Parameterized Graph Connectivity and Polynomial-Time Sub-Linear-Space Short Reductions
(Preliminary Report)

Tomoyuki Yamakami(✉)

Faculty of Engineering, University of Fukui, 3-9-1 Bunkyo, Fukui 910-8507, Japan
TomoyukiYamakami@gmail.com

Abstract. We are focused on the solvability/insolvability of the directed *s-t* connectivity problem (DSTCON) parameterized by suitable size parameters $m(x)$ on multi-tape deterministic Turing machines working on instances x to DSTCON by consuming simultaneously polynomial time and sub-linear space, where the informal term "sub-linear" refers to a function of the form $m(x)^{\varepsilon}\ell(|x|)$ on instances x for a certain absolute constant $\varepsilon \in (0,1)$ and a certain polylogarithmic function $\ell(n)$. As natural size parameters, we take the numbers $m_{ver}(x)$ of vertices and of edges $m_{edg}(x)$ of a graph cited in x. Parameterized problems solvable simultaneously in polynomial time using sub-linear space form a complexity class PsubLIN and it is unknown whether DSTCON parameterized by m_{ver} belongs to PsubLIN. Toward this open question, we wish to investigate the relative complexity of DSTCON and its natural variants and classify them according to a restricted form of many-one and Turing reductions, known as "short reductions," which preserve the polynomial-time sub-linear-space complexity. As variants of DSTCON, we consider the breadth-first search problem, the minimal path problem, and the topological sorting problem. Certain restricted forms of them fall into PsubLIN. We also consider a stronger version of "sub-linear," called "hypo-linear." Additionally, we refer to a relationship to a practical working hypothesis known as the linear space hypothesis.

Keywords: Sub-linear space · Hypo-linear space · Directed s-t-connectivity · NL search · NL optimization · Short reduction · Linear space hypothesis

1 Background and Overview

1.1 Solvability of the Directed *s-t* Connectivity Problem

Polynomial-time computation has been widely acknowledged as a natural, reasonable, theoretical model of tractable computation and all such tractable

This work was done at the University of Toronto between August 2016 and March 2017 and was supported by the Natural Sciences and Engineering Council of Canada.

M. Hague and I. Potapov (Eds.): RP 2017, LNCS 10506, pp. 176–191, 2017.
DOI: 10.1007/978-3-319-67089-8_13

decision problems are known to form a complexity class, known as P. For polynomial-time computation, we are more keen to its minimal use of memory space from theoretical and practical interest. A typical example of small memory usage may be *logarithmic-space computation* (or *log-space computation*, in short), which requires $O(\log n)$ memory space to complete its computation on each input of size n. The complexity class L is composed of all decision problems solvable in polynomial time using log space and its nondeterministic counterpart is known as NL. Unlike NP (nondeterministic polynomial time), NL is known to be closed under complementation [8,14]. Besides the well-studied P = NP question, one of the most challenging open questions is the decades-old L = NL question, which asks whether NL decision problems are all solvable using log space.

One of the most studied NL problems is probably the *directed s-t connectivity problem*[1] (DSTCON) concerning the reachability of vertices in a directed graph.

Directed *s-t* Connectivity Problem (DSTCON):

- Instance: a directed graph G and two designated vertices s and t.
- Question: is there any path from s to t in G?

Decades ago, Jones [9] demonstrated that DSTCON is NL-complete (under log-space many-one reductions). Even though instances are restricted to directed acyclic graphs of maximum degree at most 3 (3DSTCON), the *s-t* connectivity problem still remains NL-complete. Nonetheless, DSTCON has since then played a key role as a typical NL-complete problem in quest of determining the exact computational complexity of NL.

In order to solve DSTCON, a straightforward exhaustive search algorithm requires simultaneously $O(m + n)$ time and $O(n \log n)$ space for any directed graph of n vertices and m edges. A more sophisticated deterministic algorithm of Barnes et al. [4] solves it in polynomial time using at most $n/2^{c\sqrt{\log n}}$ space for a certain constant $c > 0$. This space bound is only slightly below a linear function. For restricted graphs such as planar directed graphs, on the contrary, Asano et al. [3] gave a polynomial-time, $\sqrt{n}\,\ell(n)$-space algorithm for DSTCON, where ℓ refers to a certain suitable polylogarithmic function. Moreover, Kannan et al. [10] designed an $n^{\varepsilon}\ell(n)$-space algorithm for so-called directed unique-path graphs, where ε is a constant in $(0, 1)$. In particular, when directed graphs are regular, DSTCON is solvable using $O(\log n)$ space [12]. For single-source planar directed acyclic graphs, there is an $O(\log n)$-space algorithm to determine their *s-t* connectivity [1]. The *s-t* connectivity problem for undirected graphs (USTCON) can be solved using only $O(\log n)$ space [11]. In contrast, if we allow super-polynomial (i.e., $\Theta(n^{\log n})$) execution time, then there is a known deterministic algorithm solving DSTCON using $O(\log^2 n)$ space [13].

Despite decades of vigorous studies, it is still unknown whether DSTCON or its search version can be solved in polynomial time using "significantly less" memory space than any linear function.

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

A search version of the DSTCON, denoted by Search-DSTCON, is to find a (not necessary simple) path in a given directed graph between two specified vertices if any (otherwise, it outputs a designated symbol "$\perp$"). Such a path can be easily found using log space with an adaptive queries to oracles in NL. Search-BFT, for example, is an NL search problem of constructing from a given directed graph a breadth-first tree rooted at a given starting vertex and a typical approach to solve DSTCON is to construct such breadth-first trees. In a similar fashion, many NL decision problems can be turned into *NL search problems* and *NL optimization problems* (or NLO problems, in short) [15,16].

1.2 Size Parameters and Parameterized Problems

We are more concerned with problems, which are parameterized by certain "size parameters," where size parameters in practice play important roles in measuring the precise computational complexity of the problems. All the space-complexity bounds stated in Sect. 1.1 concern with DSTCON parameterized by two particular size parameters, one of which is the total number $m_{ver}(x)$ of vertices and the other is the number $m_{edg}(x)$ of edges of a graph in a graph-related instance x. Generally, the choice of different size parameters tends to lead to different complexities. To certain restricted variants of DSTCON and Search-DSTCON, however, m_{ver} and m_{edg} endow the same complexity.

1.3 Sub-Linear and Hypo-Linear Space

For a further discussion and exposition, it is imperative to clarify our terminology of the "sub-linearity." Even though slightly unusual but for our convenience, we informally use two different terms to describe a space bound below any linear function. The informal term "sub linear" indicates functions of the form $m(x)^{\varepsilon}\ell(|x|)$ on instances x for a *certain* constant $\varepsilon \in (0,1)$ and a suitable polylogarithmic function ℓ [17], whereas the term "hypo linear" (or possibly "far sub-linear") means functions upper-bounded by $m(x)^{\varepsilon}\ell(|x|)$ on instances x for an *arbitrary* choice of $\varepsilon \in (0,1)$. The multiplicative factor ℓ may become redundant when $m(x)$ is relatively large (e.g., $m(x) \geq (\log |x|)^c$ for any constant $c \geq 1$). Corresponding to the above terms, we express as PsubLIN the class of all (parameterized) decision/search problems solvable in polynomial time using sub-linear space [17]. Similarly, PhypoLIN is defined using hypo-linear space.

1.4 Short Reductions

The notion of reducibility has served as an effective tool in comparing the computational complexity of problems.

As noted earlier, it is unknown whether DSTCON is in PsubLIN. To solve this open question, it is imperative to understand the structure of the class PsubLIN (as well as PhypoLIN). For our purpose, we will investigate the relative complexity of DSTCON and its natural variants based on appropriately chosen

"reducibilities," in particular, space-bounded many-one and Turing reducibilities, which are technical tools in measuring the relative complexity of two given problems. Since we are concerned with sub-linear and hypo-linear space computations, for our study, we particularly need a quite restricted form of reducibility, which are known as "short" reducibility [17].

Polynomial-time sub-linear-space computability makes quite different effects on various NL-complete problems. Notably, even if DSTCON is polynomial-time, sub-linear-space solvable, many other NL-complete problems may not be solved in polynomial time using sub-linear space. Let us recall 3DSTCON. The problem BDSTCON is to ask whether an s-t path of length at most k exists in a directed graph. These problems 3DSTCON and BDSTCON are NL-complete but they are not known to have the same complexity as DSTCON does from a viewpoint of polynomial-time sub-linear-space computability.

This circumstance no longer makes a standard log-space reduction suitable for our study on PsubLIN. For this very reason, we intend to use *short reducibility*, in which an outcome of a reduction must be *linear* in the size parameter of each instance. This linear-size requirement is sufficient to guarantee the closure property of PsubLIN under those short reductions.

Here, we consider three types of short reducibilities $\leq_m^{\mathrm{sL}}$, $\leq_T^{\mathrm{sL}}$, and $\leq_T^{\mathrm{sSLRF}}$, whose precise definitions will be given in Sect. 3.2. These short reducibilities help us classify various NL decision, search, and optimization problems from the viewpoint of polynomial-time sub-linear-space computability.

1.5 Major Contributions

A main motivation of this work is to determine the polynomial-time sub-linear-space solvability/insolvability of DSTCON. We hope that this work will pave a way to determining the minimum space usage necessary for all NL-complete problems, which may lead us to an answer to the NL $\subseteq$?PsubLIN problem or even the decades-old L =?NL problem. In this preliminary report, we will provide a number of results for DSTCON and other graph-related problems, aiming at the better understandings of the *relative complexity* of DSTCON and its variants parameterized by size parameters. The use of short reductions help us obtain a classification of the computational complexity of those parameterized problems. This classification result is summarized in Fig. 1, in which a lower problem is $\leq_T^{\mathrm{sSLRF}}$-reducible to an upper problem, both of which are parameterized by m_{ver}. In what follows, we will give a brief explanation of the problems cited in the figure.

One of the important properties of directed graphs is *acyclicity*, where a graph is *acyclic* if there is no cycle (including self-loops) in it. We write ADSTCON for the *acyclic directed s-t connectivity problem*, in which we determine the existence of an s-t path in each given directed acyclic graph. Technically speaking, this problem is a so-called "promise problem" because the acyclic property of a graph G is guaranteed *a priori* when instances (G, s, t) of ADSTCON are given. In contrast, the problem DCYCLE asks whether there is a cycle in a given directed graph.

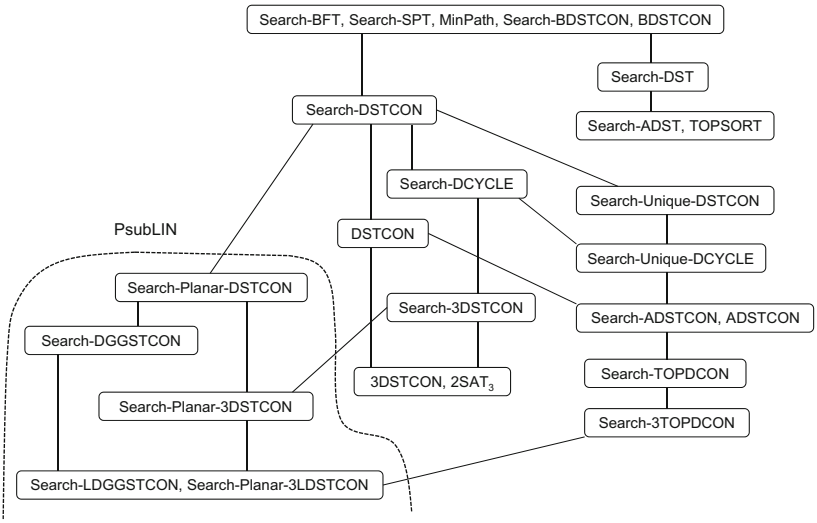

Fig. 1. $\leq_T^{\mathrm{sSLRF}}$-reducibility relationships among decision/search problems parameterized by m_{ver}. Problems below the dotted curved line are shown to be in PsubLIN.

It is useful to deal with an instance graph, which has *at most one s-t path*. This restriction gives rise to another problem, called the *unique directed s-t connectivity problem* (UniqueDSTCON). Similarly, we consider the *unique directed cycle problem* (UniqueDCYCLE) by requiring "at most one cycle."

Planarity is another important property, where a graph is *planar* if it can be drawn on a plane in a way that no two edges intersect with each other except for their endpoints. Such a drawing is called a *planar combinatorial embedding*, which is a permutation of the edges adjacent to each vertex. We write PlanarDSTCON for DSTCON whose inputs are planar graphs. Similarly, we define Planar3DSTCON from 3DSTCON.

A directed graph $G = (V, E)$ is said to be *layered* if the vertex set V is partitioned into $L_1, L_2, \ldots, L_k$ with $k \in \mathbb{N}^+$ such that, for every $i \in [k-1]$, all edges from L_i are directed to certain vertices in L_{i+1}. We write 3LDSTCON for 3DSTCON limited to inputs of layered graphs.

As a special case of planar graphs, a *grid graph* is a graph $G = (V, E)$ in which $V \subseteq \mathbb{N} \times \mathbb{N}$ and all edges are of the form either $((i,j),(i+b,j))$ or $((i,j),(i,j+b))$ for a certain $b \in \{\pm 1\}$. We consider the *directed grid graph s-t connectivity problem* (DGGSTCON). A directed grid graph is called *layered* if it contains only edges directed to east and south (i.e., rightward and downward edges). The layered version of DGGSTCON is denoted by LDGGSTCON.

All the aforementioned problems are decision problems. To refer to their associated search version, we use a simple notation of Search-P for each decision problem P. For example, a search version of DSTCON is Search-DSTCON, which is to find an s-t path in a given directed graph G.

The *minimum path problem*, Min-Path, is an optimization problem of finding a minimal s-t path in a directed graph G from each instance of the form (G, s, t).

The detailed explanations of the following graph-related concepts will be given in Sect. 4.2. The notation Search-DST denotes the problem of finding a *(directed) spanning tree* of any given directed graph G, rooted at a specified vertex r in G. When instance graphs to Search-DST are promised to be directed acyclic graphs, we write Search-ADST instead of Search-DST. We denote by Search-SPT the problem of finding from each instance (G, r) a *shortest-path tree* rooted at r. Search-BFT is the problem of finding a *breadth-first tree* of a given directed graph rooted at a given vertex (with the left vertex condition).

In many application of DSTCON, certain features of instance graphs are often used to simplify NL-completeness proofs. One such feature is *topological ordering* of vertices of a given graph. Here, a topologically-ordered version of DSTCON (resp., 3DSTCON) is denoted by TOPDCON (resp., 3TOPDCON). In contrast, TOPSORT is a search problem whose task is to produce a topological ordering of a given directed acyclic graph starting at a given vertex.

A sophisticated algorithm of Barnes et al. [4] together with the equivalence in complexity among parameterized problems $(\text{Search-BFT}, m_{ver})$, $(\text{Search-SPT}, m_{ver})$, and $(\text{Min-Path}, m_{ver})$ (see Proposition 9(2)) leads to the following solvability result of them.

Theorem 1. $(\text{Search-BFT}, m_{ver})$, $(\text{Search-SPT}, m_{ver})$, *and* $(\text{Min-Path}, m_{ver})$ *are solved in polynomial time using at most* $n/2^{c\sqrt{\log n}}$ *space for an absolute constant* $c > 0$.

Lately, the *linear space hypothesis* or *LSH*—a practical working hypothesis—was introduced in [17] in connection to polynomial-time sub-linear-space computability. The hypothesis LSH for 2SAT_3 asserts that no polynomial-time sub-linear-space deterministic algorithm solves 2SAT_3, which is the satisfiability problem restricted to 2CNF formulas, each variable of which appears at most 3 times as literals. It was shown in [17] that LSH for 2SAT_3 implies that 3DSTCON does not fall into PsubLIN. What follows in the next theorem is a simple application of LSH for 2SAT_3 to the computational complexity of $(\text{TOPSORT}, m_{ver})$. An argument similar to [17, Sect. 6] proves the insolvability of $(\text{TOPSORT}, m_{ver})$.

Theorem 2. *Assuming LSH for* 2SAT_3, *no deterministic Turing machine solves* $(\text{TOPSORT}, m_{ver})$ *in polynomial time using* $O(m_{ver}(x)^{\varepsilon/2})$ *space on instances* x *to* TOPSORT *for any fixed constant* $\varepsilon \in [0, 1)$.

2 Basic Notions and Notation

2.1 Numbers and Graphs

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 two integers m and n with $m \leq n$, an *integer interval* $[m,n]_{\mathbb{Z}}$ is the set $\{m, m+1, m+2, \ldots, n\}$. When $n \geq 1$, we conventionally write $[n]$ in place of $[1,n]_{\mathbb{Z}}$.

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

A *directed graph* G is expressed as (V,E) with a set V of vertices and a set E of edges. We explicitly express edges as pairs of vertices and this convention eliminates multi-edges. In a given graph G, a *path* from s to t is a series $(x_1, x_2, \ldots, x_n)$ of vertices in G with $n \geq 2$ such that $x_1 = s$, $x_n = t$, and (x_i, x_{i+1}) is an edge in G for every index $i \in [n-1]$. A path is called *simple* if no internal vertex is repeated (i.e., there is no cycle or self-loop in it). An *s-t path* is a path from vertex s to vertex t and is expressed as $s \rightsquigarrow t$. The *length* of a path from v to u is the number of edges in the path. and the notation $dis(u,v)$ denotes the *distance* from u to v, which is the minimal length of any path from u to v. A directed graph G is *(weakly) connected* if its underlying undirected graph is connected. For each vertex $v \in V$, let $in(v) = \{u \in V \mid (u,v) \in E\}$ and $out(v) = \{u \in V \mid (v,u) \in E\}$. Each vertex v has *indegree* $|in(v)|$ and *outdegree* $|out(v)|$. We simply call $|in(v) \cup out(v)|$ the *degree* of vertex v. A *source* is a vertex of indegree 0 and a *sink* is that of outdegree 0.

Let $G = (V,E)$ be any directed graph. Given a subset S of V, we write $G \setminus S$ for the graph obtained from G by removing all vertices in S and all edges incident on them. A subset S of V in G is called a *planarizing set* if $G \setminus S$ is a planar graph. A subset S of V is said to be *separating* if $G \setminus S$ is disconnected, and *nonseparating* otherwise.

2.2 Machine Models, Parameterized Problems, and Size Parameters

Our basic model of computation is a multi-tape Turing machine of the following form. Our *Turing machine* consists of a read-only input tape, (possibly) a write-only output tape, and a constant number of read/write work tapes. A tape head on the output tape moves only to the right if it writes a non-blank symbol, and it stays still otherwise. All other tape heads move in both directions (to the right and to the left) unless it stays still. An *oracle Turing machine* is further equipped with a *query tape*—a special output tape—on which the machine produces query strings (or query words) to transmit to the oracle for its answer. Given an oracle P, the notation $M^P(x)$ indicates an outcome of M on input x by making queries to the oracle P.

We will study the computational complexity of problems based on a suitable choice of "size parameters" in place of a standard size parameter, which is the total length $|x|$ of the binary representation of an input instance x. To emphasize the choice of m, we often write (P,m) in place of P (when we use the standard "length" of instances, we omit "m" and write P instead of (P,m)). A *(log-space) size parameter* $m(x)$ for a problem P is formally a function mapping Σ^* to $\mathbb{N}$

such that (i) $m(x)$ must be computed using $O(\log |x|)$ space and (2) there exists a polynomial p satisfying $m(x) \leq p(|x|)$ for all instances x of P.

For any graph-related problem P, let $m_{edg}(x)$ and $m_{ver}(x)$ denote respectively the total number of edges and that of vertices of a graph given as a part of instance x of P. We say that a 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, on each input x, each of the work tapes (not including input and output tapes) used by M on x are upper-bounded by $c \log m(x) + d$.

Associated with log-space computability, L and NL are respectively the classes of all decision problems solvable on deterministic and nondeterministic Turing machines using log space. It is known that the additional requirement of "polynomial runtime" does not change these classes. The notation FL stands for a class of polynomially-bounded functions that can be computed using space at most $O(\log |x|)$.

An *NL search problem* P parameterized by a (log-space) size parameter $m(x)$ is a pair (P, m) with $P = (I, SOL)$ satisfying $I \in$ L and $I \circ SOL \in$ auxFL, where I consists of (admissible) instances and SOL is a function from I to a set of strings such that, for any (x, y), $y \in SOL(x)$ implies $|y| \leq m(x)$, where $I \circ SOL$ for the set $\{(x, y) \mid x \in I, y \in SOL(x)\}$ [15,16]. In addition, for each fixed constant $k > 0$, the notation $(I \circ SOL)^{\exists}_{m}$ denotes the set $\{x \in I \mid \exists y \in SOL(x)\ [|y| \leq km(x)+k \wedge (x, y) \in I \circ SOL\ ]\}$. We say that a deterministic Turing machine M *solves* (P, m) with $P = (I, SOL)$ 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, $\perp$, otherwise. We denote by Search-NL a collection of all NL search problems. For convenience, its polynomial-time counterpart is denoted by Search-NP. It follows that Search-NL $\subseteq$ Search-NP. Search-PsubLIN denotes a search version of PsubLIN.

3 Sub-Linear/Hypo-Linear Space and Short Reductions

3.1 Sub-Linear and Hypo-Linear Space Computation

Our target is search and optimization problems parameterized by suitable size parameters $m(x)$ for instances x. Throughout this paper, we informally use the term "sub-linear" to mean a function of the form $m(x)^{\varepsilon}\, \ell(|x|)$ on input instances x for a certain constant $\varepsilon \in (0, 1)$ and a certain polylog function $\ell(n)$. In contrast, "hypo linear" (or possibly "far sub linear") refers to functions upper-bounded by the function $m(x)^{\varepsilon}\, \ell(|x|)$ on instances x for an arbitrary choice of constant $\varepsilon \in (0, 1)$ and for a certain polylog function $\ell(n)$.

A (parameterized) search problem (P, m) with $P = (I, SOL)$ is said to be *solvable in polynomial time using sub-linear space* (resp., *using hypo-linear space*) if, for a certain choice of constant $\varepsilon \in (0, 1)$ (resp., for an arbitrary choice of $\varepsilon \in (0, 1)$, there exist a deterministic Turing machine M_{ε}, a polynomial p_{ε}, and a polylog function ℓ_{ε} for which M finds a valid solution in $SOL(x)$ in at most $p_{\varepsilon}(|x|)$ steps using at most $m(x)^{\varepsilon}\, \ell_{\varepsilon}(|x|)$ tape cells for all admissible instances x in I. We use PsubLIN (resp., PhypoLIN) to denote the collection of all decision

problems (P, m) that are solvable in polynomial time using sub-linear space (resp., hypo-linear space), where the suffix "P" refers to "polynomial time."

Moreover, we introduce the notation PTIME,SPACE($s(n)$) to denote 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 to P. It thus follows that $\mathrm{L} \subseteq \mathrm{PhypoLIN} \subseteq \mathrm{PsubLIN} \subseteq \mathrm{P}$. Note that $\mathrm{L} = \mathrm{P}$ implies $\mathrm{L} = \mathrm{PhypoLIN} = \mathrm{PsubLIN} = \mathrm{P}$.

3.2 Short Reductions Among Decision and Search Problems

Let us first extend the existing notion of short reducibilities, which was first discussed in [17] for (parameterized) decision problems, to search problems.

We start with standard L-m-reducibility. For any two (parameterized) search problems (P_1, m_1) and (P_2, m_2) with $P_1 = (I_1, SOL_1)$ and $P_2 = (I_2, SOL_2)$, 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 are two functions $(f, \|), (g, \|) \in \mathrm{FL}$ (where $\|$ refers to the bit length) and two constants $k_1, k_2 > 0$ such that, for any x and y, (i) $x \in I_1$ implies $f(x) \in I_2$, (ii) $x \in I_1$ and $y \in SOL_2(f(x))$ imply $g(x, y) \in SOL_1(x)$, and (iii) $m_2(f(x)) \leq m_1(x)^{k_1} + k_1$, and (iv) $m_1(g(x, y)) \leq m_2(y)^{k_2} + k_2$. As for decision problems, we simply drop Condition (iv). Notice that all functions in FL are, by their definition, polynomially bounded.

To discuss the sub-linear-space solvability, however, we need to restrict the L-m-reducibility, which we call the *short L-m-reducibility* (or sL-m-reducibility, in short), obtained by replacing two equalities $m_2(f(x)) \leq m_1(x)^{k_1} + k_1$ and $m_1(g(x, y)) \leq m_2(y)^{k_2} + k_2$ in the above definition of $\leq_m^{\mathrm{L}}$ with $m_2(f(x)) \leq k_1 m_1(x) + k_1$ and $m_1(g(x, y)) \leq k_2 m_2(y) + k_2$, respectively. To express this new reducibility, we use another notation of $\leq_m^{\mathrm{sL}}$. Obviously, every $\leq_m^{\mathrm{sL}}$-reduction is an $\leq_m^{\mathrm{L}}$-reduction but the converse does not hold in general [17].

We say that (P_1, m_1) is *SLRF-T-reducible to* (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, a polynomial p_ε, a polylog function ℓ_ε, and three constants $k_1, k_2, k_3 \geq 1$ such that, (1) $M_\varepsilon^{P_2}(x)$ runs in at most $p_\varepsilon(|x|)$ time using at most $m_1(x)^\varepsilon \ell_\varepsilon(|x|)$ space for all instances x of P_1, provided that its query tape is not subject to a space bound, (2) when $M_\varepsilon^{P_2}(x)$ queries to P_2 with query word z written on a write-only query tape, $m_2(z) \leq m_1(x)^{k_1} + k_1$ and $|z| \leq |x|^{k_3} + k_3$ hold for all instances x to P_1, (3) for any oracle answer y, $m_1(M_\varepsilon^{P_2}(x)) \leq m_2(y)^{k_2} + k_2$, and (4) in response to the same word queries at any moment, the oracle P_2 always returns the same answer, which is a valid solution to P_2. Any oracle answer must be written on a *read-once answer tape*, in which its tape head moves from the left to the right whenever it reads a non-blank symbol. After M_ε makes a query, in a single step, it erases its query tape, it returns its tape head back to the initial cell, and an oracle writes its answer directly onto the answer tape.

We also define *short SLRF-T-reducibility* (or sSLRF-T-reducibility) by replacing the above inequalities $m_2(z) \leq m_1(x)^{k_1} + k_1$ and $m_1(M_\varepsilon^{P_2}(x)) \leq m_2(y)^{k_2} + k_2$ with $m_2(z) \leq k_1 m_1(x) + k_1$ and $m_1(M_\varepsilon^{P_2}(x)) \leq k_2 m_2(y) + k_2$,

respectively. We tend to use the notation $\leq_T^{\mathrm{sSLRF}}$ to denote this new reducibility. Every $\leq_T^{\mathrm{sSLRF}}$-reduction is obviously an $\leq_T^{\mathrm{SLRF}}$-reduction. In the case where M_ε is limited to log-space usage, we use a new notation $\leq_T^{\mathrm{sL}}$. 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 [17].

We list several fundamental properties of short reductions [17].

Lemma 3. [17]

1. $(P_1, m_1) \leq_m^{\mathrm{L}} (P_2, m_2)$ *implies* $(P_1, m_1) \leq_T^{\mathrm{L}} (P_2, m_2)$*, which further implies that* $(P_1, m_1) \leq_T^{\mathrm{SLRF}} (P_2, m_2)$*. The same statement holds also for* $\leq_m^{\mathrm{sL}}$*,* $\leq_T^{\mathrm{sL}}$*, and* $\leq_T^{\mathrm{sSLRF}}$*.*
2. *The reducibilities* $\leq_T^{\mathrm{SLRF}}$ *and* $\leq_T^{\mathrm{sSLRF}}$ *are reflexive and transitive.*
3. PhypoLIN *is closed under* $\leq_T^{\mathrm{SLRF}}$*-reductions and* PsubLIN *is closed under* $\leq_T^{\mathrm{sSLRF}}$*-reductions.*

Given any reduction, say, $\leq_r$, we say that A is $\leq_r$*-equivalent to* B, denoted by $A \equiv_r B$, if both $A \leq_r B$ and $B \leq_r A$ hold.

4 Relative Complexity of Search-DSTCON and Variants

The short reductions given in Sect. 3.2 are quite useful in determining the relative complexity of various decision, search, and optimization problems in connection to the polynomial-time sub-linear-space solvability. Figure 1 has shown numerous reducibility relationships among those problems in terms of $\leq_T^{\mathrm{sSLRF}}$-reducibility. In what follows, we will verify each of the relationships in the figure.

4.1 Connectivity of Acyclic, Planar, and Grid Graphs

We begin with DSTCON, ADSTCON, and DCYCLE and their search versions. It is important to note that a decision problem and its search version are not necessarily equivalent in complexity. For acyclic graphs, however, their associated decision and search problems are actually $\leq_T^{\mathrm{sL}}$-equivalent; namely, $(\text{Search-ADSTCON}, m) \equiv_T^{\mathrm{sL}} (\text{ADSTCON}, m)$ for any $m \in \{m_{ver}, m_{edg}\}$. This property is not yet observed for general graphs. For instance, we do not know whether $(\text{DSTCON}, m) \equiv_m^{\mathrm{sL}} (\text{Search-DSTCON}, m)$.

The uniqueness condition makes $(\text{Search-UniqueDSTCON}, m)$ and $(\text{Search-UniqueDCYCLE}, m)$ $\leq_m^{\mathrm{sL}}$-reducible to $(\text{Search-DSTCON}, m)$ and $(\text{Search-DCYCLE}, m)$, respectively, for each size parameter $m \in \{m_{ver}, m_{edg}\}$. In addition, the following relationships hold.

Lemma 4. *Let* $m \in \{m_{ver}, m_{edg}\}$.

1. $(\text{Search-DCYCLE}, m) \leq_T^{\mathrm{sL}} (\text{Search-DSTCON}, m)$.
2. $(\text{Search-UniqueDCYCLE}, m) \leq_T^{\mathrm{sL}} (\text{Search-UniqueDSTCON}, m)$.
3. $(\text{Search-ADSTCON}, m) \leq_T^{\mathrm{sL}} (\text{Search-UniqueDCYCLE}, m)$.

Proof. (1) The desired $\leq_T^{\mathrm{sL}}$-reduction works as follows. Let $G = (V, E)$ be given as an instance to Search-DCYCLE. Recursively, we choose one edge (s, t) in G. We then define another graph $G' = (V', E')$ with $V' = V$ and $E' = E - \{(s, t)\}$ and make a query of the form (G', s, t) to Search-DSTCON, used as an oracle. If the s-t path exists in G', then G must have a cycle passing through s and t. Otherwise, we choose another edge in the above recursion.

(2) This is essentially the same as (1).

(3) Given $x = (G, s, t)$ with $G = (V, E)$ as an instance to Search-ADSTCON, we define another graph $G' = (V', E')$ by setting $V' = V$ and $E' = E \cup \{(t, s)\}$. Since G is guaranteed to be acyclic, G' has a cycle (passing through s and t) iff there is an s-t path in G. Note that such a cycle is unique if it actually exists. Since $|V'| = |V|$ and $|E'| = |E| + 2$, it follows that (Search-ADSTCON, m) $\leq_T^{\mathrm{sL}}$ (Search-UniqueDCYCLE, m). □

Concerning the planarity property, the problem PlanarDSTCON belongs to UL [5], where UL is a natural variant of NL. Note that testing whether a given graph G is planar can be done in log space because such testing is reducible to USTCON [2], where USTCON is known to belong to L [11]. Moreover, a planar combinatorial embedding can be computed in log space [2].

At this point, we need to discuss the difference between two size parameters m_{ver} and m_{edg}. In certain restricted cases, the choice of those size parameters does not affect the computational complexity of graph-related problems.

Proposition 5. *1.* (Search-DSTCON, m_{edg}) $\leq_m^{\mathrm{sL}}$ (Search-DSTCON, m_{ver}).
2. (Search-3DSTCON, m_{ver}) $\equiv_m^{\mathrm{sL}}$ (Search-3DSTCON, m_{edg}).
3. (Search-PlanarDSTCON, m_{ver}) $\equiv_m^{\mathrm{sL}}$ (Search-PlanarDSTCON, m_{edg}).

It is not known at present that the opposite direction of Proposition 5(1) holds; namely, (Search-DSTCON, m_{ver}) $\leq_m^{\mathrm{sL}}$ (Search-DSTCON, m_{edg}).

Proof of Proposition 5. (1) Consider the following reduction function f. Given a graph $G = (V, E)$, if either s or t is an isolated vertex, then f immediately outputs a graph consisting of $\{s, t\}$ with no edges. Assuming otherwise, f transforms G into another graph $G' = (V', E)$ by removing all isolated vertices from G. Since $|E| \geq \frac{1}{2}|V'|$, it follows that $m_{ver}(G', s, t) \leq 2m_{edg}(G, s, t)$.

(2) An argument similar to (1) works for Search-3DSTCON, and we then obtain (Search-3DSTCON, m_{ver}) $\leq_m^{\mathrm{sL}}$ (Search-3DSTCON, m_{edg}). Conversely, let $x = (G, s, t)$ with $G = (V, E)$ be any instance to Search-3DSTCON. Assume, without loss of generality, that s is a source, t is a sink, and no isolated vertex exists in G. Since G has maximum indegree 2 and maximum outdegree 2, it follows that $|E| \leq 4|V|$. This implies that (Search-3DSTCON, m_{edg}) $\leq_m^{\mathrm{sL}}$ (Search-3DSTCON, m_{ver}).

(3) This comes from the fact that, for any planar graph $G = (V, E)$, if $|V| \geq 3$, then $|E| \leq 3|V| - 6$ holds. □

The problem LDGGSTCON is shown to be in UL ∩ co-UL [1] (stated as a comment after Theorem 20 in [1]). Note that Allender et al. [1] proved that (DGGSTCON, m_{ver}) $\equiv_m^{\mathrm{L}}$ (PlanarDSTCON, m_{ver}), but their L-m-reduction from PlanarDSTCON to DGGSTCON is *not* a short reduction.

Proposition 6. *Let* $m \in \{m_{ver}, m_{edg}\}$.

1. (Search-DGGSTCON, m) $\leq_m^{\mathrm{sL}}$ (Search-Planar3DSTCON, m).
2. (Search-LDGGSTCON, m) $\equiv_T^{\mathrm{sL}}$ (Search-Planar3LDSTCON, m).

A (k, ℓ)-*graph* is a graph in which every vertex has indegree at most k and outdegree at most ℓ. When instance graphs are limited to (k, ℓ)-graphs, we write, for example, (k, ℓ)DSTCON in place of DSTCON.

Proof Sketch. In this proof, we will show only (2). We first claim that (Search-Planar$(2,2)$LDSTCON, m) $\equiv_m^{\mathrm{sL}}$ (Search-Planar3LDSTCON, m). Next, we claim that (Search-Planar$(2,2)$LDSTCON, m) $\equiv_m^{\mathrm{sL}}$ (Search-LDGGSTCON, m). The reduction (Search-LDGGSTCON, m) $\leq_m^{\mathrm{sL}}$ (Search-Planar$(2,2)$ LDSTCON, m) is obvious. For the other direction, we use an argument of Allender et al. [1], who demonstrated that PlanarDSTCON is $\leq_m^{\mathrm{L}}$-reducible to DGGSTCON. Note that their L-m-reduction is not a short reduction, and thus we need a slight modification of their reduction using the fact that our instance graphs are layered.

One way to extend the notion of planarity is to consider embedding onto orientable surfaces of genus more than 1, where the *genus* of a closed orientable surface is roughly the number of "handles" added to a sphere. Given a constant $k \in \mathbb{N}^+$, EOS(k) denotes a set of all directed/undirected graphs whose underlying undirected graphs can be embedded on orientable surfaces of genus g. For a function $g : \mathbb{N} \to \mathbb{N}$, EOSDCON$(g)$ is DSTCON whose instance graphs $G = (V, E)$ are restricted to EOS$(g(|V|))$. In particular, EOSDCON(0) coincides with PlanarDSTCON.

Theorem 7. *Let* $m \in \{m_{ver}, m_{edg}\}$.

1. (Search-PlanarDSTCON, m) $\in$ PsubLIN.
2. *For any constant* $\varepsilon \in (0,1)$, (Search-EOSDCON(n^ε), m) $\in$ PsubLIN.
3. (Search-LDGGSTCON, m) $\in$ PhypoLIN.

Proof Sketch. (1)–(2) follow directly from [3,6].

(3) Let ε be any constant in $[0,1)$. It suffices to show that (Search-LDGGSTCON, m_{ver}) belongs to PTIME,SPACE$(n^\varepsilon \cdot polylog(n))$. Let $x = (G, s, t)$ be any instance to Search-LDGGSTCON. Let $G = (V, E)$ and assume that $V = [n] \times [n]$ for simplicity. Let $\varepsilon \in (0,1)$. For the value n^ε, we set $S = \{(a,b) \in [n] \times [n] \mid a + b = n^\varepsilon + 1\}$, $V_1 = \{(a,b) \in [n] \times [n] \mid a + b < n^\varepsilon + 1\}$, and $V_2 = \{(a,b) \in [n] \times [n] \mid a + b > n^\varepsilon + 1\}$. Note that $|S| \leq n^\varepsilon$ and $|V_1| \leq \sum_{i=1}^{n^\varepsilon - 1} i = \frac{(n^\varepsilon - 1)n^\varepsilon}{2}$. Pick $(a,b) \in S$. Consider a restricted grid graph with source (a,b) and sink (n,n). Let $V_{a,b} = \{(i,j) \mid a \leq i \leq n, b \leq j \leq n\}$. Note that $|V_{a,b}| = (n - a + 1)(n - b + 1)$. Recursively, we split this graph and then compute a path.

The optimization problem Min-Path is known to be complete for NLO$\cap$PBO (i.e., a class of nondeterministic log-space optimization problems whose solutions

are polynomially bounded) under approximation-preserving AC^0-reductions [15,16] but it is not known to be complete for NLO. In contrast, the problem Max-Path of finding the maximum simple paths of directed graphs is shown to be NPO-complete.

Given a function $f : \mathbb{N} \to \mathbb{N}$, we further define $(\text{Min-BPath}(f(n)), m)$ as a parameterized problem of finding a minimum s-t path of length at most $f(m(x))$ on every instance x to Min-Path. Here, we claim that $(\text{Min-Path}, m)$ can be reduced to its restricted form $(\text{Min-BPath}(n^\varepsilon), m)$ for each constant $\varepsilon \in (0,1)$.

Theorem 8. *For any* $m \in \{m_{ver}, m_{edg}\}$ *and* $\varepsilon \in (0,1)$, $(\text{BDSTCON}, m) \equiv_T^{\text{sL}}$ $(\text{Min-Path}, m) \equiv_T^{\text{sSLRF}} (\text{Min-BPath}(n^\varepsilon), m)$.

Proof. Let $m \in \{m_{ver}, m_{edg}\}$ and $\varepsilon \in (0,1)$. To show $(\text{Min-Path}, m) \leq_T^{\text{sL}}$ $(\text{BDSTCON}, m)$, we start with $x = (G, s, t)$, obtain the minimal length, say, ℓ of any s-t path by making queries to BDSTCON, and construct a path by setting $v_1 = s$ and by finding recursively v_{i+1} from $(v_1, v_2, \ldots, v_i)$ so that G has a v_i-t path of length $\ell - i + 1$. The converse $(\text{BDSTCON}, m) \leq_m^{\text{sL}} (\text{Min-Path}, m)$ is obvious.

It is easy to see that $(\text{Min-BPath}(n^\varepsilon), m) \leq_m^{\text{sL}} (\text{Min-Path}, m)$. For the converse, it suffices to show that $(\text{BDSTCON}, m) \leq_T^{\text{sSLRF}} (\text{Min-BPath}(n^\varepsilon), m)$. Let $x = (G, s, t, k)$ with $G = (V, E)$ be any instance to BDSTCON. We want to design an algorithm that determines whether the distance between s and t is at most k.

Let $|V| = n$ and $\tilde{n} = \lceil n^\varepsilon \rceil$. At stage $i \geq 1$, compute $D_0^{(i)} = \{u \in V \mid dis(s,u) = i\}$ by making queries of the form (G, s, u) to $\text{Min-BPath}(n^\varepsilon)$. If $|D_0^{(i)}| > n^{1-\varepsilon}$, then move to the next stage $i+1$. Assume otherwise. Starting with $j = 1$, recursively compute $D_j^{(i)} = \{u \in V \mid \exists w \in D_{j-1}^{(i)} [dis(w,u) = \tilde{n}]\}$. If $|D_j^{(i)}| > n^{1-\varepsilon}$, then move to stage $i+1$. Otherwise, increment j by one and continue until j reaches $\lfloor (k-i)/n^\varepsilon \rfloor + 1$. We then decide whether $dis(w,t) \leq \tilde{n}$ for a certain $w \in D_j^{(i)}$. This establishes the reducibility relation $(\text{BDSTCON}, m) \leq_T^{\text{sSLRF}} (\text{Min-BPath}(n^\varepsilon), m)$. □

4.2 Breadth-First Search and Topological Sorting

We have discussed in Sect. 1.5 a number of search and optimization problems without giving the meaning of technical terminology used to describe these problems. First of all, we will explain such technical terminology to clarify the definitions of Search-DST, Search-ADST, Search-SPT, Search-BFT, Search-TOPDCON, and TOPSORT, and we will verify the reducibility relationships, presented in Fig. 1, among these problems. See [7] for more information on the terminology.

Let us explain a general notion of spanning trees used for Search-DST. A *(directed) spanning tree* for a given directed graph G from vertex r is a directed tree T rooted at r for which T is a subgraph of G and all vertices reachable in G from r are also in T.

An *ordered tree* is a rooted tree in which the children of each internal vertex are linearly-ordered (in the left-to-right order among those children). Here, we assume a fixed linear ordering of vertices. For two vertices, u and v, u is *to the left of* v in G if either (i) u and v are children of a common parent and u is smaller than v or (ii) a certain ancestor of u is to the left of a certain ancestor of v. Similarly, we can define the notion of "to the right of".

In Search-SPT, a *shortest-path tree* from vertex r in a directed graph G is a directed tree T rooted at r that contains all vertices v reachable in G from r in such a way that any path in T from r to v must be a shortest path in G. All shortest-path trees are obviously spanning trees. Hence, it follows that $(\text{Search-DST}, m) \leq_m^{\text{sL}} (\text{Search-SPT}, m)$.

Breadth-first trees used in Search-BFT are a special case of ordered, shortest-path trees. A *breadth-first tree* of a directed graph G from a vertex r is an ordered, shortest-path tree T of G, rooted at r, satisfying the following *left vertex condition*: for any vertices u and v in G, (u, v) is incident to v in T iff u is to the left of every vertex w in G and (w, v) is incident to v. Search-BFT is closely related to an NL optimization problem of finding the shortest s-t paths in given directed graphs. Concerning the complexity of Search-BFT, we obtain $(\text{Search-DSTCON}, m) \leq_m^{\text{sL}} (\text{Search-BFT}, m)$ for each size parameter $m \in \{m_{ver}, m_{edg}\}$. Barnes et al. [4] demonstrated in essence that $(\text{Search-BFT}, m_{ver}) \in \text{PTIME,SPACE}(n/2^{\ell\sqrt{\log n}})$ for a certain constant $\ell > 0$.

Known polynomial-time algorithms solving DSTCON require more or less a (partial) construction of either breadth-first or depth-first trees from a given directed graph. Thus, we immediately obtain $(\text{Search-DSTCON}, m) \leq_m^{\text{sL}} (\text{Search-BFT}, m)$ for any size parameter $m \in \{m_{ver}, m_{edg}\}$. Therefore, it is important to investigate the space complexity of the breadth-first (and depth-first) tree search problems.

Theorem 9. *For any size parameter* $m \in \{m_{ver}, m_{edg}\}$, $(\text{Search-BFT}, m) \equiv_T^{\text{sL}} (\text{Search-SPT}, m) \equiv_T^{\text{sL}} (\text{Min-Path}, m)$.

Proof. Since $(\text{Search-SPT}, m) \leq_m^{\text{sL}} (\text{Search-BFT}, m)$, it suffices to show that (a) $(\text{Min-Path}, m) \leq_m^{\text{sL}} (\text{Search-SPT}, m)$ and (b) $(\text{Search-BFT}, m) \leq_T^{\text{sL}} (\text{Min-Path}, m)$.

(a) Given an instance $x = (G, s, t)$ to Min-Path, we make a query of the form (G, s) to Search-SPT, which returns a breadth-first tree rooted at s if it exists. We then output a unique s-t path in this tree. By the definition of breadth-first trees, this s-t path must be the shortest in G.

(b) Let $x = (G, r)$ with $G = (V, E)$ be any instance to Search-BFT. For each vertex $v \in V$, we make a query of the form (G, r, v) and calculate the minimum path length, $dis(r, v)$, between r and v from its oracle answer. Let L_i be a set of all vertices v satisfying $dis(r, v) = i$. Note that we can enumerate all elements in L_i according to a fixed linear ordering for G. We define a new graph $G' = (V', E')$ by setting $V' = V$ and $(u, v) \in E'$ whenever there exists an index $i \geq 1$ such that (i) $u \in L_i$, $v \in L_{i+1}$, and $(u, v) \in E$ and (ii) for any $w \in L_i$ with $w < u$, $(w, v) \notin E$. Clearly, G' is a breadth-first tree of G. □

A topological sort (topological order or topological numbering) used for the search problem TOPSORT serves as a key ingredient to the development of many elementary graph algorithms. Given a directed acyclic graph $G = (V, E)$, a *topological sort* of G from source s is a linear ordering of all its vertices starting at s satisfying that, for any edge $(u, v) \in E$, $u < v$ holds, where "$<$" is a logarithmic-space computable linear order. We express such a linear ordering as $(v_1, v_2, \ldots, v_n)$, where $n = |V|$. In Search-TOPDCON, however, we are given a directed graph $G = (V, E)$ such that (1) all vertices of G are numbered between 0 and n and (2) for any pair $i, j \in [n]$, if $(i, j) \in E$, then $i < j$ holds. The task of this problem is to find a path in G from vertex 0 to vertex n.

Lemma 10. *Let $m \in \{m_{ver}, m_{edg}\}$.*

1. $(\text{Search-TOPDCON}, m) \leq_m^{\text{sL}} (\text{Search-ADSTCON}, m)$.
2. $(\text{Search-LDGGSTCON}, m) \leq_m^{\text{sL}} (\text{Search-3TOPDCON}, m)$.

Proof. We will prove only (2). Let $x = (G, s, t)$ be any instance to Search-LDGGSTCON with $G = (V, E)$. Without loss of generality, we assume that $V = [n]$, $s = (1, 1)$, and $t = (n, n)$. We define a linear ordering $<$ over all vertices as follows: $(1,1) < (1,2) < (2,1) < \cdots < (1,i) < (2,i-1) < \cdots < (i,1) < \cdots < (n,n)$. Since G has only edges of the form $((i,j),(i,j+1))$ and $((i,j),(i+1,j))$, the above ordering satisfies that $(u, v) \in E$ implies $u < v$. This is clearly an instance to Search-$(2,2)$TOPDCON.

Next, we transform Search-$(2,2)$TOPDCON to Search-3TOPDCON. For each vertex v, we can compute the degree $deg(v)$ of v. Given each vertex v, we remove all of its outgoing edges and, instead, add an extra vertex v' and an edge set $\{(v, v'), (v', w) \mid (v, w) \in E\}$. Finally, we define a new linear order $<^*$ as follows: (1) if $u, w \in V$, then $u <^* w$ iff $u < w$, (2) $v <^* v'$, and (3) if $u = v'$, then $v' <^* w$ iff $v < w$. □

Proposition 11. *For every size parameter m taken from $\{m_{ver}, m_{edg}\}$,* $(\text{TOPSORT}, m) \equiv_T^{\text{sL}} (\text{Search-ADST}, m) \leq_m^{\text{sL}} (\text{Search-DST}, m)$.

Proof. Let $m \in \{m_{ver}, m_{edg}\}$. It is obvious that $(\text{Search-ADST}, m) \leq_m^{\text{sL}} (\text{Search-DST}, m)$. Hereafter, we want to show the $\leq_m^{\text{sL}}$-equivalence between TOPSORT and Search-ADST. Let $x = (G, s)$ with $G = (V, E)$ be any instance to TOPSORT. Assume that all vertices in G are linearly ordered. Note that G is a directed acyclic graph. Let T be a depth-first tree of G from r, which is obtained by making a query (G, r) to Search-ADST.

Let $n = |V|$. Recursively, we define L_i for $i \in [n]$. Initially, we set $L_0 = \{r\}$. For each index $i \geq 1$, from T, we determine the set L_i of all vertices in T of distance i from r. We sort L_i's according to the value i and then sort all vertices in L_i by a given linear order. Since T is a spanning tree, this process enumerates all vertices in G connected from r. This establishes $(\text{TOPSORT}, m) \leq_T^{\text{sL}} (\text{Search-ADST}, m)$.

Conversely, given an instance (G, s) to Search-ADST, we make a query to TOPSORT and obtain a topological sorting $(x_1, x_2, \ldots, x_n)$ of G starting at s.

Choose $i = n$ and consider the set $in(x_n)$. Choose the smallest index i for which x_i belongs to $in(x_n)$, keep the edge (x_i, x_n), and discard all the other edges (x_j, x_n) with $x_j \in in(x_n)$. Change i to $i - 1$ and repeat this process until $i < 0$. It is not difficult to show that the resulted graph is a spanning tree of G. □

References

1. Allender, E., Barrington, D.A.M., Chakraborty, T., Datta, S., Roy, S.: Planar and grid graph reachability problems. Theory Comput. Syst. **45**, 675–723 (2009)
2. Allender, E., Mahajan, M.: The complexity of planarity testing. Inf. Comput. **189**, 117–134 (2004)
3. Asano, T., Kirkpatrick, D., Nakagawa, K., Watanabe, O.: $\widetilde{O}(\sqrt{n})$-space and polynomial-time algorithm for planar directed graph reachability. In: Csuhaj-Varjú, E., Dietzfelbinger, M., Ésik, Z. (eds.) MFCS 2014. LNCS, vol. 8635, pp. 45–56. Springer, Heidelberg (2014). doi:10.1007/978-3-662-44465-8_5
4. Barnes, G., Buss, J.F., Ruzzo, W.L., Schieber, B.: A sublinear space, polynomial time algorithm for directed s-t connectivity. SIAM J. Comput. **27**, 1273–1282 (1998)
5. Bourke, C., Tewari, R., Vinodchandran, N.V.: Directed planar reachability is in unambiguous logspace. In: Proceedings CCC 2007, 217–221 (2007)
6. Chakraborty, D., Pavan, A., Tewari, R., Vinodchandran, N.V., Yang, L.F.: New time-space upperbounds for directed reachability in high-genus and H-minor-free graphs. In: The Proceedings of FSTTCS 2014, Leibniz International Proceedings in Informatics, pp. 585–595 (2014)
7. Gross, J.L., Yellen, J., Zhang, P.: Handbook of Graph Theory. CRC Press, Boca Raton (2014)
8. Immerman, N.: Nondeterministic space is closed under complement. SIAM J. Comput. **17**, 935–938 (1988)
9. Jones, N.D.: Space-bounded reducibility among combinatorial problems. J. Comput. Syst. Sci. **11**, 68–75 (1975)
10. Kannan, S., Khanna, S., Roy, S.: STCON in directed unique-path graphs. In: Proceedings of FSTTCS 2008, UPIcs 2, pp. 256–267 (2008)
11. Reingold, O.: Undirected connectivity in log-space. J. ACM **55**, 1–24 (2008)
12. Reingold, O., Trevisan, L., Vadhan, S.P.: Pseudorandom walks on regular digraphs and the RL vs. L problem. In: The Proceedings of STOC 2006, pp. 457–466 (2006)
13. Savitch, W.J.: Relationships between nondeterministic and deterministic tape complexities. J. Comput. Syst. Sci. **4**, 177–192 (1970)
14. Szelepcsényi, R.: The method of forced enumeration for nondeterministic automata. Acta Inf. **26**, 279–284 (1988)
15. Tantau, T.: Logspace optimization problems and their approximation properties. Theory Comput. Syst. **41**, 327–350 (2007)
16. Yamakami, T.: Uniform-circuit and logarithmic-space approximations of refined combinatorial optimization problems. In: Widmayer, P., Xu, Y., Zhu, B. (eds.) COCOA 2013. LNCS, vol. 8287, pp. 318–329. Springer, Cham (2013). doi:10.1007/978-3-319-03780-6_28. A complete version is available at arXiv:1601.01118v1
17. Yamakami, T.: The 2CNF Boolean formula satisfiability problem and the linear space hypothesis. In: The Proceedings of MFCS 2017, 7–11 August 2017. Leibniz International Proceedings in Informatics (2017, to appear)