The 2CNF Boolean Formula Satisfiability Problem and the Linear Space Hypothesis[*]

Tomoyuki Yamakami

Faculty of Engineering, University of Fukui, Japan
`TomoyukiYamakami@gmail.com`

Abstract

We aim at investigating the solvability/insolvability of nondeterministic logarithmic-space (NL) decision, search, and optimization problems parameterized by size parameters using simultaneously polynomial time and sub-linear space on multi-tape deterministic Turing machines. We are particularly focused on a special NL-complete problem, 2SAT – the 2CNF Boolean formula satisfiability problem – parameterized by the number of Boolean variables. It is shown that 2SAT with n variables and m clauses can be solved simultaneously polynomial time and $(n/2^{c\sqrt{\log n}})\, polylog(m+n)$ space for an absolute constant $c > 0$. This fact inspires us to propose a new, practical working hypothesis, called the linear space hypothesis (LSH), which states that $2\mathrm{SAT}_3$ – a restricted variant of 2SAT in which each variable of a given 2CNF formula appears as literals in at most 3 clauses – cannot be solved simultaneously in polynomial time using strictly "sub-linear" (i.e., $n^{\varepsilon}\, polylog(n)$ for a certain constant $\varepsilon \in (0,1)$) space. An immediate consequence of this working hypothesis is $\mathrm{L} \neq \mathrm{NL}$. Moreover, we use our hypothesis as a plausible basis to lead to the insolvability of various NL search problems as well as the nonapproximability of NL optimization problems. For our investigation, since standard logarithmic-space reductions may no longer preserve polynomial-time sub-linear-space complexity, we need to introduce a new, practical notion of "short reduction." It turns out that $\overline{2\mathrm{SAT}_3}$ is complete for a restricted version of NL, called Syntactic NL or simply SNL, under such short reductions. This fact supports the legitimacy of our working hypothesis.

1998 ACM Subject Classification F.1.1 Models of Computation, F.1.2 Modes of Computation, F.1.3 Complexity Measures and Classes, F.4.3 Formal Languages, G.1.2 Approximation, G.1.6 Optimization

Keywords and phrases sub-linear space, linear space hypothesis, short reduction, Boolean formula satisfiability problem, NL search, NL optimization, Syntactic NL

Digital Object Identifier 10.4230/LIPIcs.MFCS.2017.62

1 Background and Main Contributions

1.1 Motivational Discussion: Space Complexity of Parameterized 2SAT

Since Cook [4] demonstrated its NP-completeness, the Boolean formula satisfiability problem (SAT) of determining whether a given Boolean formula is satisfied by a suitably-chosen variable assignment has been studied extensively for more than 50 years. As its restricted variant, the *kCNF Boolean formula satisfiability problem* (kSAT), for an integer index $k \geq 3$, whose input formulas are of k-conjunctive normal form (kCNF) has also been a centerpiece of computational complexity theory. Since kSAT is complete for NP (nondeterministic

[*] This work was done in part at the University of Toronto between August 1, 2016 and March 30, 2017 and was supported by the Natural Sciences and Engineering Research Council of Canada.

© Tomoyuki Yamakami;
licensed under Creative Commons License CC-BY

polynomial time) [4], its solvability is linked to the computational complexity of all other NP problems; for instance, if kSAT is solved in polynomial time, then so are all NP problems. A recent study has been focused on the solvability of kSAT with n Boolean variables and m clauses within "sub-exponential" (which means $2^{\varepsilon n} poly(n+m)$ for an absolute constant $\varepsilon \in (0,1)$ and a suitable polynomial $poly(\cdot)$) runtime. In this line of study, Impagliazzo, Paturi, and Zane [10] took a new approach toward kSAT and its search version, Search-kSAT, parameterized by the number $m_{vbl}(x)$ of Boolean variables and the number $m_{cls}(x)$ of clauses in a given kCNF formula x as natural "size parameters" (which were called "complexity parameters" in [10]). To discuss such sub-exponential-time solvability for a wide range of NP-complete problems, Impagliazzo et al. further devised a crucial notion of *sub-exponential-time reduction family (or SERF-reduction)*, which preserves the sub-exponential-time complexity, and they cleverly demonstrated that the two size parameters, $m_{vbl}(x)$ and $m_{cls}(x)$, make Search-kSAT SERF-equivalent (that is, the both are SERF-reducible to each other). As a working hypothesis, Impagliazzo and Paturi [9] formally proposed the *exponential time hypothesis* (ETH), which asserts the insolvability of kSAT parameterized by $m_{vbl}(x)$ (succinctly denoted by $(k\mathrm{SAT}, m_{vbl})$) in sub-exponential time for all indices $k \geq 3$. Their hypothesis is obviously a stronger assertion than $\mathrm{P} \neq \mathrm{NP}$ and it has then led to intriguing consequences, including finer lower bounds on the solvability of various parameterized NP problems (see, e.g., a survey [14]).

Whereas ETH concerns with kSAT for $k \geq 3$, we are focused on the remaining case of $k = 2$. The decision problem 2SAT is known to be complete[*] for NL (nondeterministic logarithmic space) under log-space reductions. Since 2SAT already enjoys a polynomial-time algorithm (because $\mathrm{NL} \subseteq \mathrm{P}$), we are more concerned with how much memory space such an algorithm requires to run. An elaborate algorithm solves 2SAT with n variables and m clauses using simultaneously polynomial time and $(n/2^{c\sqrt{\log n}})\, polylog(m+n)$ space (Theorem 4.2), where $c > 0$ is a constant and $polylog(\cdot)$ is a suitable polylogarithmic function. This space bound is slightly below n; however, it is not yet known that 2SAT parameterized by $m_{vbl}(x)$ (or $m_{cls}(x)$) can be solved in polynomial time using strictly "sub-linear" space. Here, the informal term "sub-linear" for a size parameter $m(x)$ refers to a function of the form $m(x)^{\varepsilon}\ell(|x|)$ on input instances x for a certain absolute constant $\varepsilon \in (0,1)$ and an appropriately-chosen polylogarithmic function $\ell(n)$. Of course, this multiplicative factor $\ell(|x|)$ becomes redundant if $m(x)$ is relatively large (for example, $m(x) \geq \log^k |x|$ for any constant $k > 0$) and thus "sub-linear" turns out to be simply $m(x)^{\varepsilon}$.

In parallel to a restriction of SAT to kSAT, for polynomial-time sub-linear-space solvability, we further limit 2SAT to $2\mathrm{SAT}_k$, which consists of all satisfiable formulas in which each variable appears as literals in at most k clauses. Notice that $2\mathrm{SAT}_k$ for each $k \geq 3$ is also NL-complete (Proposition 4.1) as 2SAT is; in contrast, $k\mathrm{SAT}_2$ already falls into L for any index $k \geq 2$.

1.2 Sub-Linear Space and Short Reductions

All (parameterized) decision problems solvable in polynomial time using sub-linear space form a new complexity class PsubLIN (whose prefix "P" refers to "polynomial time"), which is located between L and P. This class PsubLIN naturally includes, for example, DCFL (deterministic context-free) because Cook [5] earlier showed that every language in DCFL is

[*] This is because Jones, Lien, and Laaser [13] demonstrated the NL-completeness of the complement of SAT (called UNSAT_2 in [13]) and Immerman [8] and Szelepcsényi [18] proved the closure of NL under complementation.

recognized in polynomial time using $O(\log^2 n)$-space, where n is input size. Unfortunately, there is no separation known among L, NL, PsubLIN, and P.

It turns out that PsubLIN does not seem to be closed under standard log-space reductions; thus, those reductions are no longer suitable tools to discuss the solvability of NL-complete problems in polynomial time using sub-linear space. Therefore, we need to introduce a much weaker form of reductions, called *short reductions*, which preserve polynomial-time, sub-linear-space complexity. Intuitively speaking, a short reduction is a reduction between two (parameterized) decision problems computed by a reduction machine (or a reduction function) that can generate strings of size parameter proportional to or less than size parameter of its input string. In particular, we will define three types of such short reductions in Section 3: short L-m-reducibility ($\leq_m^{\mathrm{sL}}$), short L-T-reducibility ($\leq_T^{\mathrm{sL}}$), and short sub-linear-space-T-reducibility ($\leq_T^{\mathrm{sSLRF}}$).

As noted earlier, Impagliazzo et al. demonstrated in [10, Corollary 2] that $(k\mathrm{SAT}, m_{vbl})$ is SERF-equivalent to $(k\mathrm{SAT}, m_{cls})$. Similarly, we can give a short reduction from $2\mathrm{SAT}_3$ with m_{vbl} to $2\mathrm{SAT}_3$ with m_{cls}, and vice verse; in other words, $(2\mathrm{SAT}_3, m_{vbl})$ and $(2\mathrm{SAT}_3, m_{cls})$ are equivalent under short L-T-reductions (Lemma 4.3(2)). On the contrary, such equivalence is not known for 2SAT and this circumstance signifies the importance of $2\mathrm{SAT}_3$.

Another importance of $2\mathrm{SAT}_3$ can be demonstrated by showing that $2\mathrm{SAT}_3$ is actually one of the hardest problems in a natural subclass of NL, which we call *Syntactic NL* or simply *SNL*. An *SNL formula* $\Psi \equiv \Psi(x)$ is of the form $\exists T \forall i_1 \cdots \forall i_r \forall y_1 \cdots \forall y_s \exists z_1 \cdots \exists z_t\ \psi$, starting with a second-order existential quantifier, followed by first-order quantifiers, with a supporting *semantic model*. From this model, we define a *certificate size* $m_{cert}(x)$. Their precise definitions will be given in Section 4. We say that Ψ *syntactically expresses* A if, for every x, $x \in A$ exactly when $\Psi(x)$ is true. The notation SNL stands for the collection of all (A, m), each of which is expressed syntactically by an appropriate SNL-formula Ψ and satisfies $m(x) = cm_{cert}(x)$ for a certain constant $c > 0$.

▶ **Theorem 1.1.** $(\overline{2\mathrm{SAT}_3}, m_{vbl})$ *is complete for* SNL *under short SLRF-T-reductions.*

1.3 A New, Practical Working Hypothesis for $2\mathrm{SAT}_3$

Since its introduction in 2001, ETH for kSAT ($k \geq 3$) has served as a driving force to obtain finer lower bounds on the sub-exponential-time computability of various parameterized NP problems, since those bounds do not seem to be obtained directly from the popular assumption of P $\neq$ NP. In a similar vein, we wish to propose a new working hypothesis, called the *linear space hypothesis* (LSH) for $2\mathrm{SAT}_3$, in which no deterministic algorithm solves $(2\mathrm{SAT}_3, m_{vbl})$ simultaneously in polynomial time using sub-linear space. More precisely:

> The Linear Space Hypothesis (LSH) for $2\mathrm{SAT}_3$: For any choice of $\varepsilon \in (0,1)$ and any polylogarithmic function ℓ, no deterministic Turing machine solves $2\mathrm{SAT}_3$ parameterized by m_{vbl} simultaneously in polynomial time using $m_{vbl}(x)^{\varepsilon}\ell(|x|)$ space, where x refers to an input instance to $2\mathrm{SAT}_3$.

We can replace m_{vbl} in the above definition by m_{cls} (see Section 4), and thus we often omit it. Consider the case of L = NL. Since $2\mathrm{SAT}_3$ belongs to L, it is also in PsubLIN. This consequence contradicts LSH for $2\mathrm{SAT}_3$. Therefore, we immediately obtain:

▶ **Theorem 1.2.** *If LSH for* $2\mathrm{SAT}_3$ *is true, then* L $\neq$ NL.

From Theorem 1.2, our working hypothesis LSH for $2\mathrm{SAT}_3$ is expected to lead to finer, better consequences than what the assumption L $\neq$ NL can lead to.

Let δ_3 denote the infimum of a real number $\varepsilon \in [0,1]$ for which there is a deterministic Turing machine solving $2\mathrm{SAT}_3$ simultaneously in polynomial time using at most $m_{vbl}(x)^{\varepsilon}\ell(|x|)$ space on instances x for a certain fixed polylogarithmic function ℓ. Here, we acknowledge three possible cases: (i) $\delta_3 = 0$, (ii) $0 < \delta_3 < 1$, and (iii) $\delta_3 = 1$, and one of them must be true after all. The hypothesis LSH for $2\mathrm{SAT}_3$ exactly matches (iii).

▶ **Proposition 1.3.** *The working hypothesis LSH for* $2\mathrm{SAT}_3$ *is true iff* $\delta_3 = 1$ *holds.*

For any $\leq_r$-reduction, the notation $\leq_r(\mathrm{SNL})$ refers to the collection of all (parameterized) decision problems that can be reduced by $\leq_r$-reductions to certain problems in SNL.

▶ **Proposition 1.4.** *The following statements are all logically equivalent. (1)* $(2\mathrm{SAT}_3, m_{vbl}) \in \mathrm{PsubLIN}$*. (2)* $\mathrm{SNL} \subseteq \mathrm{PsubLIN}$*. (3)* $\leq_T^{\mathrm{sSLRF}}(\mathrm{SNL}) \subseteq \mathrm{PsubLIN}$*.*

Proposition 1.4(3) can be compared to the fact that $\leq_m^{\mathrm{L}}(\mathrm{SNL}) = \mathrm{NL}$.

Furthermore, we seek two other characterizations of the hypothesis LSH for $2\mathrm{SAT}_3$. The first problem is a variant of a well-known NP-complete problem, called the $\{0,1\}$-*linear programming problem* (LP_2). In what follows, a vector of dimension n means an $n \times 1$ matrix and a rational number is treated as a pair of appropriate integers.

(2,k)-Entry $\{0,1\}$-Linear Programming Problem ($\mathrm{LP}_{2,k}$):

- Instance: a rational $m \times n$ matrix A and a rational vector b of dimension n, where $m, n \geq 1$ and each row of A has at most two nonzero entries and each column of A has at most k non-zero entries.
- Question: is there any $\{0,1\}$-vector x satisfying $Ax \geq b$?

As natural size parameters $m_{col}(x)$ and $m_{row}(x)$, we take the numbers of columns and of rows of A for instance $x = (A, b)$ given to $\mathrm{LP}_{2,k}$, respectively.

Another problem to consider is a variant of the *directed s-t connectivity problem*[†] (DSTCON) of asking whether a path between two given vertices exists in a directed graph.

Degree-k Directed s-t Connectivity Problem (kDSTCON):

- Instance: a directed graph $G = (V, E)$ of degree (i.e., indegree plus outdegree) at most k, and two designated vertices s and t.
- Question: is there any path from s to t in G?

For any instance $x = (G, s, t)$ to kDSTCON, $m_{ver}(x)$ and $m_{edg}(x)$ respectively denote the number of vertices and that of edges in G.

▶ **Theorem 1.5.** *The following statements are logically equivalent: (1) LSH for* $2\mathrm{SAT}_3$*, (2) LSH for* $\mathrm{LP}_{2,3}$ *(with* m_{row} *or* m_{col}*), and (3) LSH for* 3DSTCON *(with* m_{ver} *or* m_{edg}*).*

This theorem allows us to use $\mathrm{LP}_{2,3}$ and 3DSTCON for LSH as substitutes for $2\mathrm{SAT}_3$.

1.4 Four Examples of How to Apply the Working Hypothesis

To demonstrate the usefulness of LSH for $2\mathrm{SAT}_3$, we will seek four applications of LSH in the fields of search problems and optimization problems. Although many NL decision problems have been turned into *NL search problems* (whose precise definition is given in Section 6), not all NL problems can be "straightforwardly" converted into a framework of NL search problems. For example, 2SAT is NL-complete but the problem of finding a truth assignment (when

[†] This is also known as the graph accessibility problem and the graph reachability problem in the literature.

variables are ordered in an arbitrarily fixed way) that satisfies a given 2CNF formula does not look like a legitimate form of NL search problem. In addition, its optimization version, Max2SAT, is already complete for APX (polynomial-time approximable NP optimization) instead of NLO (NL optimization class) under polynomial-time approximation-preserving reductions (see [1]).

First, we will see two simple applications of LSH for $2\mathrm{SAT}_3$ in the area of NL search problems. Earlier, Jones et al. [13] discussed the NL-completeness of a decision problem concerning *one-way nondeterministic finite automata* (or 1nfa's). We modify this problem into an associated search problem, called Search-1NFA, as given below.

1NFA MEMBERSHIP SEARCH PROBLEM (SEARCH-1NFA):
- INSTANCE: a 1nfa $M = (Q, \Sigma, \delta, q_0, F)$ with no λ-moves, and a parameter 1^n, where λ is the empty string for $n \in \mathbb{N}$.
- SOLUTION: an input string x of length n accepted by M (i.e., when x is written on M's read-only input tape, M eventually enters a final state in F before or on reading the last symbol of x).

As a meaningful size parameter m_{nfa}, we set $m_{nfa}(x) = |Q||\Sigma|n$ for instance $x = (M, 1^n)$.

▶ **Theorem 1.6.** *Assuming that LSH for* $2\mathrm{SAT}_3$*, for every fixed value* $\varepsilon \in (0, 1/2)$*, there is no polynomial-time* $O(n^{1/2-\varepsilon})$*-space algorithm for* $(\text{Search-1NFA}, m_{nfa})$.

Jenner [11] presented a few variants of the well-known *knapsack problem* and showed their NL-completeness. Here, we choose one of them that fit into the NL-search framework by a small modification. Given a string x, a substring z of x is called *unique* if there exists a unique pair u, v satisfying $x = uzv$. Write $[n]$ for the set $\{1, 2, \ldots, n\}$.

UNIQUE ORDERED CONCATENATION KNAPSACK SEARCH PROBLEM (SEARCH-UOCK):
- INSTANCE: a string w and a sequence $(w_1, w_2, \ldots, w_n)$ of strings over a certain fixed alphabet Σ such that, for every $i \in [n]$, if w_i is a substring of w, then w_i is unique.
- SOLUTION: a sequence $(i_1, i_2, \ldots, i_k)$ of indices with $k \geq 1$ such that $1 \leq i_1 < i_2 < \cdots < i_k \leq n$ and $w = w_{i_1} w_{i_2} \cdots w_{i_k}$.

Our size parameter m_{elm} for Search-UOCK is the number of elements $w_1, w_2, \ldots, w_n$ in the above definition (namely, $m_{elm}(x) = n$ for instance x).

▶ **Theorem 1.7.** *If LSH for* $2\mathrm{SAT}_3$ *holds, then, for any* $\varepsilon > 0$*, there is no polynomial-time* $O(n^{1/2-\varepsilon})$*-space algorithm for* $(\text{Search-UOCK}, m_{elm})$.

We then turn to the area of *NL optimization problems* (or *NLO problems*, in short) [19, 20]. See Section 6 for their formal definition. We will consider a problem that belongs to $\mathrm{LSAS}_{\mathrm{NLO}}$ but does not seem to be solvable using log space. Here, $\mathrm{LSAS}_{\mathrm{NLO}}$ is the collection of NLO problems that have log-space approximation schemes, where a *log-space approximation scheme* for an NLO problem P is a deterministic Turing machine M that takes any input of the form (x, k) and outputs a solution y of P using space at most $f(k) \log |x|$ for a certain log-space computable function $f : \mathbb{N} \to \mathbb{N}$ for which the performance ratio R satisfies $R(x, y) \leq 1 + \frac{1}{k}$. Such a solution y is called a $(1 + \frac{1}{k})$*-approximate solution*. Notice that the *performance ratio* is a ratio between the value of an optimal solution and that of M's output.

In 2007, Tantau [19] presented an NL maximization problem, called Max-HPP, which falls into $\mathrm{LSAS}_{\mathrm{NLO}}$. This problem was later rephrased in [20, arXiv version] in terms of complete graphs and it was shown to be computationally hard for $\mathrm{LO}_{\mathrm{NLO}}$ (log-space computable NL optimization) under *approximation-preserving exact* NC^1*-reduction*.

MAXIMUM HOT POTATO PROBLEM (Max-HPP):

- INSTANCE: an $n \times n$ matrix A whose entries are drawn from $[n]$, a number $d \in [n]$, and a start index $i_1 \in [n]$, where $n \in \mathbb{N}^+$.
- SOLUTION: an index sequence $\mathcal{S} = (i_1, i_2, \ldots, i_d)$ of length d with $i_j \in [n]$ for any $j \in [d]$.
- MEASURE: total weight $w(\mathcal{S}) = \sum_{j=1}^{d-1} A_{i_j i_{j+1}}$.

We use the number n of columns in a given matrix as size parameter $m_{col}(A, d, i_1)$. We can show that, under the assumption of LSH for 2SAT$_3$, (Max-HPP$, m_{col}$) cannot have polynomial-time $O(k^{1/3} \log m_{col}(x))$-space approximation schemes of finding $(1 + \frac{1}{k})$-approximate solutions for instances x.

▶ **Theorem 1.8.** *If LSH for* 2SAT$_3$ *is true, then, for any* $\varepsilon > 0$*, there is no polynomial-time* $O(k^{1/3} \log m_{col}(x))$*-space algorithm finding* $(1+\frac{1}{k})$*-approximate solutions of* (Max-HPP$, m_{col}$)*, where* x *is any instance and* k *is an approximation parameter.*

The fourth example concerns with the computational complexity of transforming one type of finite automata into another type. It is known that we can convert a 1nfa M to an "equivalent" *one-way deterministic finite automaton* (or 1dfa) M' in the sense that both M and M' recognize exactly the same language. In particular, we consider the case of transforming an n-state *unary* 1nfa into its equivalent *unary* 1dfa, where a unary finite automaton takes a single-letter input alphabet. A standard procedure of such transformation requires polynomial-time and $O(n)$ space (cf. [7]). Under LSH for 2SAT$_3$, we can demonstrate that this space bound cannot be made significantly smaller.

▶ **Theorem 1.9.** *If LSH for* 2SAT$_3$ *is true, then, for any constant* $\varepsilon \in (0, 1)$*, there is no polynomial-time* $O(n^{\varepsilon})$*-space algorithm that takes an* n*-state unary 1nfa as input and produces an equivalent unary 1dfa of* $O(n \log n)$ *states.*

2 Basic Notions and Notation

Let $\mathbb{N}$ be the set of *natural numbers* (i.e., nonnegative integers) and set $\mathbb{N}^+ = \mathbb{N} - \{0\}$. Two notations $\mathbb{R}$ and $\mathbb{R}^{\geq 0}$ denote respectively the set of all *real numbers* and that of all *nonnegative real numbers*. For any two integers m and n with $m \leq n$, the notation $[m, n]_{\mathbb{Z}}$ denotes the set $\{m, m+1, m+2, \ldots, n\}$, which is an *integer interval* between m and n. For simplicity, when $n \geq 1$, we write $[n]$ for $[1, n]_{\mathbb{Z}}$.

In this paper, all *polynomials* are assumed to have nonnegative integer coefficients. All *logarithms* are to base 2. A *polylogarithmic (or polylog) function* ℓ is a function mapping $\mathbb{N}$ to $\mathbb{R}^{\geq 0}$ such that there exists a polynomial p for which $\ell(n) = p(\log n)$ holds for all $n \in \mathbb{N}$, provided that "$\log 0$" is conventionally set to be 0.

In a course of our study on polynomial-time sub-linear-space computability, it is convenient to expand the standard framework of decision problems to problems *parameterized* by properly chosen "size parameters" (called "complexity parameters" in [10]), which serve as a basis unit of the time/space complexity of an algorithm. In this respect, we follow a framework of Impagliazzo et al. [10] to work with a flexible choice of size parameter. A standard size parameter is the total length $|x|$ of the binary representation of an input instance x and it is often denoted by $||$. More generally, a *(log-space) size parameter* $m(x)$ for a problem P is a function mapping Σ^* (where Σ is an input alphabet) to $\mathbb{N}$ such that (1) m must be computed using log space (that is, by a certain Turing machine that takes input x and outputs $m(x)$ in unary on an output tape using at most $c \log |x| + d$ space for certain constants $c, d > 0$) and (2) there exists a polynomial p satisfying $m(x) \leq p(|x|)$ for all instances x of P.

As key examples, for any graph-related problem (such as 3DSTCON), $m_{edg}(x)$ and $m_{ver}(x)$ denote respectively the total number of edges and that of vertices in a given graph instance x. Clearly, m_{ver} and m_{edg} are log-space computable. To emphasize the use of size parameter m, we often write (P, m) in place of P. We say that a multi-tape Turing machine M uses *logarithmic space* (or *log space*, in short) with respect to size parameter m if there exist two absolute constants $c, d \geq 0$ such that each of the work tapes (not including input and output tapes) used by M on x are upper-bounded by $c \log m(x) + d$ on every input x.

Two specific notations L and NL respectively stand for the classes of all decision problems solvable on multi-tape deterministic and nondeterministic Turing machines using log space. It is known that the additional requirement of "polynomial runtime" does not change these classes. More generally, PTIME,SPACE($s(n)$) expresses a class composed of all (parameterized) decision problems (P, m) solvable deterministically in polynomial time (in $|x|$) using space at most $s(m(x))$ on any instance x given to P.

To define NL search and optimization problems in Section 6, it is convenient for us to use a practical notion of "auxiliary Turing machine" (see, e.g., [20]). An *auxiliary Turing machine* is a multi-tape deterministic Turing machine equipped with an extra read-only *auxiliary input tape*, in which a tape head scans each auxiliary input symbol only once by moving from the left to the right. Given two alphabets Σ and Γ, a (parameterized) decision problem (P, m) with $P \subseteq \Sigma^* \times \Gamma^*$ is in auxL if there exist a polynomial p and an auxiliary Turing machine M that takes a standard input x and an auxiliary input y of length $p(|x|)$ and decides whether M accepts (x, y) or not in time polynomial in $|x|$ using space logarithmic in $m(x)$. Its functional version is denoted by auxFL, provided that each underlying Turing machine is equipped with an extra *write-only* output tape (in which a tape head moves to the right whenever it writes a non-blank output symbol) and that the machine produces output strings of at most polynomial length.

3 Sub-Linear Space and Short Reductions

Recall from [9, 10] that the term "sub-exponential" means $2^{\varepsilon m(x)} \mathit{poly}(|x|)$ for a certain constant $\varepsilon \in (0, 1)$. In contrast, our main subject is polynomial-time, sub-linear-space computability, where the term "sub-linear" refers to functions of the form $m(x)^{\varepsilon} \mathit{polylog}(|x|)$ on input instances x for a certain constant $\varepsilon \in (0, 1)$ and a certain polylogarithmic function $\mathit{polylog}(n)$. As noted in Section 1.2, the multiplicative factor $\mathit{polylog}(|x|)$ can be eliminated whenever $m(x)$ is relatively large.

First, we will provide basic definitions for (parameterized) decision problems. A decision problem P parameterized by size parameter m is said to be *solvable in polynomial time using sub-linear space* if, for a certain choice of constant $\varepsilon \in (0, 1)$, there exist a deterministic Turing machine M_{ε}, a polynomial p_{ε}, and a polylogarithmic function ℓ_{ε} for which M solves P simultaneously in at most $p_{\varepsilon}(|x|)$ steps using space at most $m(x)^{\varepsilon} \ell_{\varepsilon}(|x|)$ for all instances x given to P.

The notation PsubLIN expresses the collection of all (parameterized) decision problems (P, m) that are solvable in polynomial time using sub-linear space. In other words, PsubLIN $= \bigcup_{\varepsilon \in (0,1)}$ PTIME,SPACE($m(x)^{\varepsilon} \ell(|x|)$) for input instances x, where m refers to an arbitrary (log-space) size parameter and ℓ refers to any polylogarithmic function. It thus follows that L $\subseteq$ PsubLIN $\subseteq$ P but none of these inclusions is known to be proper.

The notion of reducibility among decision problems is quite useful in measuring the relative complexity of the problems. For the class PsubLIN, in particular, we need a restricted form of reducibility, which we call "short" reducibility, satisfying a special property that any outcome of the reduction is linearly upper-bounded in size by an input of the reduction. We will define such restricted reductions for (parameterized) decision problems of our interest.

We begin with a description of *L-m-reducibility* for (parameterized) decision problems. Given two (parameterized) decision problems (P_1, m_1) and (P_2, m_2), we say that (P_1, m_1) is *L-m-reducible to* (P_2, m_2), denoted by $(P_1, m_1) \leq_m^{\mathrm{L}} (P_2, m_2)$, if there is a function $(f, ||) \in \mathrm{FL}$ (where $||$ refers to the bit length) and two constants $k_1, k_2 > 0$ such that, for any input string x, (i) $x \in P_1$ iff $f(x) \in P_2$ and (iii) $m_2(f(x)) \leq m_1(x)^{k_1} + k_1$. Notice that all functions in FL are, by their definition, polynomially bounded.

Concerning polynomial-time sub-linear-space solvability, we introduce a restricted variant of this L-m-reducibility, which we call the *short L-m-reducibility* (or sL-m-reducibility, in short), obtained by replacing the equality $m_2(f(x)) \leq m_1(x)^{k_1} + k_1$ in the above definition of $\leq_m^{\mathrm{L}}$ with $m_2(f(x)) \leq k_1 m_1(x) + k_1$. To express this new reducibility, we use a new notation of $\leq_m^{\mathrm{sL}}$.

Since many-one reducibility is too restrictive to use, we need a stronger notion of Turing reduction, which fits into a framework of polynomial-time, sub-linear-space computability. Our reduction is actually a *polynomial-time sub-linear-space reduction family* (SLRF, in short), performed by oracle Turing machines. A (parameterized) decision problem (P_1, m_1) is *SLRF-T-reducible to* another one (P_2, m_2), denoted by $(P_1, m_1) \leq_T^{\mathrm{SLRF}} (P_2, m_2)$, if, for every fixed value $\varepsilon > 0$, there exist an oracle Turing machine M_ε equipped with an extra write-only query tape, a polynomial p_ε, a polylog function ℓ_ε, and three constants $k_1, k_2 \geq 1$ such that, for every instance x to P_1, (1) $M_\varepsilon^{P_2}$ runs in at most $p_\varepsilon(|x|)$ time using at most $m_1(x)^\varepsilon \ell_\varepsilon(|x|)$ space, provided that its query tape is not subject to this space bound, (2) if $M_\varepsilon^{P_2}(x)$ makes a query to P_2 with query word z written on the query tape, then z satisfies both $m_2(z) \leq m_1(x)^{k_1} + k_1$ and $|z| \leq |x|^{k_2} + k_2$, and (3) after M_ε makes a query, in a single step, it automatically erases its query tape, it returns its tape head back to the initial cell, and oracle P_2 informs the machine of its answer by changing the machine's inner state.

The *short SLRF-T-reducibility* (or sSLRF-T-reducibility, in short) is obtained from the SLRF-reducibility by substituting $m_2(z) \leq k_1 m_1(x) + k_1$ for the above inequality $m_2(z) \leq m_1(x)^{k_1} + k_1$. The notation $\leq_T^{\mathrm{sSLRF}}$ denotes this restricted reducibility. In the case where M_ε is limited to log-space usage, we use a different notation of $\leq_T^{\mathrm{sL}}$. Note that any $\leq_T^{\mathrm{sSLRF}}$-reduction is an $\leq_T^{\mathrm{SLRF}}$-reduction but the converse is not true because there is a pair of problems reducible by $\leq_T^{\mathrm{SLRF}}$-reductions but not by $\leq_T^{\mathrm{sSLRF}}$-reductions.

For any reduction $\leq_r$, a decision problem P is said to be $\leq_r$*-complete* for a given class $\mathcal{C}$ of problems if (1) $P \in \mathcal{C}$ and (2) every problem Q in $\mathcal{C}$ is $\leq_r$-reducible to P. We use the notation $\leq_r(\mathcal{C})$ to express the collection of all problems that are $\leq_r$-reducible to certain problems in $\mathcal{C}$. When $\mathcal{C}$ is a singleton, say, $\mathcal{C} = \{A\}$, we write $\leq_r(A)$ instead of $\leq_r(\{A\})$.

It follows that $(P_1, m_1) \leq_m^{\mathrm{L}} (P_2, m_2)$ implies $(P_1, m_1) \leq_T^{\mathrm{L}} (P_2, m_2)$, which further implies $(P_1, m_1) \leq_T^{\mathrm{SLRF}} (P_2, m_2)$. The same statement holds for $\leq_m^{\mathrm{sL}}$, $\leq_T^{\mathrm{sL}}$, and $\leq_T^{\mathrm{sSLRF}}$. Moreover, $(P_1, m_1) \leq_m^{\mathrm{sL}} (P_2, m_2)$ implies $(P_1, m_1) \leq_m^{\mathrm{L}} (P_2, m_2)$. The same holds for $\leq_T^{\mathrm{sSLRF}}$ and $\leq_T^{\mathrm{SLRF}}$.

Here are other basic properties of SLRF-T- and sSLRF-T-reductions.

▶ **Lemma 3.1.**

1. *The reducibilities* $\leq_T^{\mathrm{SLRF}}$ *and* $\leq_T^{\mathrm{sSLRF}}$ *are reflexive and transitive.*
2. *The class* PsubLIN *is closed under* $\leq_T^{\mathrm{sSLRF}}$*-reductions.*
3. *There exist recursive decision problems* X *and* Y *such that* $X \leq_T^{\mathrm{SLRF}} Y$ *but* $X \not\leq_T^{\mathrm{sSLRF}} Y$*. A similar statement holds also for* $\leq_m^{\mathrm{L}}$ *and* $\leq_m^{\mathrm{sL}}$*.*

4 The 2CNF Boolean Formula Satisfiability Problem and SNL

We will make a brief discussion on 2SAT (2CNF Boolean formulas satisfiability problem) and the complexity class SNL. As noted in Section 1.1, 2SAT is NL-complete under L-m-reductions.

In what follows, we are focused on two specific size parameters: $m_{vbl}(x)$ and $m_{cls}(x)$, which respectively denote the numbers of propositional variables and clauses appearing in formula-related instance x (not necessarily limited to instances of 2SAT).

We further restrict 2SAT by limiting the number of literals appearing in an input Boolean formula as follows. Let $k \in \mathbb{N}^+$. We denote by $2\mathrm{SAT}_k$ the collection of all formulas ϕ in 2SAT such that, for each variable v in ϕ, the number of occurrences of v and $\overline{v}$ is at most k. Since $2\mathrm{SAT}_1$ and $2\mathrm{SAT}_2$ are solvable using only log space, we force our attention on the case of $k \geq 3$. From $(2\mathrm{SAT}, ||) \leq^{\mathrm{L}}_{m} (2\mathrm{SAT}_3, ||)$ with a help of the fact that 2SAT is NL-complete, we can immediately obtain the following.

▶ **Proposition 4.1.** *For each index $k \geq 3$, $2\mathrm{SAT}_k$ is* NL*-complete.*

To solve 2SAT in polynomial time, we need slightly larger than sub-linear space.

▶ **Theorem 4.2.** *For a certain constant $c > 0$ and a polylog function $\ell(n)$,* 2SAT *with n variables and m clauses can be solved in polynomial time using $n^{1-c/\sqrt{\log n}}\ell(m+n)$ space.*

For any reduction $\leq_r$ defined in Section 3, we write $(P_1, m_1) \equiv_r (P_2, m_2)$ if both $(P_1, m_1) \leq_r (P_2, m_2)$ and $(P_2, m_2) \leq_r (P_1, m_1)$ hold.

▶ **Lemma 4.3.** *Let $m \in \{m_{vbl}, m_{cls}\}$ and $k \geq 3$. (1) $(2\mathrm{SAT}_k, m) \equiv^{\mathrm{sL}}_{m} (2\mathrm{SAT}_3, m)$ and (2) $(2\mathrm{SAT}_3, m_{vbl}) \equiv^{\mathrm{sL}}_{m} (2\mathrm{SAT}_3, m_{cls})$.*

Contrary to Lemma 4.3(2), it is still unknown whether $(2\mathrm{SAT}, m_{vbl}) \equiv^{\mathrm{sL}}_{T} (2\mathrm{SAT}, m_{cls})$.

Hereafter, we will define the notion of SNL formulas, which induce the complexity class SNL. Let $x = (S_1, \ldots, S_a, x_1, \ldots, x_b)$ be any instance, including "sets" S_i and "objects" x_j. An *SNL formula* Ψ is of the form $\exists T \forall i_1 \cdots \forall i_r \forall y_1 \cdots \forall y_s \exists z_1 \cdots \exists z_t\ \psi$, where ψ is a quantifier-free formula, which is a Boolean combination of *atomic formulas* of the following forms: $T(i, v)$, $(u_1, \ldots, u_k) \in S_j$, $u = v$, $i \leq j$, and $symb(v, i) = a$ (i.e., a is the ith symbol of v), where T is a second-order predicate symbol, and $i_1, \ldots, i_r, y_1, \ldots, y_s, z_1 \ldots, z_t$ are first-order variables, having the following *semantic model* for Ψ. In this model, T ranges over a subset of $[p(|x|)] \times U_x$ (where U_x is a universe) with $|U_x| \leq cm(x)$, each i_j ranges a number in $[p_j(|x|)]$, each y_j takes an element in another universe $U_{x,j}$ with $|U_{x,j}| \leq c_j m(x)$, and each z_j ranges over a set $Z_{x,j}$ of at most e elements (i.e., $|Z_{x_j}| \leq e$) for absolute constants $c, c_j, e \geq 1$ and polynomials p, p_j, not depending on the choice of x. A *certificate size* $m_{cert}(x)$ is defined to be $|U_x|$ as our basis size parameter.

As a quick example, let us consider a (parameterized) decision problem (A, m) such that there are a polynomial p, a constant $c > 0$, and a deterministic Turing machine M recognizing A simultaneously in time at most $p(|x|)$ using space at most $\log_{|\Gamma|} m(x) + c$ for every instance x to A, where Γ is a work-tape alphabet. We assume that M terminates in a configuration in which the work tape is blank and all tape heads return to the initial position. For our convenience, δ is extended to include a special transition from an accepting configuration to itself. To express (A, m), we define an SNL-formula $\Psi \equiv \Psi(x)$ as: $\exists T[Func(T) \wedge \exists v_0 \exists v_1 [T(1, v_0) \wedge T(last(T), v_1) \wedge v_1 \in ACC_x \wedge \forall i \forall v \exists w [T(i, v) \rightarrow (v, w) \in Tran_\delta \wedge T(i+1, w)]]]$ with a semantic model supporting $T \subseteq [p(|x|)] \times U_x$, $i \in [p(|x|)]$, $v_0, x_1, v, w, \in U_x$, where $U_x = \Gamma^{\log_{|\Gamma|} m(x)+c}$, ACC_x is the set of a unique accepting configuration, $last(T)$ indicates the largest index i that ensures $\exists v[T(i,v)]$, $Trans_\delta$ expresses a δ-transition between two configurations, and $Func(T)$ asserts that T represents a function $f(i) = z$ satisfying $T(i,z)$. Note that $|U_x| \leq |\Gamma|^{c+1} m(x)$. Hence, (A, m) belongs to SNL.

5 The Working Hypothesis LSH for 2SAT_3

The exponential time hypothesis (ETH) has served as a driving force to obtain better lower bounds on the computational complexity of various important problems (see, e.g., [14]).

In Theorem 4.2, we have seen that 2SAT with n variables and m clauses can be solved in polynomial time using $n^{1-c/\sqrt{\log n}} polylog(m+n)$ space for a certain constant $c > 0$; however, it is not yet known to be solved in polynomial time using sub-linear space. This circumstance encourages us to propose (in Section 1.3) a practical working hypothesis – the *linear space hypothesis* (LSH) for 2SAT_3 – which asserts the insolvability of $(2\text{SAT}_3, m_{ver})$ in polynomial time using sub-linear space. The choice of m_{vbl} does not matter; as shown in Lemma 4.3(2) with a help of Lemma 3.1(2), we can replace m_{vbl} in the definition of LSH by m_{cls}. Theorem 1.5 has further given two alternative definitions to LSH in terms of $\text{LP}_{2,3}$ and 3DSTCON.

As noted in Section 1.3, Theorem 1.2 states that the above working hypothesis leads to $\text{L} \neq \text{NL}$. Moreover, Proposition 1.3 asserts that LSH for 2SAT_3 is equivalent to $\delta_3 = 1$.

The working hypothesis LSH concerns with 2SAT_3 but it also carries over to 2SAT.

▶ **Lemma 5.1.** *Assuming that LSH for* 2SAT_3 *is true, each of the following statements holds: (1)* $\leq_T^{\text{sSLRF}}(2\text{SAT}_3, m_{vbl}) \nsubseteq \text{PsubLIN}$ *and (2)* $(2\text{SAT}, m_{vbl}) \notin \text{PsubLIN}$.

As another consequence of LSH for 2SAT_3, we can show the existence of a pair of problems in the class $\leq_T^{\text{sSLRF}}(2\text{SAT}_3, m_{vbl})$, which are incomparable with respect to $\leq_T^{\text{sSLRF}}$-reductions. This indicates that the class $\leq_T^{\text{sSLRF}}(2\text{SAT}_3, m_{vbl})$ has a fine, complex structure with respect to sSLRF-T-reducibility.

▶ **Theorem 5.2.** *Assuming LSH for* 2SAT_3*, there are two decision problems* (A, m_A) *and* (B, m_B) *in* $\leq_T^{\text{sSLRF}}(2\text{SAT}_3, m_{vbl})$ *such that* $(A, m_A) \not\leq_T^{\text{sSLRF}} (B, m_B)$ *and* $(B, m_B) \not\leq_T^{\text{sSLRF}} (A, m_A)$.

6 Proofs of the Four Examples of LSH Applications

In Section 1.4, we have described four examples of how to apply our working hypothesis LSH for 2SAT_3. Here, we will give three of their proofs.

First, we will briefly describe *(parameterized) NL search problems.* In general, a search problem parameterized by (log-space) size parameter m is expressed as (I, SOL, m), where I consists of (admissible) instances and SOL is a function from I to a set of strings (called a *solution space*) such that, for any $(x, y) \in I \circ SOL$, $y \in SOL(x)$ implies $|y| \leq am(x) + b$ for certain constants $a, b > 0$, where $I \circ SOL$ stands for $\{(x, y) \mid x \in I, y \in SOL(x)\}$. In particular, when we use the standard "bit length" of instances, we omit "$||$" and write (I, SOL) instead of $(I, SOL, ||)$. Of all search problems, *(parameterized) NL search problems* are (parameterized) search problems (I, SOL, m) for which $I \in \text{L}$ and $I \circ SOL \in \text{auxL}$. Finally, we denote by Search-NL the collection of all (parameterized) NL search problems.

We say that a deterministic Turing machine M *solves* (I, SOL, m) if, for any instance $x \in I$, M takes x as input and produces a solution in $SOL(x)$ if $SOL(x) \neq \emptyset$, and produces a designated symbol $\perp$ ("no solution") otherwise. Now, we recall from Section 1.4 a special

NL search problem, called Search-1NFA, in which we are asked to find an input of length n accepted by a given λ-free 1nfa M. Theorem 1.6 states that no polynomial-time $O(n^{1/2-\varepsilon})$-space algorithm solves (Search-1NFA, m_{nfa}).

Proof of Theorem 1.6. Toward a contradiction, we assume that (Search-1NFA, m_{nfa}) is solved by a deterministic Turing machine M in time polynomial in $|y|$ using space at most $cm_{nfa}(y)^{1/2-\varepsilon}$ on instances y, where $c, \varepsilon > 0$ are constants. Our aim is to show that (3DSTCON, m_{ver}) can be solved in polynomial time using sub-linear space, because this contradicts LSH for 3DSTCON, which is equivalent to LSH for 2SAT$_3$ by Theorem 1.5(3).

Let $x = (G, s, t)$ be any instance to 3DSTCON with $G = (V, E)$ and $s, t \in V$. Let $n = |V|$. Associated with this x, we define a 1nfa $N = (Q, \Sigma, \delta, q_0, F)$ as follows. First, let $Q = V$ and $\Sigma = [0, 3]_{\mathbb{Z}}$. Define $q_0 = s$ and $F = \{t\}$. For each $v \in V$, consider its neighbor $out(v) = \{w \in V \mid (v, w) \in E\}$. We assume that all elements in $out(v)$ are enumerated in a fixed linear order as $out(v) = \{w_1, w_2, \ldots, w_k\}$ with $0 \le k \le 3$. The transition function δ is defined as $\delta(v, i) = \{w_i\}$ if $0 \le i \le k$.

Supposedly, $\gamma = (v_1, v_2, \ldots, v_d)$ is a path from $s = v_1$ to $t = v_d$ in G. For each index $i \in [d]$, we choose an index $\ell(v_i)$ satisfying $v_{i+1} = w_{\ell(v_i)} \in out(v_i)$ and we then set $z = \ell(v_1)\ell(v_2)\cdots\ell(v_{d-1})0^{n-d+1}$. When N reads z, it eventually enters v_d, which is a halting state, and therefore N accepts z. On the contrary, in the case where there is no path from s to t in G, N never accepts any input. Therefore, it follows that (*) 3DSTCON has a path from s to t iff N accepts z.

Finally, we set $y = (N, 1^n)$ as an instance to Search-1NFA parameterized by m_{nfa}. Note that $m_{nfa}(y) = |Q||\Sigma|n \le 4|V|^2 = 4m_{ver}(z)^2$. By (*), 3DSTCON can be solved by running M on y in polynomial time; moreover, the space required for this computation is upper-bounded by $cm_{nfa}(y)^{1/2-\varepsilon} \le 2cm_{ver}(x)^{1-2\varepsilon}$, which is obviously sub-linear. ◀

Another NL search problem, Search-UOCK, asks to find, for a given string w, an index sequence $(i_1, \ldots, i_k)$ in increasing order that makes the concatenation $w_{i_1} \cdots w_{i_k}$ equal to w among $\{w_1, w_2, \ldots, w_n\}$. Here, we present the proof of Theorem 1.7.

Proof of Theorem 1.7. Let us assume that there is a polynomial-time $cm_{elm}(x)^{1/2-\varepsilon}$-space algorithm A for (Search-UOCK, m_{elm}) on instances x for certain constants $\varepsilon, c > 0$. We will use this A to solve (3DSTCON, m_{ver}) in polynomial time using sub-linear space.

Let $x = (G, s, t)$ be any instance to 3DSTCON with $G = (V, E)$. For simplicity of our argument, let $V = \{1, 2, \ldots, n\}$, $s = 1$, and $t = n$. Now, we define $\langle i, j\rangle = (i-1)n + j$ for each pair $i, j \in [n]$. First, we modify G into another graph $G' = (V', E')$, where $V' = \{\langle i, j\rangle \mid i, j \in [n]\}$ and $E' = \{(\langle i, j\rangle, \langle i', j'\rangle) \in V' \times V' \mid i' = i + 1, (j, j') \in E\}$. Note that $|V'| = n^2$ and $|E'| = |V||E| \le 3|V|^2 = 3n^2$ since $|E| \le 3|V|$. Moreover, let $s' = \langle 1, s\rangle$ and $t' = \langle n, t\rangle$. This new graph G' satisfies the following property, called the *topological order*: for any pair $i, j \in V'$, $(i, j) \in E'$ implies $i < j$.

From (G', s', t'), we want to define $w = bin(1)\#bin(2)\#\cdots\#bin(n)\#$, where $bin(i)$ indicates the binary representation of a natural number i and # is a designated separator not in $\{0, 1\}$. Moreover, for each edge $(i, j) \in E'$, we define $w_{ij} = bin(i{+}1)\#bin(i{+}2)\#\cdots\#bin(j)\#$. It follows that, for each w_{ij}, if w_{ij} is a substring of w, then w_{ij} must be unique. Note that $z = (w, w_{ij})_{(i,j)\in E'}$ is an instance to Search-1NFA with $m_{elm}(z) = |E'| \le 3n^2 = 3m_{ver}(x)^2$.

By running A on input z, we can solve (3DSTCON, m_{edg}) for instance x in time polynomial in $|x|$ using space at most $cm_{elm}(z)^{1/2-\varepsilon}$, which equals $3cm_{ver}(x)^{1-2\varepsilon}$. This contradicts LSH for 3DSTCON, which implies LSH for 2SAT$_3$ by Theorem 1.5(3). ◀

The next practical application of the working hypothesis LSH for $\mathrm{2SAT}_3$ targets the area of combinatorial NL optimization. An *NL optimization problem* (or an NLO problem) P is a tuple $(I, SOL, mes, goal)$ with $I \in \mathrm{L}$, $I \circ SOL \in \mathrm{auxL}$, $mes : I \circ SOL \to \mathbb{N}^+$ in auxFL, and $goal \in \{\textsc{max}, \textsc{min}\}$. See [19, 20] for its precise definition. Let NLO stand for the class of all NLO problems. An *optimal solution* y for instance x must satisfy $mes(x,y) = mes^*(x)$, where $mes^*(x) = goal_{y \in SOL(x)}\{mes(x,y)\}$. The *performance ratio* R of a solution y on an instance x is $R(x,y) = \max\{\frac{mes^*(x)}{mes(x,y)}, \frac{mes(x,y)}{mes^*(x)}\}$.

We say that an NLO problem $P = (I, SOL, mes, goal)$ parameterized by size parameter m is *solvable using log space* if there is a deterministic Turing machine that takes any instance $x \in I$ and outputs an optimal solution in $SOL(x)$ using logarithmically many tape cells in terms of size parameter $m(x)$. We write $\mathrm{LO}_{\mathrm{NLO}}$ to denote the class of all NLO problems solvable in polynomial time.

An NPO problem P is said to be *log-space γ-approximable* if there is a log-space Turing machine such that, for any instance x, if $SOL(x) \neq \emptyset$, then M outputs a solution in $SOL(x)$ with $R(x, M(x)) \leq \gamma$; otherwise, M outputs $\perp$ ("no solution"). The notation $\mathrm{LSAS}_{\mathrm{NLO}}$ denotes the class of NLO problems P for which there exists a log-space approximation scheme for P, where a *log-space approximation scheme* for P is a deterministic Turing machine M that takes inputs of the form (x,k) and outputs a solution y of P using space at most $f(k)\log|x|$ for a certain log-space computable function $f : \mathbb{N} \to \mathbb{N}$ such that the performance ratio R satisfies $R(x,y) \leq 1 + 1/k$. It follows that $\mathrm{LO}_{\mathrm{NLO}} \subseteq \mathrm{LSAS}_{\mathrm{NLO}} \subseteq \mathrm{NLO}$. Here, we are focused on problems in $\mathrm{LSAS}_{\mathrm{NLO}}$, that is, NLO problems having log-space approximation schemes.

Let us recall an NLO problem, called Max-HPP, from Section 1.4. Theorem 1.8 states that no polynomial-time $O(k^{1/3}\log m_{col}(x))$-space algorithm that finds $(1+\frac{1}{k})$-approximate solutions solves (Max-HPP, m_{col}). To prove this theorem, we state a useful supporting lemma. An optimization problem $(I, SOL, mes, goal)$ parameterized by m is said to be *$g(m(x))$-bounded* if $mes(x,y) \leq g(m(x))$ holds for any $(x,y) \in I \circ SOL$.

▶ **Lemma 6.1.** *Let $c \geq 1$. Every $O(m(x)^c)$-bounded maximization problem in* $\mathrm{LSAS}_{\mathrm{NLO}}$*, parameterized by log-space size parameter $m(x)$, whose $(1+\frac{1}{k})$-approximate solutions are found using $O(k^{\frac{1}{2c+1}}\log m(x))$ space can be solved in polynomial time using $O(m(x)^{1/2-\varepsilon})$ space on instances x for a certain constant $\varepsilon \in (0, 1/2)$.*

Proof of Theorem 1.8. Let $\varepsilon > 0$. Note that (Max-HPP, m_{col}) is $m_{col}(z)$-bounded for any instance z. Assume that there is a polynomial-time $O(k^{1/3}\log m_{col}(z))$-space algorithm of finding $(1+\frac{1}{k})$-approximate solutions of Max-HPP on instances z. Lemma 6.1 then implies that (Max-HPP, m_{col}) is solved by a certain deterministic Turing machine M in polynomial time using space at most $cm_{col}(z)^{1/2-\varepsilon}$ on instances z for a certain constant $c > 0$. We want to use this machine M to solve (3DSTCON, m_{ver}) in polynomial time using sub-linear space.

Let $x = (G, s, t)$ be any instance given to 3DSTCON with $G = (V, E)$ and $n = |V| \geq 2$. We define another graph $G' = (V', E')$, where $V' = \{(i,v) \mid i \in [n], v \in V\}$ and $E' = \{((i,u),(i+1,v)) \mid i \in [n-1], (u,v) \in E\}$. Note that $|V'| = n^2$. We set $s' = (1,s)$ and $t' = (n,t)$. From this graph G', we want to construct an instance $z = (A, n, s')$ to Max-HPP, where A is a $|V'| \times |V'|$ matrix. By identifying vertices in V' with numbers in $[n^2]$, we set $A_{s't'} = A_{t's'} = A_{vv} = 1$ for any $v \in V' - \{t\}$ and $A_{t'v} = A_{vs'} = 1$ for all $v \in V'$. For any other pair $(u,v) \in V' \times V'$, if $(u,v) \in E'$, then we define $A_{vw} = n$; otherwise, define $A_{uv} = 1$. Note that $m_{col}(z) = n^2 = m_{ver}(x)^2$.

If there is a path $(v_1, v_2, \ldots, v_k)$ from s' to t' in G', then we define $v_{k+j} = v_k$ for all indices $j \in [n^2 - k]$. It then follows that $\sum_{i=1}^{n^2-1} A_{v_i v_{i+1}} = (n^2-1)n$ and clearly this is optimal. On the contrary, let $\gamma = (v_1, v_2, \ldots, v_{n^2})$ be an optimal solution with an optimal value $(n^2-1)n$. By the requirement of Max-HPP, v_1 must be s'. Moreover, $A_{v_i v_{i+1}} = n$ holds for each $i \in [n^2-1]$. Hence, if we allow a self-loop at vertex t' in G', then γ forms a path from s'. Since $|V'| = n^2$, γ must include t'. Hence, γ contains a subpath from s' to t' in G'.

We then run M on the input z to obtain an optimal index sequence γ. By the above argument, if $w(\gamma) = (n^2-1)n$, then a path from s to t exists; otherwise, there is no path from s to t. Since M uses at most $cm_{col}(z)^{1/2-\varepsilon}$ space, the space usage of the whole procedure is at most $cm_{col}(z)^{1/2-\varepsilon}$, which turns out to be $cm_{ver}(x)^{1-2\varepsilon}$ by $m_{col}(z) = m_{ver}(x)^2$. Therefore, 3DSTCON is solvable in polynomial time using sub-linear space. This contradicts LSH for 3DSTCON, which is equivalent to LSH for 2SAT$_3$ by Theorem 1.5(3). ◀

References

1 G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties, Springer-Verlag, 2003.

2 G. Barnes, J. F. Buss, W. L. Ruzzo, and B. Schieber. A sublinear space, polynomial time algorithm for directed s-t connectivity. SIAM J. Comput. 27 (1998) 1273–1282.

3 C. Calabro, R. Impagliazzo, V. Kabanets, and R. Paturi. The complexity of unique k-SAT: an isolation lemma for k-CNFs. J. Comput. System Sci. 74 (2008) 386–393.

4 S. A. Cook. The complexity of theorem-proving procedures. In the Proc. of STOC'71, pp.151–158, 1971.

5 S. A. Cook. Deterministic CFL's are accepted simultaneously in polynomial time and log squared space. In the Proc. of STOC'79, pp.338–345, 1979.

6 J. L. Gross, J. Yellen, and P. Zhang. Handbook of Graph Theory. CRC Press, 2014.

7 J. E. Hopcroft and J. D. Ullman. Introduction to Automata Theory, Languages, and Computation. Addison-Wesley, 1979.

8 N. Immerman. Nondeterministic space is closed under complement. SIAM J. Comput. 17 (1988) 935–938.

9 R. Impagliazzo and R. Paturi. On the complexity of k-SAT. J. Comput. System Sci. 62 (2001) 367–375.

10 R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? J. Comput. System Sci. 63 (2001) 512–530.

11 B. Jenner. Knapsack problems for NL. Inform. Process. Lett. 54 (1995) 169–174.

12 N. D. Jones. Space-bounded reducibility among combinatorial problems. J. Comput. System Sci. 11 (1975) 68–75.

13 N. D. Jones, Y. E. Lien, and W. T. Laaser. New problems complete for nondeterministic log space. Math. Systems Theory 10 (1976) 1–17.

14 D. Lokshtanov, D. Marx, and S. Saurabh. Lower bounds based on the exponential time hypothesis. Bulletin of the EATCS, No.105, pp.41–71, 2011.

15 O. Reingold. Undirected connectivity in log-space. J. ACM 55 (2008) article 17.

16 W. J. Savitch. Relationships between nondeterministic and deterministic tape complexities. J. Comput. System Sci. 4 (1970) 177–192.

17 I. H. Sudborough. On tape-bounded complexity classes and multihead finite automata. J. Comput. System Sci. 10 (1975) 62–76.

18 R. Szelepcsényi. The method of forced enumeration for nondeterministic automata. Acta Inform. 26 (1988) 279–284.

19 T. Tantau. Logspace optimization problems and their approximation properties. Theory Comput. Syst. 41 (2007) 327–350.

20 T. Yamakami. Uniform-circuit and logarithmic-space approximations of refined combinatorial optimization problems. In the Proc. of COCOA 2013, LNCS, vol.8287, pp.318–329 (2013). A complete version is available at arXiv:1601.01118v1, January 2016.