On Space Efficiency of Algorithms Working on Structural Decompositions of Graphs

Michał Pilipczuk¹ and Marcin Wrochna²

¹ Institute of Informatics, University of Warsaw, Warsaw, Poland
michal.pilipczuk@mimuw.edu.pl

² Institute of Informatics, University of Warsaw, Warsaw, Poland
m.wrochna@mimuw.edu.pl

Abstract

Dynamic programming on path and tree decompositions of graphs is a technique that is ubiquit-
ous in the field of parameterized and exponential-time algorithms. However, one of its drawbacks
is that the space usage is exponential in the decomposition’s width. Following the work of All-
ender et al. [Theory of Computing, ’14], we investigate whether this space complexity explosion is
unavoidable. Using the idea of reparameterization of Cai and Juedes [J. Comput. Syst. Sci., ’03],
we prove that the question is closely related to a conjecture that the Longest Common Sub-
sequence problem parameterized by the number of input strings does not admit an algorithm
that simultaneously uses XP time and FPT space. Moreover, we complete the complexity land-
scape sketched for pathwidth and treewidth by Allender et al. by considering the parameter
tree-depth. We prove that computations on tree-depth decompositions correspond to a model of
non-deterministic machines that work in polynomial time and logarithmic space, with access to
an auxiliary stack of maximum height equal to the decomposition’s depth. Together with the
results of Allender et al., this describes a hierarchy of complexity classes for polynomial-time non-
deterministic machines with different restrictions on the access to working space, which mirrors
the classic relations between treewidth, pathwidth, and tree-depth.

1998 ACM Subject Classification F.1.3 Complexity Measures and Classes, F.2.3 Tradeoffs
between Complexity Measures, G.2.2 Graph Theory

Keywords and phrases tree decomposition, LCS, tree-depth, NAuxSA, Savitch’s theorem

Digital Object Identifier 10.4230/LIPIcs.STACS.2016.57

1 Introduction

Treewidth is a parameter that measures how easily a graph can be decomposed into a tree-like
structure, called a tree decomposition. While initially introduced by Robertson and Seymour
in their Graph Minors project [41], treewidth has found numerous applications in the field of
algorithms, because many problems that are intractable on general graphs, become efficiently
solvable on graphs of small treewidth. Theorems of Courcelle [14] and of Arnborg et al. [5]
explain that every problem expressible in Monadic Second Order logic can be solved in time
f(s)·n on graphs of treewidth s and size n, for some function f. While f can be non-elementary
in general, for many classic problems, like VERTEX COVER, 3COLORING, or DOMINATING

* Research supported by Polish National Science Centre grant DEC-2013/11/D/ST6/03073. During the
work on these results, Michał Pilipczuk held a post-doc position at Warsaw Center of Mathematics and
Computer Science, and was supported by Foundation for Polish Science (FNP) via START stipend
programme.
On Space Efficiency of Algorithms Working on Structural Decompositions of Graphs

Set, the natural dynamic programming approach yields a running time of \( O(c^n \cdot n) \) for a small constant \( c \). Dynamic programming procedures working on tree decompositions are important for applications, as they often serve as critical subroutines in more complex techniques, e.g., subexponential parameterized algorithms derived using bidimensionality [16], or approximation schemes obtained via Baker’s approach [7]. Algorithms working on tree decompositions are usually analyzed in the paradigm of parameterized complexity, where treewidth is the considered parameter. We refer to textbooks [15, 17, 22] for a broad introduction, and to a recent survey of Langer et al. [30] for more specific results.

A certain limitation of dynamic programming on a tree decomposition is that it uses space exponential in its width, which is often a prohibitive factor in practical applications. Therefore, recently there is much focus on reducing the space complexity of exponential-time algorithms to polynomial, even at the cost of slightly worsening the time complexity [6, 9, 23, 24, 34, 37]. Here, the usage of algebraic tools proved to be an extremely useful approach. Unfortunately, algorithms working on treewidth remain a family where virtually no progress has been achieved in this matter. Therefore, a natural question arises: Can we reduce the space complexity of algorithms working on tree decompositions while keeping (or moderately worsening) their time complexity? This was first asked explicitly by Lokshtanov et al. [33], who sketched how a simple tradeoff achieves polynomial space complexity while increasing the time complexity to \( 2^O(s^2) + O(n^2) \). The question was reiterated later by Langer et al. [30].

Following early completeness results of Monien and Sudborough [36] on bandwidth-constrained problems and of Gottlob et al. [26] on conjunctive queries of bounded treewidth, Allender et al. [4] recently initiated the systematic study of satisfaction complexity in variable path- and treewidth. Essentially, they observe that CSP-like problems—say, 3COLORING for concreteness\(^1\)—when limited to instances of small treewidth or pathwidth, are complete for certain complexity classes under logspace reductions. More precisely, when the input graph is equipped with a path decomposition of width at most \( s(n) \geq \log n \), for some fixed function \( s \) of the input size, then 3COLORING (denoted in this case as pw-3COLORING[\( s \)]) is complete for the class \( \text{N}[\text{poly}, s(\text{poly})] \): problems that admit non-deterministic algorithms working simultaneously in polynomial time and space \( O(s(\text{poly})n) \). Similarly tw-3COLORING[\( s \]), where \( s(n) \) bounds the width of a given tree decomposition, is complete for the class \( \text{NAuxPDA}[\text{poly}, s(\text{poly})] \); the difference with \( \text{N}[\text{poly}, s(\text{poly})] \) is that the algorithm can use an auxiliary push-down of unlimited size, to which read/write access is only from the top. Allender et al. also describe the class in terms of semi-unbounded fan-in (SAC) circuits. They assume \( s(n) = \log^k n \), but the proof works in the more general setting given below.

\[ \textbf{Theorem 1 (}[4]\text{).} \text{ Let } s(n) \geq \log n \text{ be a nice function}\(^2\). Then pw-3COLORING[\( s \)] is complete for \( \text{N}[\text{poly}, s(\text{poly})] \) under logspace reductions, whereas tw-3COLORING[\( s \)] is complete for \( \text{NAuxPDA}[\text{poly}, s(\text{poly})] \) under logspace reductions. \]

Thus, the feasibility of various space-time tradeoffs when working on tree/path decompositions is equivalent to inclusions of corresponding complexity classes. For instance (assuming for conciseness \( \forall c, s(n^c) = O(s(n)) \), e.g., \( s(n) = \log^k n \) for \( k \geq 1 \)), pw-3COLORING[\( s \)] is solvable:

\(^1\) Allender et al. use SAT parameterized by treewidth/pathwidth of its primal graph as an exemplary problem, but SAT and 3COLORING can be easily seen to be equivalent under logspace reductions; see Lemma 10. In this paper, we prefer to use 3COLORING as an exemplary hard CSP-like problem.

\(^2\) By a nice function we mean a function \( s \) that is constructible and such that \( s(n)/\lg n \) is non-decreasing.
in time $2^{o(s(n) \log n)}$ and space $2^{o(s(n))}$ if and only if $\mathbf{N}[\text{poly}, s] \subseteq \mathbf{D}[2^{o(s \log)}, 2^{o(s)}]$;

- in time $2^{O(s(n))}$ and space $\text{poly}(n)$ if and only if $\mathbf{N}[\text{poly}, s] \subseteq \mathbf{D}[2^{O(s)}, \text{poly}]$.

Similar statements can be inferred for treewidth. In contrast, the best known determinization for $\mathbf{N}[\text{poly}, s]$ come from a brute-force approach or Savitch’s theorem [43], yielding respectively (for $s(n) \geq \log n$) $\mathbf{D}[2^{O(s)}] = \mathbf{D}[2^{O(s)}, 2^{O(s)}]$ and $\mathbf{D}[s \cdot \log] = \mathbf{D}[2^{O(s \cdot \log n)}], s \cdot \log]$.

In this manner, Allender et al. conclude that, intuitively speaking, achieving better time-space tradeoffs for algorithms working on path and tree decompositions of small width would require developing a general technique of improving upon the tradeoff of Savitch. As Lipton phrased it, “one of the biggest embarrassments of complexity theory is the fact that Savitch’s theorem has not been improved [...] Nor has anyone proved that it is tight” [31].

Allender et al. argue that such an improvement would contradict certain rescaled variants of known conjectures about the containment of time- and space-constrained classes, in particular the assumption that $\mathbf{NL} \nsubseteq \mathbf{SC}$; we refer to [4] for details. We consider the study of Allender et al. not as a definite answer in the topic, but rather as an invitation to a further investigation of the introduced conjectures.

**Our Contribution.** In the **Longest Common Subsequence** problem (LCS), we are given an alphabet $\Sigma$ and $k$ strings over $\Sigma$, and ask for the longest sequence of symbols that appears as a subsequence in each input string. The applicability of the LCS problem in, e.g., computational biology, motivated many to search for faster, more space-efficient algorithms, as the classical dynamic programming solution, running in time and space $O(n^k)$ (where $n$ is the length of each string) is often far from practical. From the point of view of parameterized complexity, LCS parameterized by $k$ is $\text{W}[t]$-hard for every level $t$ [11], remains $\text{W}[1]$-hard for a fixed-sized alphabet [39], and is $\text{W}[1]$-complete when parameterized jointly by $k$ and $\ell$, the target length of the subsequence [27]. In a recent breakthrough, Abboud et al. [1] proved that the existence of an algorithm with running time $O(n^{k-\varepsilon})$, for any $\varepsilon > 0$, would contradict the Strong Exponential Time Hypothesis. As far as the space complexity is concerned, only modest progress has been achieved: The best known result, by Barsky et al. [8], improves the space complexity to $O(n^{k-1})$. This motivates us to formulate the following conjecture.

> **Conjecture 2.** There is no algorithm for LCS that works in time $n^{f(k)}$ and space $f(k)\text{poly}(n)$ for a computable function $f$, where $k$ is the number of input strings and $n$ their total length.

Quite surprisingly, we show that Conjecture 2 is closely related to the question of time-space tradeoffs for algorithms working on small pathwidth, as detailed in Theorem 15. There, the conjecture is sandwiched between a weaker statement that it is impossible to achieve subexponential space while keeping single exponential time complexity, and a stronger statement that this holds even if we allow the time complexity exponent to increase by an arbitrarily slowly growing function of the width. To prove this, we use a completeness result of Elberfeld et al. [21] for LCS, which allows to formulate Conjecture 2 as an equivalent statement in parameterized complexity about the impossibility of determinization results improving upon Savitch’s theorem. Using the ideas of Cai and Juedes [12] connecting subexponential complexity to fixed-parameter tractability, we consider a reparameterized version of pw-3COLORING. This allows us to compare questions concerning time-space tradeoffs for pw-3COLORING and determinization of $\mathbf{N}[t, s]$ classes to those concerning parameterized classes and the complexity of LCS. In particular, we show that Conjecture 2
implies $\text{NL} \not\subseteq \text{D[time poly, space poly log]}$ (the latter class being usually called $\text{SC}$) and is implied by a rescaled version of the following stronger variant: $\text{NL} \not\subseteq \text{D[time } 2^{\text{poly}(\log^2 n), \text{space } n^{o(1)}]}$.

In the second part of this work, we complement the findings of Allender et al. [4] by considering the graph parameter tree-depth. Tree-depth of a graph is lower bounded by its pathwidth and upper bounded by its treewidth times $\log n$. Our motivation for considering this parameter is two-fold. First, recent advances have uncovered a wide range of topics where tree-depth appears naturally. For instance, it plays an important role in the theory of sparse graphs of Nešetřil and Ossona de Mendez [38], it is the key factor in classifying homomorphism problems that can be solved in logspace [13], and characterizes classes of graphs where the expressive power of First-Order and Monadic Second-Order logic coincides [19]. It was rediscovered several times under different names: minimum elimination tree height [40], ordered chromatic number [28], vertex ranking [10], or the maximum number of introduce nodes on a root-to-leaf path of a tree decomposition [24].

Second, algorithms working on tree-depth decompositions model generic exponential-time Divide&Conquer algorithms. In this approach, after finding a small, balanced separator $S$ in the graph, the algorithm tries all possible ways a solution can interact with $S$, and solves connected components of $G - S$ recursively. This naturally gives rise to a tree-depth decomposition of the graph, where $S$ is placed on top, and decompositions of the components of $G - S$ are attached below it as subtrees. The maximum total number of separator vertices handled at any moment in the recursion corresponds to the depth of the decomposition. Thus, many classic Divide&Conquer algorithms, including the ones derived for planar graphs using the Lipton-Tarjan separator theorem [32], can be reinterpreted as first building a tree-depth decomposition of the graph using a separator theorem, and then running the algorithm on it.

Most importantly for us, recursive algorithms working on tree-depth decompositions run in polynomial space. For instance, such an algorithm for 3Coloring on a tree-depth decomposition of depth $s$ runs in time $3^s \cdot \text{poly}(n)$ and space $O(s + \log n)$ (see Lemma 20), which places td-3Coloring in $\text{D[time } 2^{\text{O}(s)}\text{poly}, \text{space } s + \log n] = \text{D[time } 2^s, \text{space } s + \log n]$]. This is immediate for CSP-like problems like 3Coloring, but recently Fürer and Yu [24] showed that algebraic transforms can be used to reduce the space usage to polynomial in $n$ also for other problems, like counting perfect matchings or dominating sets. We describe how this approach gives an $3^s \cdot \text{poly}(n)$-time poly(n)-space algorithm for Dominating Set in more detail in the full version of the article. This means that the reduction of space complexity that is conjectured to be impossible for treewidth and pathwidth, actually is possible for tree-depth. Therefore, we believe that it is useful to study the computation model standing behind low tree-depth decompositions, in order to understand how it differs from the models for treewidth and pathwidth.

Consequently, mirroring Theorem 1, we prove that computations on tree-depth decompositions exactly correspond to the class $\text{NAuxSA[time } \text{poly, space } \log \text{, height } s]$; problems that can be decided by a non-deterministic Turing Machine that uses polynomial time and logarithmic space, but also has access to an auxiliary stack of maximum height $s$. The stack can be freely read by the machine, just as the input tape, but write access is only via push/pop operations.

▶ Theorem 3. Let $s(n) \geq \log^2 n$ be a nice function. Then td-3Coloring is complete for $\text{NAuxSA[time } \text{poly, space } \log, \text{height } s(\text{poly})]$ under logspace reductions.
Thus, computations on tree-depth and path decompositions differ by the access restrictions to $O(s)$ space used by the machine. While for pathwidth this space can be accessed freely, for tree-depth all except an $O(\log n)$ working buffer has to be organized in a stack.

The proof of Theorem 3 largely follows the approach of Akatov and Gottlob [3], who proved a different completeness result for the class $\text{NAuxSA}_{\text{time space height}} \left[ \text{poly log height} \right]$, which they call $\text{DC}^1$. The main idea is to regularize the run of the machine so that the push-pop tree has the rigid shape of a full binary tree. Then we can use this concrete structure to “wrap around” gadgets encoding an accepting run of a regularized NAuxSA machine. However, the motivation in the work of Akatov and Gottlob was answering conjunctive queries in a hypergraph by exploiting a kind of balanced decomposition, and hence the problem proven to be complete for $\text{DC}^1$ is a quite general and expressive problem originating in database motivations; see [2, 3] for details. In our setting, in order to get a reduction to $3\text{COLORING}$, we need to work more to encode an accepting run. In particular, to encode each part of the computation where no push or pop is performed, instead of producing a single atom in a conjunctive query, we use computation gadgets that originate in Cook’s proof of the NP-completeness of SAT. The assumption that the computation has a polynomial number of steps is essential here for bounding the tree-depth of each such gadget. This way, Theorem 3 presents a more natural complete problem for $\text{DC}^1$.

Another difference is that Theorem 3 works for any well-behaved function $s(n) \geq \log^2 n$, as opposed to the bound $s(n) = \log^2 n$ inherent to the problem considered by Akatov and Gottlob. For this, the crucial new idea is to increase the working space of the machine to $s(n)/\log n$ in order to be able to perform regularization – a move that looks dangerous at first glance, but turns out not to increase the expressive power of the computation model. This proves the following interesting by-product of our work.

**Theorem 4.** Let $s(n) \geq \log^2 n$ be a nice function. Then

$$\text{NAuxSA}_{\text{time space height}} \left[ \text{poly, log, s(poly)} \right] = \text{NAuxSA}_{\text{time space height}} \left[ \text{poly, s(poly)/log, s(poly)} \right].$$

The following determinization follows from the $O(s + \log n)$ algorithm for td-$3\text{COLORING}$.

**Theorem 5.** Let $s(n) \geq \log^2(n)$ be a nice function. Then

$$\text{NAuxSA}_{\text{time space height}} \left[ \text{poly, log, s(poly)} \right] \subseteq \text{D}_{\text{space}} \left[ \text{s(poly)} \right].$$

Theorem 5 for $s(n) = \log^2 n$ also follows from the work of Akatov and Gottlob [3]. Observe that now the justification for the assumption $s(n) \geq \log^2 n$ becomes apparent: for, say, $s(n) = \log n$, the theorem would state that $L = NL$, a highly unexpected outcome.

We find Theorem 5 interesting, because a naive simulation of the whole configuration space for NAuxSA would require space exponential in $s$. It appears, however, that the exponential blow-up of the space complexity can be avoided. We do not see any significantly simpler way to prove this result other than going through the td-$3\text{COLORING}[s]$ problem, and hence it seems that the tree-depth view gives a valuable insight into the computation model of NAuxSA. The classic relations between treewidth, pathwidth and tree-depth are, through completeness results, mirrored in a hierarchy between NAuxPDA, N, and NAuxSA classes, as detailed in the concluding section. In particular, this answers a question of Akatov and Gottlob [2, 3] about the relation of $\text{NAuxSA}_{\text{time space height}} \left[ \text{poly, log, poly log} \right]$ to other classes in $\text{NP}$.

Finally, using Theorem 3 we also give an alternative view on NAuxSA computations using alternating Turing machines in Theorem 21, answering another question of Akatov and Gottlob. From this point of view, Theorem 5 is immediate.
2 Preliminaries

Reductions and complexity classes. For two languages $P, Q$, we write $P \leq_L Q$ when $P$ is logspace reducible to $Q$. Most of the complexity classes we consider are closed under logspace reductions. Because we handle various measures of complexity and compare a wide array of classes that bound two measures simultaneously, we introduce the following notation. A complexity class is first described by the machine model: $D, N, A$ denote deterministic, non-deterministic, and alternating (see [42]) Turing machines, respectively. Then bounds on complexity measures are described (up to constant factors) as a list of functions with the measure’s name underneath. All functions except the symbol $f$ (which we reserve for classes in parameterized complexity) are functions of the input size $n$. For example, $N[ \text{time } n^{O(1)} ]$, e.g., $D[\text{poly}]=P$. An auxiliary push-down or stack is denoted as AuxPDA or AuxSA, respectively: the difference is that a push-down can only be read at the top, while a stack can be read just as a tape (both can be written to only by pushing and popping symbols at the top), see e.g. [44]. The measure named height is the maximum height of the push-down or stack.

We write $\lg$ for the logarithm with base 2 and log when the base is irrelevant. We say a function $s : \mathbb{N} \rightarrow \mathbb{N}$ is constructible if there is a Turing machine which given a number $n$ in unary outputs $s(n)$ in unary using logarithmic space; in particular, this implies $s(n) \leq \text{poly}(n)$. A function $s$ is nice if it is constructible and $\frac{4(n)}{\lg n}$ is non-decreasing. For simplicity, we will assume all functions $s : \mathbb{N} \rightarrow \mathbb{N}$ describing complexity bounds to be nice.

Note that logspace reductions can blow-up instance sizes polynomially, hence the closure of $N[\text{poly }, s ]$ under such reductions is $N[\text{poly }, \text{poly}(s)]$, for example. These are equal for functions $s(n)$ such that $s(\text{poly}(n)) \leq O(s(n))$ (that is, if for every $c > 0$ there is a $d > 0$ such that $s(n^c) \leq d \cdot s(n)$). This includes $\lg^k(n)$ for any $k \geq 1$ and $\lg n \lg \lg n$, for example.

Structural parameters. We recall the definition of tree-depth. See e.g. [15, 41] for definitions of treewidth and pathwidth. For technical reasons, we assume that in all given tree and path decompositions $T$, $|T| \leq 2|V(G)|^2$; standard methods allow to prune any decomposition to this size in logspace, see e.g. [29, Lemma 13.1.2]. For conciseness, we will refer to the certifying structures as decompositions for all three parameters.

Definition 6 (tree-depth). A tree-depth decomposition of an undirected graph $G$ is a rooted forest $T$ (disjoint union of rooted trees) together with a bijection $\mu$ from the vertices of $G$ to the nodes of $T$, such that for every edge $uv$ of $G$, $\mu(u)$ is an ancestor of $\mu(v)$ or vice-versa in $T$. The depth of $T$ is the largest number of nodes on a path between a root and a leaf. The tree-depth of $G$ is the minimum depth over all possible tree-depth decompositions of $G$.

The following lemma describes well-known inequalities between the three parameters.

Lemma 7 (♠). There is a constant $c \in \mathbb{N}$ such that for any graph $G$, $\text{td}(G) \geq \text{pw}(G) \geq \text{tw}(G) \geq \text{td}(G)/(c \cdot \log |V(G)|)$. Furthermore, each inequality is certified by an algorithm that transforms the respective graph decompositions in logspace.

For a graph problem, such as 3COLORING, a structural parameter $\pi \in \{\text{td}, \text{pw}, \text{tw}\}$, and a nice function $s : \mathbb{N} \rightarrow \mathbb{N}$, we define $\pi$-$3$COLORING$[s]$ to be the decision problem where given

\[3 \text{ Proofs of statements marked with } \spadesuit \text{ are deferred to the full version of the article (in the appendix).} \]
an instance $G$ of 3Coloring and a $\pi$-decomposition of $G$, we ask whether the decomposition has width at most $s(|V(G)|)$ and $G$ is a yes-instance of 3Coloring. The assumption that a decomposition is given on input is to factor away the complexity of finding it, which is a problem not directly relevant to our work. Note that the validity and width/depth of a decomposition given in any natural encoding can easily be checked in logarithmic space.

Observe also that for any $c > 0$, $\pi$-3Coloring[$s(n)$] is equivalent to $\pi$-3Coloring[$s(n^c)$] under logspace reductions. A reduction to $\pi$-3Coloring[$s(n^c)$] is trivial, while the reverse reduction follows easily by padding: adding isolated vertices up to size $n^c$ that do not change the answer nor the value of $\pi$. Also, as we assume $s$ to be nice, we have $\frac{s(n)}{\lg n} \leq \frac{s(n^c)}{\lg n^c}$, hence $c \cdot s(n) \leq s(n^c)$ for any $c \geq 1$. This implies that $\pi$-3Coloring[$c \cdot s(n)$] is equivalent to $\pi$-3Coloring[$s(n)$]. Thus, the hierarchy of Lemma 7 takes the following form.

**Corollary 8.** Let $s : \mathbb{N} \to \mathbb{N}$ be a nice function. Then

$$\text{td-3Coloring}[s] \leq L \text{ pw-3Coloring}[s] \leq L \text{ tw-3Coloring}[s] \leq L \text{ td-3Coloring}[s \cdot \log].$$

### Equivalence of problems.

A reduction between two graph problems preserves structural parameters if for each parameter $\text{tw, pw, td}$ and any instance with graph $G$, a decomposition of $G$ of width/depth at most $s$ can be transformed in logspace into a decomposition of the graph $H$ produced by the reduction of width/depth at most $O(s)$. Many NP-hardness reductions have this property, in particular those that replace each vertex or edge with a constant-size gadget (see the ‘local replacement’ and ‘component design’ methods in Garey and Johnson [25]). For example, 3Coloring and variants of SAT are equivalent in all our theorems, while VERTEX COVER or DOMINATING SET (defined in [25]) are at least as hard.

**Definition 9.** Let $\phi$ be a CNF formula. The primal (Gaifman) graph of $\phi$ is the graph with a vertex for each variable of $\phi$ and an edge between every pair of variables that appear together in some clause. The incidence graph of $\phi$ is the bipartite graph with a vertex for each clause and each variable of $\phi$, where every clause is adjacent to variables contained in it.

**Lemma 10 (♠).** The following problems are equivalent under logspace reductions preserving structural parameters: 3Coloring, CNF-SAT (with the primal graph), $k$-SAT (with either the primal or incidence graph) for each $k \geq 3$. Furthermore, VERTEX COVER, INDEPENDENT SET and DOMINATING SET each admit such a reduction from the above problems.

### Cook’s theorem with bounded space.

A common element in our reductions is the description of Turing machine computations using CNF formulas, as in Cook’s theorem. Already Monien and Sudborough [36] observed that Cook’s reduction applied to machines with bounded space yields formulas of bounded width. The difference is that machine’s worktape space bound can be much smaller than the input word—access to the read-only input tape has to be implemented differently. To later handle stack machines, we also need to consider a second input tape separately. We informally state the version of Cook’s construction we need.

**Lemma 11 (Computation gadget, ⊠).** Let $M$ be an NTM over alphabet $\Sigma$ with two read-only input tapes and one work tape. Given an input word $\alpha$ of length $n$ and integers $s,t,h$, in unary such that $\lg n, \lg h \leq O(s)$, one can in logspace output a CNF formula such that:

- The formula has poly$(n,t,s,h)$ variables, including named variables $u_1, \ldots, u_h$, $v_1, \ldots, v_h$, describing: a word $\bar{w}$ and two configurations $\mathbf{u}, \mathbf{v}$ of $M$ (up to $s$ symbols of the working tape, head’s positions encoded in binary, and the state).
- Any assignment to the named variables can be extended to a satisfying assignment iff $M$ on inputs $\alpha$ and $\bar{w}$ has a run from $\mathbf{u}$ to $\mathbf{v}$, using at most $t$ steps and $s$ space.
- The formula’s primal graph has pathwidth $O(s+h)$ and tree-depth $O(s \cdot \log(n+s+t+h)+h)$.
3 Connections with Tradeoffs for LCS

In this section we relate Conjecture 2 to statements of varying strength concerning different time-space tradeoffs. The results are summarized in Figure 1.

A \textit{pl-reduction} between parameterized problems is an algorithm that transforms an instance of one problem with parameter \( k \) into an equivalent instance of another problem with parameter \( k' \leq f(k) \), working in space \( f(k) + \mathcal{O}(\log n) \), for some computable \( f \). Following Elberfeld et al. [21] we define \( \mathcal{N}[f, \log] \) as the class of parameterized problems that can be solved in non-deterministic time \( f(k)\text{poly}(n) \) and space \( f(k)\log(n) \) for some computable function \( f \), where \( k \) is the parameter. Deterministic classes \( \mathcal{D}[t, s] \) are defined analogously for various expressions \( t, s \). All those mentioned in the article are closed under pl-reductions.

We do not use the better known fpt-reductions because \( \mathcal{N}[f, \log] \) is not expected to be closed under them; its closure under fpt-reductions has been called WNL by Guillemot [27].

We use \( o_{\text{eff}}(h(n)) \) as an effective variant of \( o(h(n)) \): for \( f, h : N \to N \) we write \( f = o_{\text{eff}}(h) \) if there is a non-decreasing, unbounded, computable function \( g(n) \) such that \( f = O\left(\frac{h}{g}\right) \).

The \textit{inverse} of a function \( f \) is the function \( f^{-1}(n) := \max\{i \mid f(i) \leq n\} \); observe that \( f(f^{-1}(n)) \leq n \leq f^{-1}(f(n)) \). Conjecture 2 concerns the following parameterized problem.

<table>
<thead>
<tr>
<th>LCS</th>
<th>Parameter: ( k )</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> A finite alphabet ( \Sigma ), ( k ) strings ( s_1, s_2, \ldots, s_k ) over ( \Sigma ), and an integer ( \ell ).</td>
<td></td>
</tr>
<tr>
<td><strong>Question:</strong> Is there a common subsequence of ( s_1, s_2, \ldots, s_k ) of length at least ( \ell )?</td>
<td></td>
</tr>
</tbody>
</table>

Elberfeld et al. [21], drawing on the work of Guillemot [27], pinpointed the complexity of LCS, allowing Conjecture 2 to be phrased as a general statement in parameterized complexity.

\begin{itemize}
  \item \textbf{Theorem 12 ([21])}. LCS is complete for \( \mathcal{N}[\text{poly}, \log] \) under pl-reductions.
  \item \textbf{Corollary 13}. Conjecture 2 holds if and only if \( \mathcal{N}[\text{poly}, \log] \not\subseteq \mathcal{D}[n^f, \text{poly}] \).
\end{itemize}

Similarly as described in the introduction, the best known determinization results can only place \( \mathcal{N}[\text{poly}, \log] \) in \( \mathcal{D}[n^f, n^f] \) (commonly known as \( \text{XP} \)) and \( \mathcal{D}[n^{f(k)\log n}, f(k)\cdot \log^2 n] \).

Following Cai and Juedes [12], to relate parameterized tractability bounds to subexponential bounds, we define a reparameterized version of \( \text{pw-3COLORING} \).

<table>
<thead>
<tr>
<th>\text{pw-3COLORING}^{\log n}</th>
<th>Parameter: ( \frac{s}{\log n} )</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> A graph ( G ) and a path decomposition of ( G ) of width ( s )</td>
<td></td>
</tr>
<tr>
<td><strong>Question:</strong> Is ( G ) 3-colorable?</td>
<td></td>
</tr>
</tbody>
</table>

Similarly as in Theorem 1, pathwidth-constrained problems turn out to be complete for non-deterministic computation with simultaneous time and space bounds.

\begin{itemize}
  \item \textbf{Theorem 14 (♠)}. \text{pw-3COLORING}^{\log n} is complete for \( \mathcal{N}[\text{poly}, \log] \) under pl-reductions.
\end{itemize}

Containment follows from a poly-time, \( \mathcal{O}(s + \log n) \)-space non-deterministic algorithm that proceeds on consecutive bags of the decomposition, guessing each vertex color and remembering only those in the current bag. Completeness is proved with a direct application of Lemma 11 to a problem with parameter \( k \) solved in space \( \mathcal{O}(f(k)\log n) \), yielding through Lemma 10 a \text{pw-3COLORING} instance of width \( s = \mathcal{O}(f(k)\log n) \).

Conjecture 2 is thus equivalent to the statement that \text{pw-3COLORING}^{\log n} is not in \( \mathcal{D}[n^f, \text{poly}] \), which gives Theorem 15.1. To contrast pathwidth with tree-depth, Lemma 20...
(introduced later) places td-3COLORING$^{\log n}$ in $D[n^f, f \log f]$, a class known as XL. Similarly as in the work of Cai and Juedes [12], we show that also subexponential bounds on the complexity of pw-3COLORING are related to the parameterized complexity of pw-3COLORING$^{\log n}$. This gives the sandwiching of Conjecture 2 between two similar statements in Theorem 15.2, 15.3. An even weaker statement is proved equivalent to $NL \not\subseteq SC$ by a simple padding argument in Theorem 15.4. For a somewhat less natural, stronger variant of Conjecture 2, we can show a similar, but exact correspondence in 15.5 (note the quasi-polynomial factor on both sides).

> Theorem 15 (a). Consider deterministic algorithms for pw-3COLORING working on instances of size $n$ with a given path decomposition of width $s$ (uniformly for all values of $s$).

1. There is no such algorithm working in time $n^{o(s/\log n)}$ and space $f(s/\log n)\text{poly}(n)$ for any computable $f$ if and only if Conjecture 2 holds.
2. Assuming Conjecture 2, there is no such algorithm working in time $2^{O(s)}\text{poly}(n)$ and space $2^{o(s)}\text{poly}(n)$.
3. If Conjecture 2 fails, then for every unbounded, computable function $g$, there is such an algorithm working in time $2^{g(s)}\text{poly}(n)$ and space $2^{o(g(s))}\text{poly}(n)$.
4. There is no such algorithm working in time $2^{O(s)}\text{poly}(n)$ and space $\text{poly}(s, \log n)$ if and only if $NL \not\subseteq SC$.
5. There is no such algorithm working in time $2^{o(s^2)}n^{O(\log n)}$ and space $2^{o(s)}\text{poly}(n)$ if and only if $N[f, \text{poly}, f \log f] \not\subseteq D[n^f, f \text{poly}]$.

Proof of Theorem 15(2). Suppose to the contrary that pw-3COLORING can be solved in time $2^{O(s)}\text{poly}(n)$ and space $2^{o(s)}\text{poly}(n)$. We show that $N[f, \text{poly}, f \log f] \not\subseteq D[n^f, f \text{poly}]$, contradicting Conjecture 2. The assumption implies that pw-3COLORING$^{\log n}$ can be solved in time $2^{O(k \log n)} = n^{O(k)}$ and space $2^{k \log n/g(k \log n)}$ for some unbounded and non-decreasing computable function $g(\cdot)$. If $k \leq g(k \cdot \log n)$, then the bound on space is bounded by $n$. Otherwise, if $k > g(k \log n) \geq g(\log n)$, then $n \leq 2^{g^{-1}(k)}$. In this case the bound on space is bounded by a computable function of $k$, namely $2^{k \cdot g^{-1}(k)}$. Hence in each case, the same algorithm solves pw-3COLORING$^{\log n}$ in time $n^{O(k)}$ and space $n + 2^{k \cdot g^{-1}(k)}$. By Theorem 14, this implies $N[f, \text{poly}, f \log f] \not\subseteq D[n^f, f \text{poly}]$.

We summarize the relationships around Conjecture 2 in Figure 1. The weakest statement there is $NL \not\subseteq SC$, a widely explored hypothesis in complexity theory. Since Directed $(s, t)$-Reachability (asking given a directed graph and two nodes $s, t$, whether $t$ reachable from $s$) is an NL-complete problem, this is also equivalent to the question of whether this problem can be decided in polynomial time and polylogarithmic space. However, even this weakest statement is not known to be implied by better known conjectures such as the Exponential Time Hypothesis. It seems that the simultaneous requirement on bounding two complexity measures—time and space—has a nature independent of the usual time complexity considerations. Hence, new assumptions may be needed to explore this paradigm, and we hope that Conjecture 2 may serve as a transparent and robust example of such.

In a certain restricted computation model (allowing operations on graph nodes only, not on individual bits), unconditional tight lower bounds have been proved by Edmonds et al. [18]: it is impossible to decide Directed $(s, t)$-Reachability in time $2^{o(\log^2 \log n)}$ and space $O(n^{1-\varepsilon})$ (for any $\varepsilon > 0$), even if randomization is allowed. Essentially all known techniques for solving Directed $(s, t)$-Reachability are known to be implementable in this model [35] (including DFS, BFS, theorems of Savitch, of Immerman and Szelepcsényi, as well as Reingold’s breakthrough), therefore this strongly suggests that no algorithm running in time $2^{o(\log^2 \log n)}$ and space $n^{o(1)}$ is possible, that is, $NL \not\subseteq D[2^{o(\log^2 \log n)}, n^{o(1)}]$.
There is no algorithm for \( \text{pw-3Coloring} \) working in time \( 2^{o_{\text{eff}}(s^2)} n^{O(\log n)} \) and space \( 2^{o_{\text{eff}}(s)} \text{poly}(n) \) by a trivial padding argument, but the reverse implication is also probable in the sense that any proof of the latter would likely scale to prove the former. However, it is still possible that an algorithm working in polynomial space refutes the stronger statement even though \( \text{NL} \not\subseteq \text{D}[2^{o_{\text{eff}}(\log^2 n)}, \text{poly}(n)] \).

**Figure 1** A summary of the relationships between various statements related to Conjecture 2.

By Theorem 1, this is equivalent to saying that \( \text{pw-3Coloring}[\log] \) cannot be solved in these time and space bounds. The strongest statement on Figure 1 is a rescaling of this, that is, it implies \( \text{NL} \not\subseteq \text{D}[2^{o_{\text{eff}}(\log^2 n)}, n^{o_{\text{eff}}(1)}] \) by a trivial padding argument, but the reverse implication is also probable in the sense that any proof of the latter would likely scale to prove the former. However, it is still possible that an algorithm working in polynomial space refutes the stronger statement even though \( \text{NL} \not\subseteq \text{D}[2^{o_{\text{eff}}(\log^2 n)}, n^{o_{\text{eff}}(1)}] \).

**4 Treedepth**

In this section we sketch the proof of Theorem 3. Let \( s : \mathbb{N} \rightarrow \mathbb{N} \) be a nice function. First, we discuss more precisely the model of machines used to define class \( \text{NAuxSA}[\text{poly}, \log, s] \).

The machine has three tapes, each using a fixed, finite alphabet \( \Sigma \): a read-only input tape, a working tape of length \( \mathcal{O}(\log n) \), and a stack tape of length \( s(n) \). On each of the tapes there is a head; the transitions of the machine depend on its state and the triple of symbols under the heads. The input tape is read-only. The stack tape can be read but not freely written on; instead, the transitions of the machine may contain instructions \( \text{push} \sigma \) or \( \text{pop} \), working naturally. Since \( s \) is nice, \( s(n) \leq \text{poly}(n) \) so within the working tape the machine can keep track of the current height of the stack and the indices on which the heads are positioned.

We start the proof of Theorem 3 by showing containment, exemplifying how the resources are used. The idea is to perform a depth-first search of the tree-depth decomposition, guessing the color of each entered vertex, pushing it onto the stack and popping it when withdrawing from the vertex. Thus, the stack maintains the guessed colors on the path from the current vertex to the root, allowing correctness to be checked.
Lemma 16 (♣). For any nice \(s(n)\), \text{td-3Coloring}[s] is in \(\text{NAuxSA}[\text{poly}, \log, s]\).

Clearly if \(s(n) \geq \log^2 n\), \(\text{NAuxSA}[\text{poly}, \log, s] \subseteq \text{NAuxSA}[\text{poly}, s/\log, s]\).

The next step is to show how the stack operations of the latter class’ machines can be regularized. This idea originates in the approach of Akatov and Gottlob [3]. Following [3], we define a regular stack machine in the following way. For any valid sequence \(S\) of push/pop operations that starts and ends with an empty stack, define the corresponding push-pop tree \(\tau(S)\) to be the ordered tree (a rooted tree with an order imposed on the children of each node) in which a depth-first search would result in the sequence \(S\), where entering/withdrawing from a vertex corresponds to a push/pop operation. We say that a language is in \(\text{reg-NAuxSA}[\text{poly}, s/\log, s]\) if it is recognized by an NTM \(M\) with \(s(n)/\log(n)\) working space and an auxiliary stack of height \(s(n)\) that has the following properties:

1. \(M\) pushes and pops blocks of \(b = \lceil s(n)/\lg(n) \rceil\) symbols at a time, say, simultaneously from/to the first \(b\) positions of the worktape.
2. Whenever \(M\) decides to push or pop, it can only change its state. Moreover, the decision about using a push or pop transition depends only on the machine’s state.
3. If \(M\) accepts input \(a\), then there is a run on \(a\) where the push-pop tree (where pushing/popping a block is considered atomic) is the full binary tree of depth exactly \(c[\lg n]\), for some fixed integer \(c\). In particular, at the moment of accepting the stack is empty.

Restriction (2) is a technical adjustment. Restriction (1) is easily achieved by simulating the top symbols from the stack in a length \(b\) buffer on the working tape, pushing and popping a full buffer when needed. The most important restriction is (3): the push-pop tree has a fixed shape of a full binary tree. For this, we use the following observation of Akatov and Gottlob [3, 2], used also by Elberfeld et al. [20]. The traversal ordering of the nodes of an ordered tree is the linear ordering which places a parent before its children and, for children \(a, b\) of a node, \(a\) occurring before \(b\), places all descendants of \(a\) before all descendants of \(b\).

Lemma 17 (Lemma 3.3 of [2]; Theorem 3.14 of [20]). Given an ordered tree \(T\) with \(n\) nodes and depth at most \(\lg n\), one can in logarithmic space compute an embedding (an injection that preserves the ancestor relation and traversal ordering) into a full binary tree of depth \(4\lg n\).

As in [3], this allows us to regularize our machines, as dummy pushes/pops can be non-deterministically guessed so that the push-pop tree of at least one run is a full binary tree.

Lemma 18 (♣). \(\text{NAuxSA}[\text{poly}, s/\log, s] \subseteq \text{reg-NAuxSA}[\text{poly}, s/\log, s]\).

Knowing that computations for \(\text{NAuxSA}\) can be conveniently regularized, we can describe the existence of such a computation by a CNF formula “wrapped around” the rigid shape of the full binary tree that encodes the push-pop tree of the run. We think of the computation as starting at the root node, moving down an edge whenever a push is made and moving up an edge whenever a pop is made. This was also the idea of Akatov and Gottlob [3], but our reduction needs to introduce many more elements, in particular copies of the gadget of Lemma 11 for every fragment between two push/pop operations. Each part of the computation depends only on symbols pushed onto the stack on the path to the root. This, together with the \(O\left(\frac{n}{\log n}\cdot \log n\right) = O(s(n))\) bound on the tree-depth of the computation gadget, will give rise to a tree-depth decomposition of depth \(O(s(n))\) of the obtained formula’s primal graph.
Lemma 19 (♠). If \( L \in \text{reg-NAuxSA}[\text{poly}, s/\log, s] \), then \( L \leq_{\text{td}} \text{CNF-SAT}[s] \).

Lemmas 18 and 19 show that \( \text{td-CNF-SAT}[s] \) is hard for \( \text{NAuxSA}[\text{poly}, s/\log, s] \), and by Lemma 10 so is \( \text{td-3COLORING}[s] \). Since the closure of \( \text{NAuxSA}[\text{poly}, \log, s] \) under logspace reductions is \( \text{NAuxSA}[\text{poly}, \log, s(\text{poly})] \), Lemmas 16, 18, 19 give the following chain of containments (here \([A]^L\) denotes the class of problems reducible to \( A \) in logspace):

\[
[t\text{-3COLORING}[s(\text{poly})]^L] \subseteq [t\text{-3COLORING}[s]^L] \subseteq \text{NAuxSA}[\text{poly}, s(\text{poly})]/\log, s(\text{poly})] \\
\subseteq [t\text{-3COLORING}[s(\text{poly})]^L] \\
\subseteq [\text{NAuxSA}[\text{poly}, s(\text{poly})]]^L \\
\subseteq [\text{NAuxSA}[\text{poly}, s(\text{poly})]]^{\log_2}\text{time}
\]

Therefore, all containments must be equalities, which concludes the proof of Theorems 3 and 4. Now, to prove the determination of Theorem 5, we only need an algorithm for \( \text{td-3COLORING}[s] \). The following lemma implies \( [t\text{-3COLORING}[s]]^L \subseteq \text{NAuxSA}[\text{poly}, s(\text{poly})] \), hence Theorem 3, and the fact that \( D[s(\text{poly})] \) is closed under logspace, yield Theorem 5.

Lemma 20 (♠). \( \text{td-3COLORING}[s] \) can be solved in time \( 3^s \cdot \text{poly}(n) \) and space \( O(s + \log n) \).

Characterization via alternating machines. In the full version of this paper, we use Theorem 3 to give another characterization in terms of alternating Turing machines with polynomial size of an accepting tree, i.e. \( \text{treesize} \). Both the notions of ATMs and that of \( \text{treesize} \) later introduced by Ruzzo [42] gave a new unified view on various complexity classes, simplifying a few containment proofs. Ruzzo showed that \( \text{NAuxPDA}[\text{poly}, s, \text{poly}^2] = \text{A}[s, \text{poly}^2, \text{poly}] \).

We show that bounding the time (as opposed to space) of a polynomial \( \text{treesize} \) ATM, leads to the classes corresponding to small tree-depth, as opposed to small treewidth.

Theorem 21 (♠). Let \( s(n) \geq \log^2 n \) be a nice function. Then

\[
\text{NAuxSA}[\text{poly}, \log, s(\text{poly})] = \text{A}[s(\text{poly}), \text{poly}] \\
\text{treesize}
\]

5 Conclusions

Let \( s(n) \geq \log^2 n \) be a nice function such that \( \forall c, s(nc) = O(s(n)) \) (e.g. \( s(n) = \log^k n, k \geq 2 \)). The hierarchy of graph parameters of Corollary 8 together with Theorems 1, 3, and 21 implies the following hierarchy of complexity classes between \( \text{NL} \) and \( \text{NP} \).

\[
\begin{align*}
\text{NAuxSA}[\text{poly}, \log, s] & = [\text{td-3COLORING}[s]]^L = \text{A}[s, \text{poly}] \subseteq \text{D}[s] \text{time} \\
\text{N}[\text{poly}, s] & = [\text{pw-3COLORING}[s]]^L = \text{N}[\text{poly}, s] \text{time} \\
\text{NAuxPDA}[\text{poly}, s] & = [\text{tw-3COLORING}[s]]^L = \text{A}[s, \text{poly}] \subseteq \text{D}[2^{O(s)}] \text{time} \\
\text{NAuxSA}[\text{poly}, \log, s \cdot \log] & = [\text{td-3COLORING}[s \cdot \log]]^L = \text{A}[s, \log, \text{poly}] \subseteq \text{D}[s \cdot \log] \text{time}
\end{align*}
\]

For \( s(n) = \log^k n \), the classes have been considered under different names:
NAuxSA[poly, log , log^k] was named DC^{k-1} (for divide and conquer) in [3, 2],

N[poly, log^k] are known as NSC^k (the non-deterministic variant of Steve’s Class),

NAuxPDA[poly, log^k] is shown equal to a class named SAC^k quasi in [4].

This yields the following hierarchy:

\[
L \subseteq \text{NSC}^1 \subseteq \text{SAC}^1 \quad \text{quasi} \subseteq \text{DC}^{1} \subseteq \cdots \subseteq \text{DC}^{k-1} \subseteq \text{NSC}^k \subseteq \text{SAC}^k \quad \text{quasi} \subseteq \text{DC}^{k} \subseteq \cdots \subseteq \text{NP}
\]

References


