k, have also been developed by Avidor and Zwick [8], Bertsimas et al. [11], and Arif et al. [5] using linear and semidefinite programming.

In this paper, we also consider another related optimization problem Max-AND-SAT. This problem is similar to Max-SAT except that each clause is a conjunction of literals (as opposed to being a disjunction of literals in Max-SAT). Trevisan studied this problem in the guise of parallel algorithms [41].

In this work, we aim to understand the space complexity of Max-SAT, Min-SAT, k-SAT, and Max-AND-SAT in the streaming model. In this setting, clauses are presented one by one in the stream in an online fashion and the objective is to use sublinear space $o(mn)$ while obtaining a guaranteed non-trivial approximation.

Motivation and past work. Constraint satisfaction problems and their optimization counterparts have recently received notable attention in the data stream model. Some examples include vertex coloring [6, 10], Max-2-AND [20], Max-Boolean-CSPs and Max-k-SAT [13, 14], Min-Ones d-SAT [2], and Max-2-XOR [27].

In terms of applications, SAT, Max-SAT, and Min-SAT have been used in model-checking, software package management, design debugging, AI planning, bioinformatics, combinatorial auctions etc. [3, 4, 12, 16, 22, 23, 31, 32, 34, 35, 39]. Many of these applications have inputs that are too large to fit in a single machine's main memory. Examples of large Max-SAT benchmarks that arise from real-world applications can be found at [1]. This motivates us to study this problem in the streaming setting that aims to use sublinear memory.

Max-SAT and Max-AND-SAT were also studied by Trevisan in [41] in the guise of parallel algorithms. Trevisan showed that there is a parallel algorithm that finds a $3/4 - \epsilon$ approximation to Max-SAT in $O(\text{poly}(1/\epsilon, \log m))$ time using $O(n + m)$ processors [41]. Our results here show that it suffices to use $O(n)$ processors.

The most relevant result is by Chou et al. [14]. They showed that $\Omega(\sqrt{n})$ space is required

to yield a $\sqrt{2}/2 + \epsilon$ approximation of the optimum value of `Max-`k`-SAT` for $k \geq 2$; they also presented an algorithm that yields a $\sqrt{2}/2 - \epsilon$ approximation of the optimum value of `Max-SAT` using $O(\log n/\epsilon^2)$ space.

In many cases, we want to not only approximate the optimum value, but also output the corresponding Boolean assignment which is an objective of this work. It is worth noting that storing the assignment itself requires $\Omega(n)$ space. While our algorithms use more space, this allows us to actually output the assignment and to obtain a better approximation.

To the best of our knowledge, unlike `Max-SAT`, somewhat surprisingly, there is no prior work on `SAT`, `Min-SAT`, and `Max-AND-SAT` in the streaming model.

Main results. Hereinafter, the memory use is measured in terms of bits. All randomized algorithms succeed w.h.p. For `Max-SAT`, we show that it is possible to obtain a non-trivial approximation using only $\tilde{O}(n)$ space which is the space needed to store the output assignment. Throughout this paper, algorithms actually output an assignment to the variables along with an estimate for the number of satisfied clauses. In this paper, we solely focus on algorithms that use a single pass over the stream. Furthermore, unless stated otherwise, we assume insertion-only streams.

The algorithms for `Max-SAT` rely on two simple observations. If $m = \omega(n/\epsilon^2)$ and we sample $\Theta(n/\epsilon^2)$ clauses uniformly at random, then w.h.p. an α approximation on the sampled clauses corresponds to an $\alpha - \epsilon$ approximation on the original input. Moreover, if a clause is large, it can be satisfied w.h.p. as long as each literal is set to `true` independently with a not-too-small probability and therefore we may ignore such clauses. This second observation also allows us to extend our result to insertion-deletion (dynamic) streams..

Based on the above observations, we proceed by simply ignoring large clauses. Then, among the remaining (small) clauses, we sample $\Theta(n/\epsilon^2)$ clauses uniformly at random, denoted by W. Finally, we run some randomized α approximation algorithm on W in post-processing in which every literal is set to `true` with some small probability. This will lead to an $\alpha - \epsilon$ approximation on the original set of clauses w.h.p.

There is a subtlety regarding duplicate clauses, especially for dynamic streams. Suppose two (or more) similar clauses appear in the stream, would we consider them as one clause or two separate clauses (or equivalently, one clause with weight 2)? This boils down to the choice of using an L_0 sampler or an L_1 sampler. Specifically, an L_0 sampler samples clauses whose net frequency is non-zero uniformly at random. On the other hand, an L_1 sampler (and Reservoir sampler in insertion-only streams) samples clauses proportionally to their net frequency. In this paper, we will use Reservoir sampling for insertion-only streams and L_0 sampling for dynamic streams. Unlike the L_0 sampler, the L_1 sampler is known to be imperfect [25] which results in some small space or additive approximation error overhead. To facilitate our discussion, we assume that there is no duplicate in dynamic streams.

▶ **Theorem 1.** *We have the following randomized streaming algorithms for* `Max-SAT`*.*

1. *A $3/4 - \epsilon$ and a $1 - \epsilon$ approximations for insertion-only streams while using $\tilde{O}(n/\epsilon^2)$ and $\tilde{O}(n/\epsilon^3)$ space respectively. These algorithms have $O(1)$ update time. If there is no duplicate, the space of the $3/4 - \epsilon$ approximation can be further reduced to $\tilde{O}(n/\epsilon)$.*
2. *A $3/4 - \epsilon$ and a $1 - \epsilon$ approximations for no-duplicate dynamic streams while using $\tilde{O}(n/\epsilon)$ and $\tilde{O}(n/\epsilon^4)$ space respectively. The update time can be made $\tilde{O}(1)$ with an additional $\log n/\epsilon$ factor in the space use.*

The decision problem `SAT` is however much harder in terms of streaming space complexity. Specifically, we show that $o(m)$ space is generally not possible. Our lower bound holds even for the decision k`-SAT` problem where each clause has exactly k literals and $m = \Theta((n/k)^k)$.

▶ **Theorem 2.** *Suppose $k \leq n/e$. Any one-pass streaming algorithm that solves k-SAT with success probability at least 3/4 requires $\Omega(m)$ space. This lower bound operates when $k \leq \log(m+1)$ and $m = \Theta((n/k)^k)$.*

This lower bound for k-SAT is tight up to polylogarithmic factors in the following sense. If $k > \log m$, we know that it is possible to satisfy all the input clauses via a probabilistic argument. In particular, we can independently assign each variable to `true` or `false` equiprobably then the probability that a clause is not satisfied is smaller than $1/m$. Hence, a union bound over m clauses implies that the probability that we satisfy all the clauses is positive. On the other hand, if $k < \log m$, storing the entire stream requires $\tilde{O}(m)$ space. Hence, we have an $\tilde{O}(m)$-space algorithm.

For Min-SAT, we observe that one can obtain a $1+\epsilon$ approximation in $\tilde{O}(n^2)$ space using a combination of F_0 sketch and brute-forte search. However, it is not clear how to run polynomial time algorithms based on the sketch (see Section 3 for further discussion). We provide another approach that sidesteps this entirely.

▶ **Theorem 3.** *Suppose there is an α offline approximation algorithm for* Min-SAT *that runs in T time. Then, we have a randomized single-pass, $\tilde{O}(n^2/\epsilon^2)$-space streaming algorithm for* Min-SAT *that yields an $\alpha+\epsilon$ approximation and uses T post-processing time.*

Other results. For Min-SAT, if each variable appears in at most f clauses, then it is possible to yield a $2\sqrt{nf}$ approximation to Min-SAT in $\tilde{O}(n)$ space. On the lower bound side, we show that any streaming algorithm for Min-SAT that decides if OPT $= 0$ must use $\Omega(n)$ space.

Finally, we present a space lower bound on streaming algorithms that approximate the optimum value of Max-AND-SAT up to a factor $1/2+\epsilon$. In particular, we show that such algorithms must use $\Omega(mn)$ space.

Notation and preliminaries. We occasionally use set notation for clauses. For example, we write $x_i \in C_j$ (or $\overline{x_i} \in C_j$) if the *literal* x_i (or $\overline{x_i}$ respectively) is in clause C_j. Furthermore, $C_j \setminus x_i$ denotes the clause C_j with the literal x_i removed. We often write OPT(P) to denote the optimal value of the problem P, and when the context is clear, we drop P and just write OPT. We write poly(x) to denote x^c for an arbitrarily large constant c. The constant c is absorbed in the big-O. Throughout this paper, we use K to denote a universally large enough constant. We assume that our algorithms know m or an upper bound of m in advance; this is necessary since the space depends on $\log m$.

We use the following version of Chernoff bound.

▶ **Theorem 4** (Chernoff bound). *If $X = \sum_{i=1}^{n} X_i$ where X_i are negatively correlated binary random variables and $\mathbb{P}(X_i = 1) = p$, then for any $\eta > 0$, $\mathbb{P}(|X - pn| \geq \eta pn) \leq 2 \cdot \exp\left(-\frac{\eta^2}{2+\eta} \cdot pn\right)$.*

2 Streaming Algorithms for Max-SAT

Without loss of generality, we may assume that there is no *trivially true* clause, i.e., clauses that contain both a literal and its negation and there are no duplicate literals in a clause. We show that if we sample $\Theta(n/\epsilon^2)$ clauses uniformly at random and run a constant approximation algorithm for Max-SAT on the sampled clauses, we obtain roughly the same approximation. Note that if $m \leq Kn/\epsilon^2$ we might skip the sampling part.

▶ **Lemma 5.** *For* `Max-SAT`*, an* α *approximation on* Kn/ϵ^2 *clauses sampled uniformly at random corresponds to an* $\alpha - \epsilon$ *approximation on the original input clauses with probability at least* $1 - e^{-n}$*.*

Proof. We recall the folklore fact that $\texttt{OPT} \geq m/2$. Consider an arbitrary assignment. If it satisfies fewer than $m/2$ clauses, we can invert the assignment to satisfy at least $m/2$ clauses.

Suppose that an assignment A satisfies m_A clauses. Let the number of sampled clauses that A satisfies be m'_A and let $p = Kn/(\epsilon^2 m)$. For convenience, let $y = m_A$ and $y' = m'_A$. We observe that $\mathbb{E}[y'] = py$.

Suppose the assignment A satisfies clauses $C_{\sigma_1}, \ldots, C_{\sigma_y}$. We define the indicator variable $X_i = [C_{\sigma_i} \text{ is sampled}]$ and so $y' = \sum_{i=1}^{y} X_i$. Let $\eta = \epsilon \,\texttt{OPT}\,/y$. Since we sample without replacement, $\{X_i\}$ are negatively correlated. Appealing to Chernoff bound, we have

$$\begin{aligned}\mathbb{P}\left(|y' - py| \geq \epsilon p \,\texttt{OPT}\right) &\leq 2 \cdot \exp\left(-\frac{\eta^2}{2+\eta} py\right) \leq 2 \cdot \exp\left(-\frac{\epsilon^2 \,\texttt{OPT}^2/y^2}{2 + \epsilon \,\texttt{OPT}/y} py\right) \\ &= 2 \cdot \exp\left(-\frac{\epsilon^2 \,\texttt{OPT}^2}{2y + \epsilon \,\texttt{OPT}} p\right) \leq 2 \cdot \exp\left(-\epsilon^2 \,\texttt{OPT}\, p/3\right).\end{aligned}$$

The last inequality follows because $3\,\texttt{OPT} \geq 2y + \epsilon \,\texttt{OPT}$. Therefore,

$$\begin{aligned}\mathbb{P}\left(m'_A = pm_A \pm \epsilon p \,\texttt{OPT}\right) &\geq 1 - 2 \cdot \exp\left(-\epsilon^2 p \,\texttt{OPT}/3\right) = 1 - 2 \cdot \exp\left(-\epsilon^2 (Kn/(m\epsilon^2))\,\texttt{OPT}/3\right) \\ &\geq 1 - 2 \cdot \exp\left(-Kn/6\right) \geq 1 - \exp\left(-100n\right).\end{aligned}$$

The second inequality follows from the fact that $\texttt{OPT} \geq m/2$. A union bound over 2^n distinct assignments implies that with probability at least $1 - e^{-n}$, we have $m'_A = pm_A \pm \epsilon p \,\texttt{OPT}$ for all assignments A.

Suppose an assignment $\tilde{A}$ is an α approximation to `Max-SAT` on the sampled clauses. Let A^* be an optimal assignment on the original input clauses. From the above, with probability at least $1 - e^{-n}$, we have $pm_{\tilde{A}} + \epsilon p \,\texttt{OPT} \geq m'_{\tilde{A}} \geq \alpha m'_{A^*} \geq \alpha \,\texttt{OPT}\, p(1-\epsilon)$. Hence, $m_{\tilde{A}} \geq \alpha \,\texttt{OPT}(1-\epsilon) - \epsilon \,\texttt{OPT} \geq (\alpha - 2\epsilon)\,\texttt{OPT}$. Reparameterizing $\epsilon \leftarrow \epsilon/2$ completes the proof. ◀

Note that storing a clause may require $\Omega(n)$ space and hence the space use can still be $\Omega(n^2/\epsilon^2)$ after sampling. We then observe that large clauses are probabilistically easy to satisfy. We define β-large clauses as clauses that have at least β literals.

▶ **Lemma 6.** *If each literal is set to* `true` *independently with probability at least* γ*, then the assignment satisfies all* $(K \log m)/\gamma$*-large clauses with probability at least* $1 - 1/\operatorname{poly}(m)$*.*

Proof. The probability that a $(K \log m)/\gamma$-large clause is not satisfied is at most $(1-\gamma)^{(K \log m)/\gamma} \leq e^{-K \log m} \leq 1/\operatorname{poly}(m)$ which implies that all such clauses are satisfied with probability at least $1 - 1/\operatorname{poly}(m)$ by appealing to a union bound over at most m large clauses. ◀

From the above observations, we state a simple meta algorithm, formalized as Algorithm 1, that can easily be implemented in several sublinear settings. We will then present two possible post-processing algorithms and the corresponding γ values for this meta algorithm.

Let L and S be the set of β-large and small clauses respectively. Furthermore, let $\texttt{OPT}_L$ and $\texttt{OPT}_S$ be the number of satisfied clauses in L and S respectively in the optimal assignment.

Algorithm 1 A meta algorithm for sublinear `Max-SAT`

1. Ignore all β-large clauses where $\beta = (K \log m)/\gamma$. Among the remaining clauses, sample and store Kn/ϵ^2 clauses uniformly at random. Let the set of collected clauses be W.
2. **Post-processing:** Run an α approximation for `Max-SAT` on the collected clauses W in which each literal is set to `true` independently with probability at least γ.

Post-processing algorithm 1: An exponential-time $1-\epsilon$ approximation. Here, we set $\gamma = \epsilon$. Suppose in post-processing, we run the exact algorithm to find an optimal assignment A^* on the set of collected clauses W. Ideally, we would like to apply Lemma 5 to argue that we yield a $1-\epsilon$ approximation on the small clauses in S w.h.p. and then apply Lemma 6 to argue that we satisfy all the large clauses L w.h.p. However, the second claim requires that each literal is set to `true` with probability at least ϵ whereas the exact algorithm is deterministic. Our trick is to randomly perturb the assignment given by the exact algorithm.

If the exact algorithm sets $x_i = q \in \{\texttt{true}, \texttt{false}\}$, we set $x_i = q$ with probability $1-\epsilon$ (and $x_i = \overline{q}$ with probability ϵ). We will show that this yields a $1-\epsilon$ approximation in expectation which can then be repeated $O(\log m/\epsilon)$ times to obtain the "w.h.p." guarantee.

Algorithm 2 A post-processing algorithm that obtains a $1-2\epsilon$ approximation w.h.p.

1. Obtain an optimal assignment A^* on W. Let $Q = \lceil (K \log m)/\epsilon \rceil$.
2. For each trial $t = 1, 2, \ldots, Q$, if $x_i = q$ in A^*, then set $x_i = q$ with probability $1-\epsilon$ and $x_i = \overline{q}$ with probability ϵ in assignment A_t.
3. Return the assignment A_t that satisfies the most number of clauses in W.

Before analyzing the above post-processing algorithm, we observe that if we obtain an $\alpha \geq 1/2$ approximation in expectation, we can repeat the corresponding algorithm $O(\log m/\epsilon)$ times and choose the best solution to obtain an $\alpha - \epsilon$ approximation w.h.p.

▶ **Lemma 7.** *An $\alpha \geq 1/2$ approximation to `Max-SAT` in expectation can be repeated $O(\log m/\epsilon)$ times to yield an $\alpha - \epsilon$ approximation w.h.p.*

Proof. Let Z be the number of clauses satisfied by the algorithm. First, note that since $m/4 \leq \alpha\,\texttt{OPT}$ (since $\alpha \geq 1/2$), we have $\alpha\,\texttt{OPT}/(m - \alpha\,\texttt{OPT}) \geq 1/3$. We have $\mathbb{E}[m - Z] \leq m - \alpha\,\texttt{OPT}$. Appealing to Markov inequality,

$$\begin{aligned}\mathbb{P}(Z \leq (1-\epsilon)\alpha\,\texttt{OPT}) &= \mathbb{P}(m - Z \geq m - (1-\epsilon)\alpha\,\texttt{OPT}) \\ &\leq \frac{m - \alpha\,\texttt{OPT}}{m - (1-\epsilon)\alpha\,\texttt{OPT}} = \frac{1}{1 + \epsilon\alpha\frac{\texttt{OPT}}{(m - \alpha\,\texttt{OPT})}} \leq \frac{1}{1+\epsilon/3}.\end{aligned}$$

Hence, we can repeat the algorithm $O(\log_{1+\epsilon/3} m) = O(1/\epsilon \cdot \log m)$ times and choose the best solution to obtain an $\alpha - \epsilon$ approximation with probability at least $1 - 1/\operatorname{poly}(m)$. ◀

We now analyze the post-processing Algorithm 2.

▶ **Lemma 8.** *Algorithm 2 yields a $1-3\epsilon$ approximation on the original input w.h.p.*

Proof. Clearly, in each trial $t = 1, 2, \ldots, Q$, each literal is set to `true` with probability at least ϵ. According to Lemma 6, each assignment A_t satisfies all the large clauses in L with

probability at least $1 - 1/\operatorname{poly}(m)$. Taking a union bound over $Q < m$ trials, we conclude that all assignments A_t satisfy all the clauses in L with probability at least $1 - 1/\operatorname{poly}(m)$.

Next, let B be the set of clauses in W satisfied by the exact assignment A^*. Consider any trial t. The expected number of satisfied clauses in B after we randomly perturb the exact assignment is

$$\sum_{C \in B} \mathbb{P}\left(C \text{ is satisfied}\right) \geq \sum_{C \in B} (1 - \epsilon) = (1 - \epsilon)|B|.$$

The first inequality follows from the observation that at least one literal in C must be `true` in the exact assignment and it remains `true` with probability at least $1 - \epsilon$. The assignment returned by post-processing Algorithm 2 yields a $1 - 2\epsilon$ approximation on W with probability at least $1 - 1/\operatorname{poly}(m)$ by Lemma 7. This, in turn, implies that we obtain a $1 - 3\epsilon$ approximation on S with probability at least $1 - 1/\operatorname{poly}(m) - e^{-n} \geq 1 - 1/\operatorname{poly}(m) - 1/\operatorname{poly}(n)$ by Lemma 5.

Therefore, we satisfy at least $(1 - 3\epsilon)\,\texttt{OPT}_S + \texttt{OPT}_L \geq (1 - 3\epsilon)\,\texttt{OPT}$ clauses with probability at least $1 - 1/\operatorname{poly}(m) - 1/\operatorname{poly}(n)$. ◀

Post-processing algorithm 2: A polynomial-time $3/4 - \epsilon$ approximation. We can set $\beta = K \log m$ if we settle for a $3/4 - \epsilon$ approximation. This saves a factor $1/\epsilon$ in the memory use. Consider the standard linear programming (LP) formulation for `Max-SAT` given by Goemans and Williamson [18].

$$\begin{aligned} (LP) \quad \text{maximize} \quad & \sum_{j=1}^{m} z_j \\ \text{subject to} \quad & \sum_{i \in P_j} y_i + \sum_{i \in N_j} (1 - y_i) \geq z_j && \text{for all } 1 \leq j \leq m \\ & 0 \leq y_i, z_j \leq 1 && \text{for all } 1 \leq i \leq n, 1 \leq j \leq m\ . \end{aligned}$$

The integer linear program where $y_i, z_j \in \{0, 1\}$ corresponds exactly to the `Max-SAT` problem. In particular, if $x_i \in C_j$ then $i \in P_j$ and if $\overline{x_i} \in C_j$ then $i \in N_j$. We associate $y_i = 1$ with x_i being set to `true` and $y_i = 0$ if it is set to `false`. Similarly, we associate $z_i = 1$ with clause C_j being satisfied and 0 otherwise. Let $\texttt{OPT}(LP)$ denote the optimum of the above LP.

▶ **Lemma 9** ([18], Theorem 5.3)**.** *The optimal fractional solution of the LP for* `Max-SAT` *can be rounded to yield a 3/4 approximation in expectation by independently setting each variable to* `true` *with probability* $1/4 + y_i^*/2$ *where* y_i^* *is the value of* y_i *in* $\texttt{OPT}(LP)$*.*

Algorithm 3 A post-processing algorithm that obtains a $3/4 - 2\epsilon$ approximation w.h.p.

1. Obtain an optimal solution z^*, y^* for the linear program of `Max-SAT` on W.
2. Let $Q = \lceil (K \log m)/\epsilon \rceil$. For each trial $t = 1, 2, \ldots, Q$, set $x_i =$ `true` with probability $1/4 + y_i^*/2$ in assignment A_t.
3. Return the assignment A_t that satisfies the most number of clauses in W.

The following lemma and its proof are analogous to Lemma 8.

▶ **Lemma 10.** *Algorithm 3 yields a* $3/4 - 3\epsilon$ *approximation on the original input w.h.p.*

Proof. In each trial $t = 1, 2, \ldots, Q$, each literal is set to `true` with probability at least $1/4$. This is because $1/4 \leq 1/4 + y_i^*/2 \leq 3/4$. Appealing to Lemma 6 with $\gamma = 1/4$, all assignments A_t satisfy all the large clauses in L with probability at least $1 - 1/\operatorname{poly}(m)$. The rest of the argument is then similar to that of Lemma 8. ◀

Handling deletions. To support deletions, we use the L_0 sampler in [24, 26]. Suppose each stream token is either an entry deletion or insertion on a vector v of size N then the L_0 sampler returns a non-zero coordinate j uniformly at random. The sampler succeeds with probability at least $1 - \delta$ and uses $O(\log^2 N \cdot \log(1/\delta))$ space. We can use a vector of size $N = 2^{2n}$ to encode the characteristic vector of the set of the clauses we have at the end of the stream. However, this results in an additional factor n^2 in the space use. Fortunately, since we only consider small clauses of size at most β, we can use a vector of size

$$N = \binom{2n}{1} + \binom{2n}{2} + \ldots + \binom{2n}{\beta} \leq \sum_{i=1}^{\beta} (2n)^i \leq O((2n)^\beta).$$

Thus, to sample a small clause with probability at least $1 - 1/\operatorname{poly}(n)$, we need $O(\beta^2 \log^3 n)$ space. To sample Kn/ϵ^2 small clauses, the space is $\tilde{O}(n\beta^2/\epsilon^2)$. Finally, to sample clauses without replacement, one may use the approach described by McGregor et al. [37] that avoids an increase in space.

Note that a naive implementation to sample s clauses is to run s different L_0 samplers in parallel which results in an update time of $\tilde{O}(s)$. However, [37] showed that it is possible to improve the update time to $\tilde{O}(1)$ with an additional factor $\log N = O(\beta \log n)$ in the space use.

Putting it all together. We finalize the proof of the first main result by outlining the implementation of the algorithms above in the streaming model. We further present an improvement that saves another $1/\epsilon$ factor under the no-duplicate assumption in the $3/4 - \epsilon$ approximation.

Proof of Theorem 1 (1). We ignore β-large clauses during the stream. Among the remaining clauses, we can sample and store $\Theta(n/\epsilon^2)$ small clauses in the stream uniformly at random as described.

For insertion-only streams, we may use Reservoir sampling [42] to sample clauses. Since storing each small clause requires $\tilde{O}(\beta)$ space, the total space is $\tilde{O}(n\beta/\epsilon^2)$. We can run post-processing Algorithm 2 where $\beta = (K \log m)/\epsilon$ and yield a $1 - \epsilon$ approximation; alternatively, we set $\beta = (K \log m)$ and run the polynomial post-processing Algorithm 3 to yield a $3/4 - \epsilon$ approximation.

If there is no-duplicate, we can further improve the space for the $3/4 - \epsilon$ approximation as follows. If the number of 1-literal clauses is at most ϵm, we can safely ignore these clauses. This is because the number of 1-literal clauses is then at most $2\epsilon\,\texttt{OPT}$. If we randomly set each variable to `true` with probability $1/2$, we will yield a $3/4$ approximation in expectation on the remaining clauses since each of these clause is satisfied with probability at least $1 - 1/2^2 = 3/4$. By Lemma 7, we can run $O((\log m)/\epsilon)$ copies in parallel and return the best assignment to yield a $3/4 - \epsilon$ approximation w.h.p. Therefore, the overall approximation is $3/4 - 3\epsilon$ and the space is $O(n/\epsilon \cdot \log m)$. If the number of 1-literal clauses is more than ϵm, then we know that $m \leq 2n/\epsilon$ since the number of 1-literal clauses is at most $2n$ if there are no duplicate clauses. Since $m \leq 2n/\epsilon$, the sampling step can be ignored. It then suffices to

store only clauses of size at most $K \log m$ and run post-processing Algorithm 3. The space use is $O(m \log m) = \tilde{O}(n/\epsilon)$. ◀

Proof of Theorem 1 (2). For insertion-deletion streams without duplicates, we sample clauses using the L_0 sampler as described earlier. We can run post-processing Algorithm 2 where $\beta = (K \log m)/\epsilon$ and obtain a $1-\epsilon$ approximation while using $\tilde{O}(n/\epsilon^4)$ space. Alternatively, we set $\beta = (K \log m)$ and run the polynomial post-processing Algorithm 3 to yield a $3/4-\epsilon$ approximation. This leads to a $\tilde{O}(n/\epsilon^2)$-space algorithm that yields a $3/4-\epsilon$ approximation. ◀

We remark that it is possible to have an approximation slightly better than $3/4-\epsilon$ in polynomial post-processing time using semidefinite programming (SDP) instead of linear programming [19]. In the SDP-based algorithm that obtains a $0.7584-\epsilon$ approximation, we also have the property that each literal is true with probability at least some constant. The analysis is analogous to what we outlined above.

3 Streaming Algorithms for `Min-SAT`

In this section, we provide several algorithms for `Min-SAT` in the streaming setting.

An approach based on F_0 sketch. We first show that using F_0 sketch, we can obtain a $1+\epsilon$ approximation in $\tilde{O}(n^2/\epsilon^2)$ space. Given a vector $x \in \mathbb{R}^n$, $F_0(x)$ is defined as the number of elements of x which are non-zero. Consider a subset $S \subseteq \{1,\ldots,n\}$, let $x_S \in \{0,1\}^n$ be characteristic vector of S (i.e., $x_i = 1$ iff $i \in S$). Note that $F_0(x_{S_1} + x_{S_2} + \ldots)$ is exactly the coverage $|S_1 \cup S_2 \cup \ldots|$. We will use the following result for estimating F_0.

▶ **Theorem 11** (F_0 Sketch, [9]). *Given a set $S \subseteq [n]$, there exists an $\tilde{O}(\epsilon^{-2} \log \delta^{-1})$-space algorithm that constructs a data structure $\mathcal{M}(S)$ (called an F_0 sketch of S). The sketch has the property that the number of distinct elements in a collection of sets $S_1, S_2, \ldots, S_t$ can be approximated up to a $1+\epsilon$ factor with probability at least $1-\delta$ provided the collection of F_0 sketches $\mathcal{M}(S_1), \mathcal{M}(S_2), \ldots, \mathcal{M}(S_t)$.*

The above immediately gives us a $1+\epsilon$ approximation in $\tilde{O}(n^2/\epsilon^2)$ space. Each literal ℓ corresponds to a set S_ℓ that contains the clauses that it is in, i.e., $S_\ell = \{C_j : \ell \in C_j\}$. Hence, the goal is to find a combination of $\ell_1, \ell_2, \ldots, \ell_n$ where $\ell_i \in \{x_i, \overline{x_i}\}$ such that the coverage $|C_{\ell_1} \cup \ldots \cup C_{\ell_n}|$ is minimized. We can construct $\mathcal{M}(S_{x_i})$ and $\mathcal{M}(S_{\overline{x_i}})$ for each $i = 1, 2, \ldots, n$ with failure probability $\delta = 1/(n2^n)$. Then, we return the smallest estimated coverage among 2^n such combinations based on these sketches. This is a $1+\epsilon$ approximation w.h.p. since for all $\ell_1, \ell_2, \ldots, \ell_n$ where $\ell_i \in \{x_i, \overline{x_i}\}$, we have an estimate $(1 \pm \epsilon)|S_{\ell_1} \cup \ldots \cup S_{\ell_n}|$ with probability $1 - 1/n$ by a simple union bound.

This approach's drawback is its exponential post-processing time. To see why this is hard to extend to other offline algorithms, let us review the algorithm by Kohli et al. [29] that yields a 2-approximation in expectation. The algorithm goes through the variables $x_1, x_2, \ldots, x_n$ one by one. At step i, it processes the variable x_i. Let a_i and b_i be the number of newly satisfied clauses if we assign x_i to `true` and `false` respectively. We randomly set $x_i \leftarrow$ `true` with probability $\frac{b_i}{a_i+b_i}$ and set $x_i \leftarrow$ `false` with probability $\frac{a_i}{a_i+b_i}$. The algorithm updates the set of satisfied clauses and move on to the next variable. A simple induction shows that this is a 2-approximation in expectation. Note that we can run this algorithm $O(\log n/\epsilon)$ times and return the best solution to obtain a $2+\epsilon$ approximation w.h.p.

Unfortunately, F_0 sketch does not support set subtraction (i.e., if we have the sketches $\mathcal{M}(A)$ and $\mathcal{M}(B)$, we are not guaranteed to get a $1 \pm \epsilon$ multiplicative approximation of $|A \setminus B|$) and thus it is unclear how to compute or estimate a_i and b_i at each step i.

Besides, better approximations to Min-k-SAT, for small values of k, have also been developed [5, 8, 11] using linear and semidefinite programming. It is not clear how to combine F_0 sketch with these approaches either. We now show how to sidestep the need to use F_0 sketch entirely and run any offline algorithm of our choice.

The subsampling framework. We first present a framework that allows us to assume that the optimum value is at most $O(n/\epsilon^2)$.

▶ **Lemma 12.** *Suppose* $\frac{Kn}{\epsilon^2} \leq \texttt{OPT} \leq z \leq 2\,\texttt{OPT}$ *and* $0 \leq \epsilon < 1/4$. *An* α *approximation to* Min-SAT *on clauses sampled independently with probability* $p = \frac{Kn}{\epsilon^2 z}$ *corresponds to an* $\alpha + \epsilon$ *approximation on the original input w.h.p. Furthermore, the optimum of the sampled clauses* $\texttt{OPT}' = O(n/\epsilon^2)$.

Proof. Suppose that an assignment A satisfies m_A clauses. Let the number of sampled clauses that A satisfies be m'_A. For convenience, let $y = m_A$ and $y' = m'_A$. We observe that $\mathbb{E}[y'] = py$. Note that $y \geq \texttt{OPT} \geq z/2$. Suppose the assignment A satisfies clauses $C_{\sigma_1}, \ldots, C_{\sigma_y}$. We define the indicator variable $X_i = [C_{\sigma_i} \text{ is sampled}]$ and so $y' = \sum_{i=1}^{y} X_i$. Appealing to Chernoff bound,

$$\mathbb{P}\left(|y' - py| \geq \epsilon py\right) \leq 2\exp\left(-\frac{\epsilon^2}{3}py\right) \leq 2\exp\left(-\frac{K\epsilon^2 ny}{z\epsilon^2}\right) \leq \exp\left(-100n\right).$$

Taking a union bound over 2^n distinct assignments, we have $\mathbb{P}\left(m'_A = (1 \pm \epsilon)pm_A\right) \geq 1 - \exp(-50n)$ for all assignments A. We immediately have that $\texttt{OPT}' \leq (1+\epsilon)p\,\texttt{OPT} = O(n/\epsilon^2)$.

Suppose an assignment $\tilde{A}$ is an α approximation to Min-SAT on the sampled clauses. Let A^* be an optimal assignment on the original input clauses. From the above, with probability at least $1 - e^{-50n}$, we have $(1-\epsilon)pm_{\tilde{A}} \leq m'_{\tilde{A}} \leq \alpha m'_{A^*} \leq \alpha\,\texttt{OPT}\,p(1+\epsilon)$. Hence, $m_{\tilde{A}} \leq (1+3\epsilon)\alpha\,\texttt{OPT}$. Reparameterizing $\epsilon \leftarrow \epsilon/3$ completes the proof. ◀

Based on the above lemma, we can run any α approximation algorithm on sampled clauses with sampling probability $p = \frac{Kn}{\epsilon^2 z}$ to yield an almost as good approximation. Furthermore, the optimum value on these sampled clauses is at most $O(n/\epsilon^2)$.

Since we do not know z, we can run different instances of our algorithm corresponding to different guesses $z = 1, 2, 4, \ldots, 2m$. At least one of these guesses satisfies $\texttt{OPT} \leq z \leq 2\,\texttt{OPT}$. The algorithm instances that correspond to wrong guesses may use too much space. In that case, we simply terminate those instances. We then return the best solutions among the instances that are not terminated. Hereinafter, we may safely assume that $\texttt{OPT} = O(n/\epsilon^2)$.

Proof of Theorem 3 . We first show that an $\tilde{O}(n^2/\epsilon^2)$-space algorithm exists. Let $\mathcal{C}(\ell) = \{C_j : \ell \in C_j\}$ be the set of clauses that contains the literal ℓ. It must be the case that either all clauses $\mathcal{C}(x_i)$ are satisfied or all clauses in $\mathcal{C}(\overline{x_i})$ are satisfied. For time being, assume we have an upper bound u of $\texttt{OPT}$.

Originally, all variables are marked as unsettled. If at any point during the stream, $|\mathcal{C}(x_i)| > u$, we can safely set $x_i \leftarrow \texttt{false}$; otherwise, $\texttt{OPT} > u$ which is a contradiction. We then say the variable x_i is settled. The case in which $|\mathcal{C}(\overline{x_i})| > u$ is similar.

When a clause C_j arrives, if C_j contains a settled literal ℓ that was set to `true`, we may simply ignore C_j since it must be satisfied by any optimal solution. On the other hand, if the fate of C_j has not been decided yet, we store $C_j \setminus Z$ where Z is the set of settled literals in

C_j that were set to `false` at this point. For example, if $C_j = (x_1 \vee x_2 \vee \overline{x_3})$ and $x_1 \leftarrow$ `false` at this point while x_2 and x_3 are unsettled, then we simply store $C_j = (x_2 \vee \overline{x_3})$.

Since we store the input induced by unsettled variables and each unsettled variable appears in at most $2u$ clauses (positively or negatively), the space use is $\tilde{O}(nu)$. By using the aforementioned subsampling framework, we may assume that $u = O(n/\epsilon^2)$. Thus, the total space is $\tilde{O}(n^2/\epsilon^2)$. ◀

If each variable appears in at most f clauses, we have a slightly non-trivial approximation that uses less space.

▶ **Theorem 13.** *Suppose each variable appears in at most f clauses. There exists a single-pass, $\tilde{O}(n)$-space algorithm that yields a $2\sqrt{fn}$ approximation to* `Min-SAT`.

Proof. It is easy to check if $\texttt{OPT} = 0$ in $\tilde{O}(n)$ space. For each variable x_i, we keep track of whether it always appears positively (or negatively); if so, it is safe to set $x_i \leftarrow$ `false` (or $x_i \leftarrow$ `true` respectively). If that is the case for all variables, then $\texttt{OPT} = 0$. We run this in parallel with our algorithm that handles the case $\texttt{OPT} > 0$.

Now, assume $\texttt{OPT} \geq 1$. Since each variable involves in at most f clauses. There are fewer than $\sqrt{fn}$ clauses with size larger than $\sqrt{fn}$. We ignore these large clauses from the input stream. Let $S_{x_i} = \{C_j : |C_j| \leq \sqrt{fn} \text{ and } x_i \in C_j\}$. For each variable x_i, if $|S_{x_i}| \leq |S_{\overline{x_i}}|$, then we set $x_i \leftarrow$ `false`; otherwise, we set $x_i \leftarrow$ `true`. Let $\ell_i = x_i$ if x_i was set to `true` and $\ell_i = \overline{x_i}$ if x_i was set to `false` by our algorithm. Furthermore, let $\ell_i^\star = x_i$ if x_i was set to `true` and $\ell_i^\star = \overline{x_i}$ if x_i was set to `false` in the optimal assignment. Let $\texttt{OPT}_1$ be the number of small clauses that are satisfied by the optimal assignment and let $\texttt{OPT}_2$ be the number of large clauses that are satisfied by the optimal assignment.

Note that $\sum_{i=1}^n |S_{\ell_i^\star}| \leq \sqrt{fn}\,\texttt{OPT}_1$ since each small clauses belong to at most $\sqrt{fn}$ different $S_{\ell_i^\star}$. The number of clauses that our algorithm satisfies at most

$$\sqrt{fn} + \sum_{i=1}^n |S_{\ell_i}| \leq \sqrt{fn} + \sum_{i=1}^n |S_{\ell_i^\star}| \leq \sqrt{fn}\,\texttt{OPT}_2 + \sqrt{fn}\,\texttt{OPT}_1 \leq 2\sqrt{fn}\,\texttt{OPT}\,. \qquad ◀$$

4 Space Lower Bounds

Lower bound for k-`SAT`. Let us first prove that any streaming algorithm that decides whether a k-`SAT` formula, where $k < \log m$, is satisfiable requires $\Omega(\min\{m, n^k\})$ space. We consider the one-way communication problem k-`SAT` defined as follows. In this communication problem, Alice and Bob each has a set of k-`SAT` clauses. Bob wants to decide if there is a Boolean assignment to the variables that satisfies all the clauses from both sets. The protocol can be randomized, but the requirement is that the success probability is at least some large enough constant. Note that a streaming algorithm that solves k-`SAT` yields a protocol for the k-`SAT` communication problem since Alice can send the memory of the algorithms to Bob.

We proceed with a simple claim. If $k = 2$, then this claim says that there is no assignment that satisfies all of the following clauses $(x \vee y), (\overline{x} \vee y), (x \vee \overline{y}), (\overline{x} \vee \overline{y})$. The claim generalizes this fact for all $k \in \mathbb{N}$.

▷ Claim 14. Consider k Boolean variables $x_1, \ldots, x_k$. No Boolean assignment simultaneously satisfies all clauses in the set $S_k = \{(\bigvee_{\ell \in S} \overline{x_\ell}) \bigvee (\bigvee_{\ell' \notin S} x_{\ell'}) : S \subseteq [k]\}$ for all $k \in \mathbb{N}$.

Proof. The proof is a simple induction on k. The base case $k = 1$ is trivial since S_1 consists of two clauses x_1 and $\overline{x_1}$. Now, consider $k > 1$. Suppose that there is an assignment that

satisfies all clauses in S_k. Without loss of generality, suppose $x_k = \texttt{true}$ in that assignment (the case $x_k = \texttt{false}$ is analogous). Consider the set of clauses $A = \{\overline{x_k} \vee \phi : \phi \in S_{k-1}\} \subset S_k$. Since $\overline{x_k} = \texttt{false}$, it must be the case that all clauses in S_{k-1} are satisfied which contradicts the induction hypothesis. ◀

Without loss of generality, we assume n/k is an integer. Now, we consider the one-way communication problem Index. In this problem, Alice has a bit-string A of length t where each bit is independently set to 0 or 1 uniformly at random. Bob has a random index i that is unknown to Alice. For Bob to output A_i correctly with probability 2/3, Alice needs to send a message of size $\Omega(t)$ bits [30]. We will show that a protocol for for k-SAT will yield a protocol for Index.

Proof of Theorem 2. Without loss of generality, assume n/k is an integer. Consider the case where Alice has a bit-string A of length $(n/k)^k$. For convenience, we index A as $[n/k] \times \ldots \times [n/k] = [n/k]^k$. We consider n Boolean variables $\{x_{a,b}\}_{a\in[k],b\in[n/k]}$.

For each $j \in [n/k]^k$ where $A_j = 1$, Alice generates the clause $(x_{1,j_1} \vee x_{2,j_2} \vee \ldots \vee x_{k,j_k})$. Let $S = \{(1, i_1), (2, i_2) \ldots, (k, i_k)\}$. Bob, with the index $i \in [n/k]^k$, generates the clauses in

$$\left\{ \left(\bigvee_{l\in Z} \overline{x_l} \right) \left(\bigvee_{l'\in S\setminus Z} x_{l'} \right) : Z \subseteq S \right\} \setminus \{(x_{1,i_1} \vee x_{2,i_2} \vee \ldots \vee x_{k,i_k})\}.$$

If $A_i = 0$, then all the generated clauses can be satisfied by setting the variables $x_l = \texttt{false}$ for all $l \in S$ and all remaining variables to $\texttt{true}$. All the clauses that Bob generated are satisfied since each contains at least one literal among $\overline{x_{1,i_1}}, \overline{x_{2,i_2}} \ldots, \overline{x_{k,i_k}}$. Note that the clause $(x_{1,i_1} \vee x_{2,i_2} \vee \ldots \vee x_{k,i_k})$ does not appear in the stream since $A_i = 0$. For any $j \in [n/k]^k$, where $(j_1, \ldots, j_k) \neq (i_1, \ldots, i_k)$, such that $A_j = 1$, there must be some $j_z \notin \{i_1, i_2, \ldots, i_k\}$. Hence, $x_{z,j_z} = \texttt{true}$ and therefore the clause $(x_{1,j_1} \vee \ldots \vee x_{k,j_k})$ is satisfied.

If $A_i = 1$, then by Claim 14, we cannot simultaneously satisfy the clauses

$$\left\{ \left(\bigvee_{l\in Z} \overline{x_l} \right) \left(\bigvee_{l'\in S\setminus Z} x_{l'} \right) : Z \subseteq S \right\}$$

that were generated by Alice and Bob. Hence, any protocol for k-SAT yields a protocol for Index. Hence, such a protocol requires $\Omega(n^k)$ bits of communication.

Finally, we check the parameters' range where the lower bound operates. We first show that in the above construction, the number of clauses m concentrates around $\Theta((n/k)^k)$. First, note that for fixed n, the function $(n/k)^k$ is increasing in the interval $k \in [1, n/e]$. Since we assume $k \leq n/e$, we have $(n/k)^k \geq n$. Let m_A be the number of clauses Alice generates. Because independently each $A_j = 1$ with probability 1/2, by Chernoff bound, we have

$$\mathbb{P}\left(|m_A - 0.5(n/k)^k| > 0.01(n/k)^k\right) \leq \exp\left(-\Omega((n/k)^k)\right) \leq \exp\left(-\Omega(n)\right).$$

Bob generates $m_B = 2^k - 1$ clauses. Thus, w.h.p., $m = m_A + m_B = \left(0.5(n/k)^k + 2^k - 1\right) \pm 0.01(n/k)^k$. Note that $2^k \leq (n/k)^k$ since $k \leq n/e < n/2$. As a result, w.h.p., $0.49(n/k)^k - 1 \leq m \leq 1.51(n/k)^k + 1$ which implies $m = \Theta((n/k)^k)$.

When the formula is not satisfiable, the number of clauses $m \geq 2^k \iff k \leq \log m$. When the formula is satisfiable, the number of clauses $m \geq 2^k - 1 \iff k \leq \log(m+1)$. ◀

Note that the streaming algorithm (or the communication protocol) does not know m *exactly* in advance. If the algorithm knows m exactly in advance and it happens that

$\log(m+1) \geq k > \log m$ then it could simply accepts since there is a satisfying assignment via a probabilistic argument. Nonetheless, it is enough to remark that the lower bound operates in the setting where $k \leq \log m$.

Lower bound for `Max-AND-SAT`. We show that a $(1/2+\epsilon)$-approximation for `Max-AND-SAT` requires $\Omega(mn)$ space. In particular, our lower bound uses the algorithm to distinguish the two cases $\texttt{OPT} = 1$ and $\texttt{OPT} = 2$. This lower bound is also a reduction from the Index problem. Recall that in `Max-AND-SAT`, each clause is a conjunction of literals. We first need a simple observation.

▷ Claim 15. Let $T = K \log m$. There exists a set of m (conjunctive) clauses $C_1, \ldots, C_m$ over the set of Boolean variables $\{z_1, \ldots, z_T\}$ such that exactly one clause can be satisfied.

Proof. We show this via a probabilistic argument. For $i = 1, 2, \ldots, m$ and $j \in [T]$, let $C_i = \wedge_{j=1}^{T} l_{i,j}$ where each $l_{i,j}$ is independently set to z_j or $\overline{z_j}$ equiprobably.

Two different clauses C_i and $C_{i'}$ can be both satisfied if and only if for all $j = 1, 2, \ldots, T$, it holds that the variable z_j appears similarly (i.e., as z_j or as $\overline{z_j}$) in both clauses. This happens with probability $(1/2)^{K \log m} = 1/\operatorname{poly}(m)$. Hence, by a union bound over $\binom{m}{2}$ pairs of clauses, exactly one clause can be satisfy with probability at least $1 - 1/\operatorname{poly}(m)$. ◀

Alice and Bob agree on such a set of clauses $C_1, \ldots, C_m$ over the set of Boolean variables $\{z_1, \ldots, z_T\}$ (which is hard-coded in the protocol). Suppose Alice has a bit-string of length mn, indexed as $[m] \times [n]$. For each $k \in [m]$: if $A_{k,j} = 1$, she adds the literal x_j to clause D_k and if $A_{k,j} = 0$, she adds the literal $\overline{x_j}$ to clause D_k. Finally, she concatenates C_k to D_k. More formally, for each $k \in [m]$,

$$D_k = \left(\bigwedge_{j \in [n]: A_j = 1} x_j \right) \bigwedge \left(\bigwedge_{j \in [n]: A_j = 0} \overline{x_i} \right) \bigwedge (C_k) \,.$$

Bob, with the index $(i_1, i_2) \in [m] \times [n]$, generates the clauses $D_B = x_{i_2} \wedge C_{i_1}$. If $A_{i_1,i_2} = 1$, then both clauses D_B and D_{i_1} can be satisfied since the set of literals in D_B is a subset of the set of literals in D_{i_1}. If $A_{i_1,i_2} = 0$, exactly one of D_B and D_{i_1} can be satisfied since $x_{i_2} \in D_B$ and $\overline{x_{i_2}} \in D_{i_1}$. Furthermore, by Claim 15, we cannot simultaneously satisfy D_B and any other clause D_k where $k \neq i_1$ (since C_{i_1} is part of D_B and C_k is part of D_k). Therefore, if $\texttt{OPT} = 2$, then $A_{i_1,i_2} = 1$ and if $\texttt{OPT} = 1$ then $A_{i_1,i_2} = 0$. Thus, a streaming algorithm distinguishing the two cases yields a protocol for the communication problem. The number of variables in our construction is $n + K \log m \leq 2n$, if we assume $n \geq K \log m$. Reparameterizing $n \leftarrow n/2$ gives us the theorem below.

▶ **Theorem 16.** *Suppose $n = \Omega(\log m)$. Any single-pass streaming algorithm that decides if $\texttt{OPT} = 1$ or $\texttt{OPT} = 2$ for `Max-AND-SAT` with probability 2/3 must use $\Omega(mn)$ space.*

Lower bound for `Min-SAT`. For `Min-SAT`, it is easy to show that deciding if $\texttt{OPT} > 0$ requires $\Omega(n)$ space. The algorithm in the proof of Theorem 13 shows that this lower bound is tight.

▶ **Theorem 17.** *Any single-pass streaming algorithm that decides if $\texttt{OPT} > 0$ for `Min-SAT` with probability 2/3 must use $\Omega(n)$ space.*

Proof. Consider an Index instance of length n. If $A_j = 1$, Alice adds the literal x_j to the clause C_1; otherwise, she adds the literal $\overline{x_j}$ to C_1. Alice then puts C_1 in the stream. Bob, with the index i, puts the clause $C_2 = (x_i)$ in the stream. If $A_i = 0$, then $\overline{x_i} \in C_1$ and $x_i \in C_2$ and therefore $\texttt{OPT} > 0$. Otherwise, it is easy to see that we have an assignment that does not satisfy any of C_1 or C_2. ◀

References

1. Maxsat evaluation 2021. https://maxsat-evaluations.github.io/2021/index.html.
2. Akanksha Agrawal, Arindam Biswas, Édouard Bonnet, Nick Brettell, Radu Curticapean, Dániel Marx, Tillmann Miltzow, Venkatesh Raman, and Saket Saurabh. Parameterized streaming algorithms for min-ones d-sat. In *FSTTCS*, volume 150 of *LIPIcs*, pages 8:1–8:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.
3. Josep Argelich, Daniel Le Berre, Inês Lynce, João P. Marques Silva, and Pascal Rapicault. Solving linux upgradeability problems using boolean optimization. In *LoCoCo*, volume 29 of *EPTCS*, pages 11–22, 2010.
4. Josep Argelich and Felip Manyà. Partial max-sat solvers with clause learning. In *SAT*, volume 4501 of *Lecture Notes in Computer Science*, pages 28–40. Springer, 2007.
5. Umair Arif, Robert Benkoczi, Daya Ram Gaur, and Ramesh Krishnamurti. On the minimum satisfiability problem. In *CALDAM*, volume 12016 of *Lecture Notes in Computer Science*, pages 269–281. Springer, 2020.
6. Sepehr Assadi, Yu Chen, and Sanjeev Khanna. Sublinear algorithms for $(\Delta + 1)$ vertex coloring. In *SODA*, pages 767–786. SIAM, 2019.
7. Adi Avidor, Ido Berkovitch, and Uri Zwick. Improved approximation algorithms for MAX NAE-SAT and MAX SAT. In *WAOA*, volume 3879 of *Lecture Notes in Computer Science*, pages 27–40. Springer, 2005.
8. Adi Avidor and Uri Zwick. Approximating MIN k-sat. In *ISAAC*, volume 2518 of *Lecture Notes in Computer Science*, pages 465–475. Springer, 2002.
9. Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, D. Sivakumar, and Luca Trevisan. Counting distinct elements in a data stream. In *RANDOM*, volume 2483 of *Lecture Notes in Computer Science*, pages 1–10. Springer, 2002.
10. Suman K. Bera, Amit Chakrabarti, and Prantar Ghosh. Graph coloring via degeneracy in streaming and other space-conscious models. In *ICALP*, volume 168 of *LIPIcs*, pages 11:1–11:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.
11. Dimitris Bertsimas, Chung-Piaw Teo, and Rakesh Vohra. On dependent randomized rounding algorithms. *Oper. Res. Lett.*, 24(3):105–114, 1999.
12. Yibin Chen, Sean Safarpour, João Marques-Silva, and Andreas G. Veneris. Automated design debugging with maximum satisfiability. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 29(11):1804–1817, 2010.
13. Chi-Ning Chou, Alexander Golovnev, Madhu Sudan, and Santhoshini Velusamy. Approximability of all boolean csps in the dynamic streaming setting, 2021. arXiv:2102.12351.
14. Chi-Ning Chou, Alexander Golovnev, and Santhoshini Velusamy. Optimal streaming approximations for all boolean max-2csps and max-ksat. In *FOCS*, pages 330–341. IEEE, 2020.
15. Stephen A. Cook. The complexity of theorem-proving procedures. In *STOC*, pages 151–158. ACM, 1971.
16. James Cussens. Bayesian network learning by compiling to weighted MAX-SAT. In *UAI*, pages 105–112. AUAI Press, 2008.
17. M. R. Garey, David S. Johnson, and Larry J. Stockmeyer. Some simplified np-complete graph problems. *Theor. Comput. Sci.*, 1(3):237–267, 1976.
18. Michel X. Goemans and David P. Williamson. New 3/4-approximation algorithms for the maximum satisfiability problem. *SIAM J. Discrete Math.*, 7(4):656–666, 1994.
19. Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, 1995.
20. Venkatesan Guruswami, Ameya Velingker, and Santhoshini Velusamy. Streaming complexity of approximating max 2csp and max acyclic subgraph. In *APPROX-RANDOM*, volume 81 of *LIPIcs*, pages 8:1–8:19. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.
21. Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, 2001.

22 Antti Hyttinen, Sergey M. Plis, Matti Järvisalo, Frederick Eberhardt, and David Danks. A constraint optimization approach to causal discovery from subsampled time series data. *Int. J. Approx. Reason.*, 90:208–225, 2017.

23 Daniel Jackson, Ian Schechter, and Ilya Shlyakhter. Alcoa: the alloy constraint analyzer. In *ICSE*, pages 730–733. ACM, 2000.

24 Rajesh Jayaram and David P. Woodruff. Perfect lp sampling in a data stream. In *FOCS*, pages 544–555. IEEE Computer Society, 2018.

25 Rajesh Jayaram, David P. Woodruff, and Samson Zhou. Truly perfect samplers for data streams and sliding windows. *CoRR*, abs/2108.12017, 2021.

26 Hossein Jowhari, Mert Saglam, and Gábor Tardos. Tight bounds for lp samplers, finding duplicates in streams, and related problems. In *PODS*, pages 49–58. ACM, 2011.

27 Michael Kapralov and Dmitry Krachun. An optimal space lower bound for approximating MAX-CUT. In *STOC*, pages 277–288. ACM, 2019.

28 Rajeev Kohli, Ramesh Krishnamurti, and Prakash Mirchandani. The minimum satisfiability problem. *SIAM J. Discret. Math.*, 7(2):275–283, 1994.

29 Rajeev Kohli, Ramesh Krishnamurti, and Prakash Mirchandani. The minimum satisfiability problem. *SIAM J. Discret. Math.*, 7(2):275–283, 1994.

30 Ilan Kremer, Noam Nisan, and Dana Ron. On randomized one-round communication complexity. *Comput. Complex.*, 8(1):21–49, 1999.

31 Chu Min Li, Zhu Zhu, Felip Manyà, and Laurent Simon. Minimum satisfiability and its applications. In *IJCAI*, pages 605–610. IJCAI/AAAI, 2011.

32 Inês Lynce and João Marques-Silva. Efficient haplotype inference with boolean satisfiability. In *AAAI*, pages 104–109. AAAI Press, 2006.

33 Madhav V. Marathe and S. S. Ravi. On approximation algorithms for the minimum satisfiability problem. *Inf. Process. Lett.*, 58(1):23–29, 1996.

34 Joao Marques-Silva. Practical applications of boolean satisfiability. In *2008 9th International Workshop on Discrete Event Systems*, pages 74–80. IEEE, 2008.

35 João Marques-Silva, Mikolás Janota, Alexey Ignatiev, and António Morgado. Efficient model based diagnosis with maximum satisfiability. In *IJCAI*, pages 1966–1972. AAAI Press, 2015.

36 Andrew McGregor, David Tench, Sofya Vorotnikova, and Hoa T. Vu. Densest subgraph in dynamic graph streams. *CoRR*, abs/1506.04417, 2015.

37 Andrew McGregor, David Tench, Sofya Vorotnikova, and Hoa T. Vu. Densest subgraph in dynamic graph streams. In *MFCS (2)*, volume 9235 of *Lecture Notes in Computer Science*, pages 472–482. Springer, 2015.

38 Matthias Poloczek, Georg Schnitger, David P. Williamson, and Anke van Zuylen. Greedy algorithms for the maximum satisfiability problem: Simple algorithms and inapproximability bounds. *SIAM J. Comput.*, 46(3):1029–1061, 2017.

39 Xujie Si, Xin Zhang, Radu Grigore, and Mayur Naik. Maximum satisfiability in software analysis: Applications and techniques. In *CAV (1)*, volume 10426 of *Lecture Notes in Computer Science*, pages 68–94. Springer, 2017.

40 Boris A. Trakhtenbrot. A survey of russian approaches to perebor (brute-force searches) algorithms. *IEEE Annals of the History of Computing*, 6(4):384–400, 1984.

41 Luca Trevisan. Parallel approximation algorithms by positive linear programming. *Algorithmica*, 21(1):72–88, 1998.

42 Jeffrey Scott Vitter. Random sampling with a reservoir. *ACM Trans. Math. Softw.*, 11(1):37–57, 1985.

43 Mihalis Yannakakis. On the approximation of maximum satisfiability. *J. Algorithms*, 17(3):475–502, 1994.