Survey of Space Complexity in Relation to Reachability Problems

Jason Vander Woude

I. Preliminaries

Before surveying a number or results in space complexity, we introduce definitions of some of the important classes which will be discussed throughout. The definitions in this paper are adapted from [1], [2]. We also note that the surveys [3], [4] served as a starting point for understanding and organizing the results presented here.

Definition 1. *Let $s : \mathbb{N} \to \mathbb{N}$ and $L \subseteq \{0,1\}^*$. We say that $L \in \mathbf{DSPACE}(s(n))$ if there exists a deterministic Turing machine (DTM) M deciding L such that the computation of M on input x accesses at most $O(s(|x|))$ cells of the work tapes for all $x \in \{0,1\}^*$. Similarly, we say that $L \in \mathbf{NSPACE}(s(n))$ if there exists a non-deterministic Turing machine (NDTM) M deciding L that accesses at most $O(s(|x|))$ cells of the work tapes for any non-deterministic computation for all $x \in \{0,1\}^*$.*

Many of the results that we discuss in this paper deal with the particular parameterization of the above classes where $s(n) = \log(n)$ representing logspace computable languages and non-deterministic logspace computable languages. These classes are given special names.

Definition 2. *The class $\mathbf{DSPACE}(\log(n))$ is denoted by $\mathbf{L}$, and the class $\mathbf{NSPACE}(\log(n))$ is denoted by $\mathbf{NL}$.*

Another useful class, $\mathbf{UL}$, standing for unambiguous logspace, is the collection of languages in $\mathbf{NL}$ which can be decided in logspace by a NDTM which has at most one accepting computation path. In other words, a NDTM which has no accepting paths on inputs not in the language, and exactly one accepting computation path on inputs belonging to the language. Formally, this is as follows.

Definition 3. *Let $L \subseteq \{0,1\}^*$. We say that $L \in \mathbf{UL}$ if there exists a NDTM M deciding L such that every computation of M on input x accesses at most $O(\log|x|)$ cells of the work tapes for all $x \in \{0,1\}^*$, and M has exactly one computation path if $x \in L$.*

Some results below will simultaneously consider the time and space complexity of a language with respect to a single Turing machine (TM), which gives motivation for the definition of the class $\mathbf{TISP}$ (standing for time in space).

Definition 4. *Let $s, t : \mathbb{N} \to \mathbb{N}$ and $L \subseteq \{0,1\}^*$. We say that $L \in \mathbf{TISP}(t(n), s(n))$ if there exist a DTM M deciding L such that the computation of M on input x accesses at most $O(s(|x|))$ cells of the work tapes and runs for at most $O(t(|x|))$ steps for all $x \in \{0,1\}^*$.*

Along with these classes come various notions of reducibility. Reductions between problems in these classes should at the very least occur in logspace, but even the identity function cannot be computed in logspace because if the input has n bits, then output will require n bits. This leads to the following definition of an implicit logspace computable function. Essentially, a function is implicitly logspace computable if the length of the output is logspace computable and each bit of the output is logspace computable.

Definition 5. *A function $f : \{0,1\}^* \to \{0,1\}^*$ is implicitly logspace computable if there exists c such that $|f(x)| \leq |x|^c$ for all $x \in \{0,1\}^*$ and the languages $L_f = \{\langle x, i\rangle : f(x)_i = 1\}$ and $L'_f = \{\langle x, i\rangle : i \leq |f(x)|\}$ are in $\mathbf{L}$.*

Definition 6. *A language B is said to be logspace reducible to language C, denoted $B \leq_l C$, if there is an implicitly logspace computable function $f : \{0,1\}^* \to \{0,1\}^*$ and $x \in B$ if and only if $f(x) \in C$ for all $x \in \{0,1\}^*$. C is called $\mathbf{NL}$-hard if for every $B \in \mathbf{NL}$, $B \leq_l C$, and C is called $\mathbf{NL}$-complete if C is $\mathbf{NL}$-hard and $C \in \mathbf{NL}$.*

We will also soon need the definition of a space constructible function, so we present that definition as well.

Definition 7. *A function $s : \mathbb{N} \to \mathbb{N}$ is space constructible if there is a DTM that given $x \in \{0,1\}^*$ as input, computes $s(|x|)$ and accesses at most $O(s(|x|))$ cells of the work tapes.*

The last preliminary definition is that of the co-classes.

Definition 8. *If* $\mathbf{CLASS}$ *is a class of languages, then define* $\mathbf{coCLASS} = \{L \subseteq \{0,1\}^* \colon \{0,1\}^* \setminus L \in \mathbf{CLASS}\}$.

A final note is that throughout this survey we will use the language of the literature and use the term graph always to denote a directed graph. In the case that we discuss undirected graphs, we say so explicitly. Note that every undirected graph can be encoded as a directed graph by using two directed arcs for each edge of the undirected graph.

II. Foundational Results

One of the earliest and most foundational results in the area of space complexity is due to Savitch in 1970 showing that $\mathbf{NSPACE}(s(n)) \subseteq \mathbf{DSPACE}(s^2(n))$ with only mild restrictions on $s(n)$. In other words, though it is widely believed that non-determinism offers super-polynomial power over determinism in regards to time complexity (i.e. it is believed that $\mathbf{P} \neq \mathbf{NP}$), this result shows that with respect to space complexity, non-determinism offers no more than quadratic savings over a deterministic algorithm (for mild restrictions on $s(n)$).

Theorem 9. *[5] Let $s : \mathbb{N} \to \mathbb{N}$ be a space constructible function with $s(n) \geq \log n$. Then* $\mathbf{NSPACE}(s(n)) \subseteq \mathbf{DSPACE}(s^2(n))$.

Savitch's proof of this theorem relies on the fact that every TM (whether deterministic or non-deterministic) has a configuration graph associated with each input, and the TM will accept if and only if there is a directed path from the initial configuration to a unique accepting configuration in the configuration graph of a given input.

If a NDTM M uses $O(s(n))$ space, then for some fixed c, there are at most $c^{s(n)}$ different configurations of M on input x where $n = |x|$. A configuration is specified by the position of all the heads (requiring $O(\max(\log n, \log s(n)))$ bits), the current state ($O(1)$ bits), and the contents of the work tapes ($O(s(n))$ bits). Thus the configuration graph has at most $c^{s(n)}$ nodes and Savitch's algorithm can solve this directed graph connectivity problem in space $O(\log^2(c^{s(n)})) = O(s^2(n))$ using the pseudo-code of Algorithm 1. Note that as written, Algorithm 1 determines if there is a path of length at most k from s to t in an input graph. This suffices to prove the claim above if this task can be done in $O(\log^2(\eta))$ space where η denotes the number of vertices in the input graph, s represents the initial configuration, t represents the accepting configuration, and k is initialized to η.

Algorithm 1 Find a directed path in a graph of length at most k

```
1: procedure FindDirectedPath($(V,E), s, t, k$)
2:     if $k == 0$ then
3:         return $s == t$
4:     if $k == 1$ then
5:         return $s == t$ OR $(s,t) \in E$
6:     for $v \in V$ do
7:         if FindDirectedPath($(V,E), s, v, \lfloor \frac{k}{2} \rfloor$) AND FindDirectedPath($(V,E), v, t, \lceil \frac{k}{2} \rceil$) then
8:             return TRUE
9:     return FALSE
```

The space complexity of Algorithm 1 is $O(\log^2(\eta))$ because the call depth is $\log(\eta)$ since the path size is cut in half at each recursive call, and the space needed to store the variables s and t in each call is $\log(\eta)$.

The idea of configuration graphs is relevant to understanding the class $\mathbf{NL}$ because they can be used to show that determining if there is an st-path in a graph is an $\mathbf{NL}$-complete problem and thus characterizes $\mathbf{NL}$. The proof involves the idea that a NDTM M can be naturally reduced to an st-graph as above by considering the configuration graph of M with s being the start configuration and t the accepting configuration, and a graph can be naturally reduced to a NDTM by considering a NDTM determining reachability in graphs.

Definition 10. *Under some fixed binary string representation of graphs (including adjacency matrix or incidence matrix), with specified vertices s and t, the language* STCONN *is the set of all strings representing graphs containing a directed path from s to t, and the language* USTCONN *is the set of all strings representing undirected graphs containing a path between s and t.*

Theorem 11. STCONN *is* $\mathbf{NL}$*-complete.*

Another major result due independently to Immerman and Szelepcsènyi around 1988 is that $\mathbf{NSPACE}(s(n))$ is the same as $\mathbf{coNSPACE}(s(n))$ for $s(n) \geq \log n$; this is true even if s is not space constructible. The proof technique is to take any language in $\mathbf{NSPACE}(s(n))$

along with a NDTM M deciding it and describe an $\mathbf{NSPACE}(s(n))$ machine M' which has an accepting computation if and only if M does not, and so $\mathbf{NSPACE}(s(n))$ is closed under language complement.

Theorem 12. *[6] Let $s : \mathbb{N} \to \mathbb{N}$ be a function such that $s(n) \geq \log n$. Then $\mathbf{NSPACE}(s(n)) = \mathbf{coNSPACE}(s(n))$ (even if $s(n)$ is not space constructible).*

Corollary 13. $\mathbf{NL} = \mathbf{coNL}$.

We present pseudocode demonstrating the technique of Immerman and Szelepcsènyi, but we restrict to showing that $\mathbf{NL} = \mathbf{coNL}$ by showing that the complement of STCONN is in $\mathbf{NL}$. That is, Algorithm 2 is a non-deterministic logspace algorithm that accepts if and only if there is no st-path in $G = (V, E)$. The idea of the algorithm is to first iteratively (or inductively) count how many vertices can be reached from s for each possible path length (1 up to $n-1$) to determine the total number of vertices reachable from s. This is accomplished in lines 1-20. The next step is then to guess a path from s to every other vertex and ensure a valid path was guessed to every reachable vertex by comparing the counts. Thus, if there is a path to t, then it is guessed and the algorithm rejects. This means the algorithm can only accept if there is no path to t, and it does accept if there is no path to t. Further, this is logspace algorithm because it is non-recursive, and the only variables are counters.

Algorithm 2 Immerman-Szelepcsènyi non-deterministic algorithm for solving complement of STCONN in logspace

```
1:  procedure AcceptIfNoPathExists($(V, E), s, t$)
2:      $n \leftarrow |V|$
3:      $prevCount \leftarrow 0$
4:      $count \leftarrow 1$
5:      for $d \in \{1, 2, \ldots, n-1\}$ do
6:          $prevCount \leftarrow count$
7:          $count \leftarrow 0$
8:          for $target \in V$ do
9:              $targetFound \leftarrow$ FALSE
10:             $runningPrevCount \leftarrow 0$
11:             for $prevTarget \in V$ do
12:                 $visiting \leftarrow$ EndVrtx(GuessPath($s, prevTarget, d-1$))
13:                 if $visiting == prevTarget$ then
14:                     $runningPrevCount += 1$
15:                     if $\langle prevTarget, target\rangle \in E$ then
16:                         $targetFound \leftarrow$ TRUE
17:             if $runningPrevCount \neq prevCount$ then
18:                 REJECT
19:             if $targetFound$ then
20:                 $count \leftarrow count + 1$
21:     $countVerification \leftarrow 0$
22:     for $target \in V$ do
23:         $visiting \leftarrow$ EndVrtx(GuessPath($s, target, n-1$))
24:         if $visiting == target$ then
25:             $countVerification += 1$
26:             if $target == t$ then
27:                 REJECT
28:     if $countVerification \neq count$ then
29:         REJECT
30:     ACCEPT
```

III. Undirected Graphs

One of the important open questions in space complexity research, and possibly the most fundamental, is whether $\mathbf{L} = \mathbf{NL}$, or equivalently, whether STCONN $\in \mathbf{L}$. One attempt at answering this question in the affirmative is to develop efficient algorithms for solving reachability, and one possible approach to answering in the negative would be to show that USTCONN $\notin \mathbf{L}$, because if undirected connectivity cannot be solved in logspace, then directed connectivity cannot either.

In a very significant result of the last decade, Reingold showed that the statement in the latter approach is false.

Theorem 14. *[7]* USTCONN $\in \mathbf{L}$.

This leaves open the question of whether STCONN $\in \mathbf{L}$. Work following the initial proof of the above theorem demonstrated that it can be extended beyond undirected graphs to regular directed graphs and Eulerian directed graphs.

Definition 15. *A directed graph is said to be D-regular if each vertex has D arcs into the vertex (indegree D) and D arcs out of the vertex (outdegree D) and is called regular if there exits a D such that it is D-regular. A directed graph is said to be Eulerian if the indegree and outdegree of each vertex is the same.*

Note that regular graphs are a special case of Eulerian graphs.

Theorem 16. *[8] There exists a logspace algorithm for deciding reachability in Eulerian (and regular) directed* *graphs.*

The authors note that this suggests an interesting property about Reignold's original algorithm; the important property of undirected graphs is not that the traversal is reversible (i.e. each edge can be traversed in either direction), but that without loss of generality every undirected graph can be assumed to be regular.

With limited progress in determining if STCONN $\in$ $\mathbf{L}$ other than the above result, much research has focused on particular classes of graphs—especially planar graphs and a further restriction of planar graphs known as grid graphs.

IV. Restricted Graphs

A planar graph is one which can be embedded on the plane with no crossing edges. This is the extent to which we offer a definition because a proper definition of planarity requires the use topology, and that is outside the scope of this survey. Computationally, planar graphs can be conveniently represented by a combinatorial embedding—a specification for each vertex of the clockwise ordering of the incident arcs. However, while the combinatorial embedding is a convenient way to algorithmically exploit the planarity of a graph, it is not the typical encoding of a graph. For this reason, the complexity of determining a planar embedding (or determining that a graph is not planar) given an adjacency matrix was studied initially in [9]. Eventually, the space complexity was pinned down between two well-studied classes.

Theorem 17. *[10] Given an adjacency matrix of a graph, the problem of finding a combinatorial embedding is hard for* $\mathbf{L}$ *but lies in* $\mathbf{SL}$.[1]

The class $\mathbf{SL}$ was one of historical importance defined specifically to understand the space complexity of reachability in undirected graphs, but the above answer was nearly satisfactory as the conjecture of [11] that $\mathbf{SL} = \mathbf{L}$ was known at the time. Nearly a decade later, the result of Reingold (Theorem 14) proved as a corollary that $\mathbf{SL} = \mathbf{L}$ thus resolving the above with satisfaction.

Corollary 18. *[10], [7] Given an adjacency matrix of a graph, the problem of finding a combinatorial embedding is* $\mathbf{L}$*-complete.*

[1]Because the notions of $\mathbf{L}$-hard and $\mathbf{L}$-complete in this survey are limited to this theorem and the next, we do not present a definition of these here and simply note that this is with respect to the typical $\mathbf{AC}^0$ reductions for $\mathbf{L}$.

This result implies that is unimportant to specify a combinatorial embedding when considering reachability algorithms on planar graphs because a graph can be specified in the standard form of an adjacency matrix or incidence matrix, and a planar embedding can be determined in logspace.

A further restriction of planar graphs are grid graphs which are those planar graphs specified in such a way that the vertices lie on a grid and two vertices can only be adjacent if they lie on adjacent intersections of the grid.

Definition 19. *A grid graph is a planar graph in which each vertex is identified with an ordered pair of natural numbers, and two vertices can be adjacent only if the first coordinate of the ordered pairs is the same and the second coordinate differs by one, or if the second coordinate of the ordered pairs is the same and the first coordinate differs by one.*

Grid graphs are a convenient restriction to study because when traversing the graph, there is a sense of consistent directionality. An early result showed that this directionality made traversing graphs strictly easier in some models of computation. It was shown that visiting all vertices of undirected grid graphs is easier than visiting all vertices of undirected 3-regular graphs for certain modified automata. Further, the proof can be used to construct a logspace polynomial time TM to solve reachability in such graphs; this in itself was significant as it predated the more general results of Reingold (Theorem 14) by roughly three decades. We state the result, but leave any definitions for the reader to find in the original paper as they will not be used in the remainder of this survey.

Theorem 20. *[12] There is a finite automaton with 2 pebbles which can visit all vertices of any undirected grid graph, but there is no such automaton which can traverse all vertices of arbitrary 3-regular graphs.*

Corollary 21. *[12] Undirected grid graph reachability is in* $\mathbf{TISP}(n^2, \log(n))$.

We make note of a clever technique used in the proof. The authors wanted to detect whether a certain cycle in the graph was traversed clockwise or counter-clockwise, and this could be done by counting the number of right turns minus the number of left turns since all turns are 90 degrees in a grid graph. The value would be 4 or -4 if the traversal was clockwise or counter-clockwise respectively. Using a typical counter (which is not available to automata) would require non-constant space. However, noting that -4 and 4 are distinct mod 3, a modular counter can be implemented in the finite states of the automaton to detect this property.

While the feature of having a consistent sense of direction made undirected reachability strictly easier than general undirected reachability in the computation model above, it turns out that the same does not hold with respect to planar graphs in the TM model, even in the directed case. Grid graph reachability is trivially logspace reducible to planar reachability, because grid graphs are planar, but the converse is true as well.

Definition 22. *Under some fixed binary string representation of graphs (including adjacency matrix or incidence matrix), with specified vertices s and t, the language* $\mathrm{PLANARREACH}$ *is the set of all strings representing planar graphs containing a directed path from s to t, and the language* GGR *is the set of all strings representing grid graphs containing a directed path from s to t.*

Theorem 23. *[13]* $\mathrm{PLANARREACH} \leq_l \mathrm{GGR}$.

This gives even more motivation for studying grid graphs, as the added structure makes them more convenient than general planar graphs. In [14], the authors study a variety of further restricted types of grid graphs. They address two local restrictions on GGR: instances where each vertex has outdegree at most 1 (1GGR) and instances where each vertex has indegree at most 1 and outdegree at most 1 (11GGR). They also discuss two global restrictions: the first is that both s and t are on the boundary of the graph (GGRB), and the second is that the graph is layered such that all horizontal edges are oriented eastward and all vertical edges are oriented southward (LGGR). By consider pairwise combinations of these restrictions, this gives nine restricted grid graph type reachability problems. Further considering the undirected case (UGGR) and the undirected boundary case (UGGRB) results in the eleven[2] languages presented in Table I.

The authors proved that these eleven problems are really at most five unique problems with respect to first order projections (a type of reduction weaker than logspace reductions or even $\mathbf{AC}^0$ reductions). This reduction is denoted with $\leq_{\mathrm{proj}}^{\mathrm{FO}}$. They do so by showing six of them are all equivalent, and another pair are also equivalent. The following two theorems are summarized by the shading of Table I.

Theorem 24. *[14]* $\mathrm{1GGR} \leq_{proj}^{FO} \mathrm{11GGR} \leq_{proj}^{FO} \mathrm{UGGR} \leq_{proj}^{FO} \mathrm{UGGRB} \leq_{proj}^{FO} \mathrm{11GGRB} \leq_{proj}^{FO} \mathrm{1GGRB} \leq_{proj}^{FO} \mathrm{1GGR}$.

Theorem 25. *[14]* $\mathrm{GGR} \leq_{proj}^{FO} \mathrm{GGRB} \leq_{proj}^{FO} \mathrm{GGR}$.

GGR	1GGR	11GGR	UGGR
GGRB	1GGRB	11GGRB	UGGRB
LGGR	1LGGR	11LGGR	

TABLE I
ELEVEN TYPES OF GRID GRAPH REACHABILITY WITH SHADED FIRST ORDER PROJECTION EQUIVALENCIES.

The authors suggest that there is weak evidence for believing that GGRB (and thus GGR) is not $\mathbf{NL}$-complete because there is a known relatively simple first order projection from GGRB to its complement, but the reduction from general $\mathbf{NL}$ problems to their complements by the Immerman- Szelepcsènyi algorithm (Theorem 12, Algorithm 2) seems more complicated.

The authors also indicate that while it seems reasonable to believe that LGGR would be easier than the more general GGR (because if there is a path from s to t, it must progress only eastward and southward), they found no evidence of this. At the time of publication, it was known that $\mathrm{LGGR} \in \mathbf{UL} \subseteq \mathbf{NL}$, but many researches believe it is feasible that $\mathbf{UL} = \mathbf{NL}$ and thus, this would not separate the complexity of LGGR from that of GGR.

Theorem 26. *[13]* $\mathrm{LGGR} \in \mathbf{UL}$.

In fact, it is known that the classes $\mathbf{UL}$ and $\mathbf{NL}$ are equivalent in the nonuniform case. By nonuniform we mean that the TM deciding a language has access to a common advice string for all inputs of the same length.

Definition 27. $\mathbf{UL}/poly$ *is the class of decision problems solvable by a* $\mathbf{UL}$ *NDTM that receives an additional input string y called an advice string which depends only on the size $|x|$ of the input, and such that $|y| \leq |x|^{O(1)}$. Similarly,* $\mathbf{NL}/poly$ *is the class of decision problems solvable by an* $\mathbf{NL}$ *NDTM that receives an additional input string y called an advice string which depends only on the size $|x|$ of the input, and such that $|y| \leq |x|^{O(1)}$.*

Theorem 28. *[15]* $\mathbf{UL}/poly = \mathbf{NL}/poly$.

It was later shown that $\mathrm{GGR} \in \mathbf{UL}$, and thus the classification above of $\mathrm{LGGR} \in \mathbf{UL}$ does not help to resolve if LGGR and GGR differ in space complexity.

[2]It does not make sense to consider the layered version of undirected reachability.

Theorem 29. *[16]* GGR $\in$ **UL**.

Using Theorem 23, and the fact that **UL** is closed under complement proves the following corollary.

Corollary 30. *[16]* PLANARREACH $\in \mathbf{UL} \cap \mathbf{coUL}$

V. Space and Time

While some research is focused on the grid graph restriction when considering logspace algorithms, STCONN in its generality remains an interesting language to study because it is **NL**-complete. Broadly, in the field of complexity, the relation between simultaneous time and space complexity is of much interest, but little is known about the trade-offs between the two notions of complexity. In the context of reachability, Savitch's Theorem (Theorem 9) demonstrates the existence of sublinear space algorithms for deciding STCONN, but the algorithm of this construction runs in time $2^{\Omega(\log^2(n))}$. In the realm of time complexity, this is not desirable as algorithms are generally considered efficient if they run in polynomial time. It is also known that reachability can be solved in linear time and linear space using the breadth first search algorithm.

Theorem 31. STCONN $\in \mathbf{TISP}(n, n)$

A natural question, then, is whether there is a sublinear space algorithm for STCONN which also runs in polynomial time. The first affirmative answer to this question came in 1998.

Theorem 32. *[17] There is a sublinear space, polynomial time algorithm for* STCONN. *In particular* STCONN $\in \mathbf{TISP}(n^{O(1)}, \frac{n}{2^{O(\sqrt{\log n})}})$.

As before, by restricting attention to particular classes of graphs, better bounds can be obtained, and again, planar graphs and grid graphs are a natural place to start. However, it is not known if these two types of graphs can be considered equivalent with respect to simultaneous time and space bounds because the known logspace algorithms for reducing planar graphs to grid graphs may increase the size of the graph by a polynomial factor and thus affect the property of sublinearity of the space bound.

It was shown in [18] that for general planar graphs, reachability could be determined in polynomial time and $O(n^{1/2+\epsilon})$ space for any $\epsilon > 0$; this bound was later improved.

Theorem 33. *[19]* PLANARREACH $\in \mathbf{TISP}(n^{O(1)}, n^{1/2} \log^{O(1)}(n))$.

A famous result in graph theory is that a graph is planar if and only if it contains no copy of K_5 or $K_{3,3}$ as a subgraph; these properties are referred to as K_5-free and $K_{3,3}$-free respectively. Here K_5 denotes the complete graph on five vertices and $K_{3,3}$ denotes the complete bipartite graph with three vertices in each part. Complexity bounds for planar graph problems can sometimes (though not always) be extended to graphs that are either K_5-free or $K_{3,3}$-free, without requiring both properties. This was shown to be the case with reachability.

Theorem 34. *[20] Reachability for a directed graph which is either K_5-free or $K_{3,3}$-free can be decided in* $\mathbf{TISP}(n^{O(1)}, n^{1/2+\epsilon})$.

The authors also mention that it was known prior to their work that reachability in graphs which are either K_5-free or $K_{3,3}$-free logspace reduces to the planar case as in the next theorem. However, this reduction increases the size of the graph by a polynomial factor, and thus is insufficient for obtaining the space bound they achieved.

Theorem 35. *[21] Reachability for K_5-free graphs and for $K_{3,3}$-free graphs logspace reduces to* PLANARREACH.

The above result can be viewed as extending Corollary 30 showing that **UL** contains problems that are intuitively even more difficult than PLANARREACH and thus serves as a step in trying to collapse **NL** to **UL**.

The above results show that reasonable bounds can be obtained without relying on the full property of planarity in graphs. Another way to lessen the planarity restriction is to consider graphs which, though not embeddable on the plane, can be embedded on more complicated surfaces. Topologically, the genus of a surface can roughly be described as how many "holes" there are. For example, a plane and a sphere have genus 0, a torus has genus 1, and there are graphs which cannot be embedded on the plane without crossing edges but can be embedded on a torus without crossing edges. Letting g denote the genus of a graph, the following gives a space bound for polynomial time algorithms solving reachability.

Theorem 36. *[20] Reachability can be decided in* $\mathbf{TISP}(n^{O(1)}, n^{2/3} \log^{O(1)}(n) \cdot g^{1/3} \log^{O(1)}(g))$ *provided that the input is a combinatorial embedding of the graph on a genus g surface.*

We mentioned earlier that in the case of planar graphs, a combinatorial embedding can be determined in logspace, but this is not known to be possible in the case of higher genus graphs. In fact, it has been shown that even determining the numeric value of the genus of a graph is $\mathbf{NP}$-complete, and this does not consider actually finding a combinatorial embedding. Because it is consistent with current knowledge that $\mathbf{L} = \mathbf{NP}$ (as will be discussed in Section VI), this does not actually rule out that the genus cannot be determined in logspace; however, if $\mathbf{P} \neq \mathbf{NP}$ as is widely believed, then it is impossible to find the genus of general graphs in logspace.

Theorem 37. *[22] Given a graph G and natural number k, it is $\mathbf{NP}$-complete to determine if the genus of G is less than or equal to k.*

By restricting consideration to grid graphs, these time in space bounds can be improved. The first such simultaneous time and space bound for GGR was given in [23] where it was shown that GGR $\in \mathbf{TISP}(n^{O(1)}, n^{1/2+\epsilon})$ for any $\epsilon > 0$. This bound was later improved to GGR $\in \mathbf{TISP}(n^{O(1)}, n^{1/3}\log^{O(1)}(n))$ in [24], and the most recent result, within the past year, reduces the space bound even further.

Theorem 38. *[25] For any $\epsilon > 0$,* GGR $\in \mathbf{TISP}(n^{O(1)}, n^{1/4+\epsilon})$.

VI. Other Results

A. JAG and NNJAG Models

Another important open question is whether the square bound of Savitch's Theorem (Theorem 9) is tight. While this question is open for the TM model of computation, it has been resolved for the restricted JAG (Jumping Automata on Graphs) and NNJAG models of computation defined in [26] and by Poon in [27]. While this is an interesting result, so far it has not led to advances in answering the question for the TM model.

Theorem 39. *[28] Deciding* STCONN *requires $\Omega(\log^2(n))$ space in the JAG and NNJAG models of computation.*

B. Hartmanis

Not only has it been difficult to resolve the question of whether $\mathbf{L} = \mathbf{NL}$, but it has been difficult to resolve class similarities or differences which should be easier than this. Some of the class containments that are known are as follows, where most of the inclusions are trivial and come directly from the definitions of the classes.[3]

$$\mathbf{L} \subseteq \mathbf{UL} \subseteq \mathbf{NL} \subseteq \mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE}$$

However, the only proper containment in the above chain that is known is

$$\mathbf{NL} \subsetneq \mathbf{NPSPACE} = \mathbf{PSPACE}$$

The equality comes from Savitch's Theorem (Theorem 9), and the proper inclusion is from the well known space hierarchy theorem. This is to say that not only is it unknown whether $\mathbf{L} = \mathbf{NL}$, but it is currently an open question whether $\mathbf{L} = \mathbf{P}$ or even $\mathbf{L} = \mathbf{NP}$.

In the realm of time complexity, one of the biggest questions is whether $\mathbf{P} = \mathbf{NP}$, and while any solution to this question still seems a long way off, one way to understand the problem at a deeper level is to understand how the question relates to sparse sets—sets which have at most $O(1) \cdot n^{O(1)}$ strings of length n. Mahaney proved in [29] that if there exists a sparse $\mathbf{NP}$-hard set, then $\mathbf{P} = \mathbf{NP}$, where by $\mathbf{NP}$-hard we mean a set that is hard under polynomial time many-one reductions. A space analog of this was originally conjectured by Hartmanis in [30] and eventually proved by Cai and Sivakumar.

Theorem 40. *[31] If there exists a sparse hard set for $\mathbf{P}$ under logspace many-one reductions, then $\mathbf{L} = \mathbf{P}$.*

Hartmanis also made a similar conjecture in regards to the equality of $\mathbf{L}$ and $\mathbf{NL}$. This too was proven by Cai and Sivakumar.

Theorem 41. *[32] There exists a sparse hard set for $\mathbf{NL}$ under logspace many-one reductions if and only if $\mathbf{L} = \mathbf{NL}$.*

C. Randomized Space

So far, the classes discussed have been deterministic or non-deterministic, but a third broad category of both theoretical and practical significance is randomized computation.

Definition 42. *Let $s : \mathbb{N} \to \mathbb{N}$ and $L \subseteq \{0,1\}^*$. We say that $L \in \mathbf{RSPACE}^{\infty}(s(n))$ if there exists a NDTM M deciding L such that the computation of M on input x accesses at most $O(s(|x|))$ cells of the work tapes for all $x \in \{0,1\}^*$, and for any input x accepted by M, at least half of the computation paths accept. We say that*

[3]We leave to the reader to look up any definitions we have not given as they are of little relevance to this survey, so will not be offered.

$L \in \mathbf{RSPACE}(s(n))$ if M also runs in polynomial time, and we let $\mathbf{RL} = \mathbf{RSPACE}(\log(n))$.[4]

One of the foundational results in this area is that randomized logspace computation can decide exactly the same languages as non-deterministic logspace computation.

Theorem 43. *[34] $\mathbf{RSPACE}^{\infty}(\log(n)) = \mathbf{NL}$.*

It follows directly that STCONN is complete for $\mathbf{RSPACE}^{\infty}(\log(n))$, and thus STCONN $\in \mathbf{RL}$ if and only if $\mathbf{RL} = \mathbf{NL}$, but this remains an open question. An early result in the field, about three decades prior to the general result of Reingold (Theorem 14) was that the undirected version of reachability was in $\mathbf{RL}$.

Theorem 44. *[33]* USTCONN $\in \mathbf{RL}$.

More recently, building on the techniques of Reingold's proof of Theorem 14, it was conjectured in [8], based on strong evidence, that $\mathbf{L} = \mathbf{RL}$.

References

[1] S. Arora and B. Barak, *Computational Complexity: A Modern Approach*. Cambridge ; New York: Cambridge University Press, 1 edition ed., Apr. 2009.

[2] S. Aaronson, G. Kuperberg, C. Granade, and V. Russo, "Complexity Zoo."

[3] N. V. Vinodchandran, "Space Complexity of the Directed Reachability Problem over Surface-Embedded Graphs," in *Perspectives in Computational Complexity* (M. Agrawal and V. Arvind, eds.), pp. 37–49, Cham: Springer International Publishing, 2014.

[4] A. Wigderson, "The complexity of graph connectivity," in *Mathematical Foundations of Computer Science 1992* (G. Goos, J. Hartmanis, I. M. Havel, and V. Koubek, eds.), vol. 629, pp. 112–132, Berlin, Heidelberg: Springer Berlin Heidelberg, 1992.

[5] W. J. Savitch, "Relationships between nondeterministic and deterministic tape complexities," *Journal of Computer and System Sciences*, vol. 4, pp. 177–192, Apr. 1970.

[6] N. Immerman, "Nondeterministic Space is Closed under Complementation," *SIAM J. Comput.*, vol. 17, pp. 935–938, Oct. 1988.

[7] O. Reingold, "Undirected Connectivity in Log-space," *J. ACM*, vol. 55, pp. 17:1–17:24, Sept. 2008.

[8] O. Reingold, L. Trevisan, and S. Vadhan, "Pseudorandom walks on regular digraphs and the RL vs. L problem," in *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing - STOC '06*, (Seattle, WA, USA), p. 457, ACM Press, 2006.

[4]We define the unconventional parameterized class $\mathbf{RSPACE}^{\infty}$ in order to handle historical changes to the definition of the class $\mathbf{RSPACE}$ which was defined in [33] without any limits on the runtime. However, the results of [34] give reason to include a polynomial time bound in modern definitions to avoid having two names for the class $\mathbf{NL}$. We define this class solely to provide convenient notation for stating these historical results.

[9] V. Ramachandran and J. Reif, "Planarity testing in parallel," *Journal of Computer and System Sciences*, vol. 49, pp. 517–561, Dec. 1994.

[10] E. Allender and M. Mahajan, "The Complexity of Planarity Testing," in *STACS 2000* (H. Reichel and S. Tison, eds.), Lecture Notes in Computer Science, pp. 87–98, Springer Berlin Heidelberg, 2000.

[11] M. Saks, "Randomization and derandomization in space-bounded computation," in *Proceedings of Computational Complexity (Formerly Structure in Complexity Theory)*, pp. 128–149, May 1996.

[12] M. Blum and D. Kozen, "On the power of the compass (or, why mazes are easier to search than graphs)," *IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 132–142, 1978.

[13] E. Allender, S. Datta, and S. Roy, "The Directed Planar Reachability Problem," in *FSTTCS 2005: Foundations of Software Technology and Theoretical Computer Science* (S. Sarukkai and S. Sen, eds.), Lecture Notes in Computer Science, pp. 238–249, Springer Berlin Heidelberg, 2005.

[14] E. Allender, D. A. M. Barrington, T. Chakraborty, S. Datta, and S. Roy, "Grid Graph Reachability Problems," *21st Annual IEEE Conference on Computational Complexity (CCC&#39;06)*.

[15] K. Reinhardt and E. Allender, "Making Nondeterminism Unambiguous," *SIAM J. Comput.*, vol. 29, pp. 1118–1131, Jan. 2000.

[16] R. Tewari, *On the Space Complexity of Directed Graph Reachability*. 2007.

[17] G. Barnes, J. F. Buss, W. L. Ruzzo, and B. Schieber, "A SUBLINEAR SPACE, POLYNOMIAL TIME ALGORITHM FOR DIRECTED s-t CONNECTIVITY," p. 10.

[18] T. Imai, K. Nakagawa, A. Pavan, N. Vinodchandran, and O. Watanabe, "An O(n+?)-Space and Polynomial-Time Algorithm for Directed Planar Reachability," pp. 277–286, June 2013.

[19] T. Asano, D. Kirkpatrick, K. Nakagawa, and O. Watanabe, "\widetilde{O}(\sqrt{n})$ -Space and Polynomial-Time Algorithm for Planar Directed Graph Reachability," vol. 8635, pp. 45–56, Aug. 2014.

[20] D. Chakraborty, A. Pavan, R. Tewari, N. Vinodchandran, and L. Yang, "New time-space upperbounds for directed reachability in high-genus and H-minor-free graphs," vol. 29, pp. 585–595, Dec. 2014.

[21] T. Thierauf and F. Wagner, "Reachability in K3,3-Free Graphs and K5-Free Graphs Is in Unambiguous Log-Space," in *Fundamentals of Computation Theory* (M. Kutyowski, W. Charatonik, and M. Gbala, eds.), Lecture Notes in Computer Science, (Berlin, Heidelberg), pp. 323–334, Springer, 2009.

[22] C. Thomassen, "The graph genus problem is NP-complete," *Journal of Algorithms*, vol. 10, pp. 568–576, Dec. 1989.

[23] T. Asano and B. Doerr, "Memory-Constrained Algorithms for Shortest Path Problem," in *CCCG*, 2011.

[24] R. Ashida and K. Nakagawa, "O (nˆ1/3)-Space Algorithm for the Grid Graph Reachability Problem," in *34th International Symposium on Computational Geometry (SoCG 2018)* (B. Speckmann and C. D. Tth, eds.), vol. 99 of *Leibniz International Proceedings in Informatics (LIPIcs)*, (Dagstuhl, Germany), pp. 5:1–5:13, Schloss DagstuhlLeibniz-Zentrum fuer Informatik, 2018.

[25] R. Jain and R. Tewari, "Grid Graph Reachability," *arXiv:1902.00488 [cs]*, Feb. 2019. arXiv: 1902.00488.

[26] S. A. Cook and C. W. Rackoff, "Space Lower Bounds for Maze Threadability on Restricted Machines," *SIAM J. Comput.*, vol. 9, pp. 636–652, Aug. 1980.

[27] *Foundations of Computer Science, 34th Symposium on (FOCS '93).* Los Alamitos: IEEE Computer Society Press, 1993. OCLC: 812644562.

[28] J. Edmonds, C. Poon, and D. Achlioptas, "Tight Lower Bounds for st-Connectivity on the NNJAG Model," *SIAM J. Comput.*, vol. 28, pp. 2257–2284, Jan. 1999.

[29] S. R. Mahaney, "Sparse complete sets for NP: Solution of a conjecture of Berman and Hartmanis," *Journal of Computer and System Sciences*, vol. 25, pp. 130–143, Oct. 1982.

[30] J. Hartmanis, "On log-tape isomorphisms of complete sets," *Theoretical Computer Science*, vol. 7, pp. 273–286, Dec. 1978.

[31] Jin-Yi Cai and D. Sivakumar, "The resolution of a Hartmanis conjecture," in *Proceedings of IEEE 36th Annual Foundations of Computer Science*, pp. 362–371, Oct. 1995.

[32] Jin-Yi Cai and D. Sivakumar, "Resolution of hartmanis' conjecture for nl-hard sparse sets,"

[33] R. Aleliunas, R. M. Karp, R. J. Lipton, L. Lovasz, and C. Rackoff, "Random walks, universal traversal sequences, and the complexity of maze problems," in *20th Annual Symposium on Foundations of Computer Science (sfcs 1979)*, pp. 218–223, Oct. 1979.

[34] J. Gill, "Computational Complexity of Probabilistic Turing Machines," *SIAM J. Comput.*, vol. 6, pp. 675–695, Dec. 1977.