TECHNISCHE UNIVERSITÄT DARMSTADT

Department of Mathematics

Master Thesis

Quantum Space-Bounded Computation and Graph Connectivity

Roman Edenhofer

17th December, 2023

Supervisors: PD Dr. Kord Eickmeyer, Dr. Simon Apers

Contents

Introduction

Graph connectivity is a central problem in computational complexity theory, and of particular importance in the space-bounded setting. Given a graph G and two vertices, the task is to decide whether there is a path from the first vertex to the second.

For undirected graphs the problem is denoted as USTCON and a beautifully simple random walk algorithm due to Aleliunas et al. from 1979 [1] showed that one can decide it in *randomized logspace*, **RL**. After a long line of work, Reingold was able to derandomize the result in 2008 [2] and showed that USTCON is already in *deterministic logspace*, **L**, and in fact complete for that class.

In the directed setting the problem is denoted as STCON and complete for *non-deterministic logspace*, **NL**. The best known deterministic algorithm in terms of space-complexity is due to Savitch from 1970 [3] and runs in space $O(\log^2 n)$. Since then, a lot of effort has been invested into improving this bound and for restricted classes of graphs better bounds are known. For example for *strongly-unambiguous* graphs, also called *mangroves*, for which the number of paths between any two nodes is at most one, Allender and Lange [4] were able to beat Savitch's upper bound by presenting a deterministic $O(\log^2 n/\log\log n)$-space algorithm. Due to a result of Garvin et al. [5] their algorithm actually extends to the larger class of *reach-few* graphs for which the number of paths to all nodes that are reachable from the first one is polynomially bounded.

As our main contribution, we show that in the quantum setting we can further improve on their bound for the restricted class of *strongly-few* graphs for which the number of paths between any two nodes is polynomially bounded. So far, mostly linear algebraic problems have been considered for space-bounded quantum algorithms. Ta-Shma showed in [6] how to space-efficiently approximate the inverse of a well-conditioned matrix. His algorithm runs in *bounded error quantum logspace*, **BQL**, if the condition number of the matrix κ is polynomial. This improved on the best known classical procedure for matrix inversion due to Csanky [7] running in space $O(\log^2 n)$. Unfortunately, most common matrices associated to directed graphs have exponential condition number, and so Ta-Shma's algorithm does not put STCON in **BQL**. However, for strongly-few graphs we find that we can do so by considering the operator $\mathcal{L}^{\#} = I - A$ where I is the identity and A the adjacency matrix of a digraph. We call that operator the counting Laplacian. A spectral analysis of it shows that it has polynomial condition number if and only if the underlying graph is strongly-few. This allows us to prove our main result.

Theorem 1. STCON *restricted to strongly-few graphs can be decided in* **BQL**.

The thesis is structered as follows. In the first chapter we define our computational model and give an overview of central complexity classes, in particular classes related to unambiguity and fewness. We further discuss the use of classical random walks on directed and undirected graphs to decide connectivity. In the second chapter we give a high-level description of the quantum algorithm for linear system solving due to Harrow, Hassidim and Lloyd [8] which is usually referred to as the HHL-algorithm. This algorithm is at the heart of the matrix inversion result from Ta-Shma. The reader who is already familiar with it or not interested in the quantum part of this thesis may skip this part. In the third and last chapter we prove our main result that STCON restricted to strongly-few graphs is in **BQL**. We also make a comparison of the resulting procedure with random walks.

Acknowledgements

First and foremost, I want to express my sincere appreciation to my supervisors, Dr. Simon Apers and PD. Dr. Kord Eickmeyer, for entrusting me with this fascinating topic. I am especially grateful to Dr. Apers for the insightful weekly discussions that significantly contributed to my understanding of the subject, and to PD. Dr. Kord Eickmeyer for his facilitation of my stay abroad during the thesis. Their guidance has been invaluable to the successful completion of this thesis. Furthermore, I would like to offer my special thanks to my family for their continuous support throughout my life.

Chapter 1

Preliminaries

1.1 The Computational Model

In this section we define our computational model and introduce some space-bounded complexity classes. All of the complexity classes that we define are classes of promise problems. Promise problems are a generalization of decision problems where the input is chosen from a subset of all possible binary strings. They allow to more accurately capture the computational complexity of some tasks because verifying whether a given input satisfies a promised property can be hard, in fact it can even be harder than the actual task one is interested in.

Formally, a promise problem is a pair $L = (L_{\text{yes}}, L_{\text{no}})$, where $L_{\text{yes}}, L_{\text{no}} \subseteq \{0,1\}^*$ are sets of binary strings satisfying $L_{\text{yes}} \cap L_{\text{no}} = \emptyset$. The strings in L_{yes} are called yes-instances and the strings in L_{no} are called no-instances. The union $L_{\text{yes}} \cup L_{\text{no}}$ is called the promise. A machine solving a promise problem is only required to distinguish the yes- and no-instances. It is allowed to do whatever on inputs outside the promise. Of key interest to us in this thesis are the following two problems.

USTCON

Input:	Adjacency matrix A of an undirected graph G with nodes $v_1, ..., v_n$.
Output:	1 if there is a path from v_1 to v_n, 0 otherwise.

STCON

Input:	Adjacency matrix A of a directed graph G with nodes $v_1, ..., v_n$.
Output:	1 if there is a path from v_1 to v_n, 0 otherwise.

A space-bounded deterministic Turing machine (DTM) acts according to a transition function δ on three semi-infinite tapes: A read-only tape where the input is stored, a read-and-write work tape and a uni-directional write-only tape for the output. The space-complexity is defined as the number of used cells on the work tape. A function $s : \mathbb{N} \to \mathbb{N}$ is said to be space-constructible if there is a DTM M that computes $s(n)$ on any n-bit input in space $O(s(n))$. We restrict ourselves to such space-bounds in the following.

A non-deterministic Turing machine (NTM) is similar to a DTM except that it has two transition functions δ_0 and δ_1. At each step in time the machine non-deterministically chooses to apply either one of the two. It is said to accept an input x if there is a sequence of these choices so that it reaches an accepting state and it is said to reject input x if there is no such sequence of choices.

This gives rise to the following complexity classes.

Definition 2. DSPACE($s(n)$) (resp. NSPACE($s(n)$)) is the class of promise problems $L = (L_{\text{yes}}, L_{\text{no}})$ for which there exists a DTM (resp. NTM) M running in space $O(s(n))$ such that

- (Completeness) for all $x \in L_{\text{yes}}$, we find that M accepts x and
- (Soundness) for all $x \in L_{\text{no}}$, we find that M rejects x.

Further, let $\mathbf{L}^k :=$ DSPACE($\log^k n$) and $\mathbf{NL} :=$ NSPACE($\log n$).

We write $\mathbf{L}$ for $\mathbf{L}^1$. As is common, we somewhat overload the term in the determinsitc case and say that a boolean function f is computable in DSPACE($s(n)$) if there is a DTM running in space $O(s(n))$ that outputs $f(x)$ on input x.

A probabilistic Turing machine (PTM) is a DTM with the ability to toss random coins. This can be conveniently formulated by a fourth random-coins tape that is uni-directional, read-only and initialized with random bits.

Instead of this random-coins tape, a quantum Turing machine (QTM) has a fourth read-and-write quantum work tape. This tape is made up of qubits with two tape-heads moving on it. The operations on the qubits are chosen from some universal gate set, say $\{\mathrm{HAD}, \mathrm{CNOT}, \mathrm{T}\}$. At each point in time the two heads can either apply a gate from the gate set to the qubits below them or perform a measurement to a projection in the standard basis. The space-complexity is given by the used cells of the classical and of the quantum work tape. Note that every universal gate set needs to contain a gate that acts on two or more qubits, making two tape heads on the quantum tape mandatory.

This gives rise to the following probabilistic and quantum complexity classes.

Definition 3. $\mathrm{BSPACE}_{a,b}(s(n))$ (resp. $\mathrm{BQSPACE}_{a,b}(s(n))$) is the class of promise problems $L = (L_{\text{yes}}, L_{\text{no}})$ for which there exists PTM (resp. QTM) M running in space $O(s(n))$ and time $2^{O(s(n))}$ such that

- (Completeness) for all $x \in L_{\text{yes}}$, we find $\mathbb{P}[M \text{ accepts } x] \geq a$ and
- (Soundness) for all $x \in L_{\text{no}}$, we find $\mathbb{P}[M \text{ accepts } x] \leq b$.

We say that a is the completeness error and b is the soundness error. Further $\mathbf{RL} := \mathrm{BSPACE}_{\frac{1}{2},0}(\log n)$, $\mathbf{BPL} := \mathrm{BSPACE}_{\frac{2}{3},\frac{1}{3}}(\log n)$ and $\mathbf{BQL} := \mathrm{BQSPACE}_{\frac{2}{3},\frac{1}{3}}(\log n)$.

We again overload the terms and say that a boolean function is computable in $\mathrm{BSPACE}(s(n))$ (resp. $\mathrm{BQSPACE}(s(n))$) if there is a PTM (resp. QTM) running in space $O(s(n))$ that outputs $f(x)$ on input x with bounded error.

Clearly, our definition of $\mathrm{BQSPACE}(s(n))$ allows intermediate measurements. It was recently shown in [9] that delaying all measurements until the end of the computation does not weaken the computational model. Similarly, the specific choice of the universal gate set does not alter the resulting complexity class due to the space-efficient version of the Solovay-Kitaev theorem in [10].

Space-bounded computation is closely related to parallel computation. The standard model for which are bounded-depth circuits. We define the most important complexity classes below.

Definition 4. $\mathbf{NC}^k$ is the class of promise problems solved by $\mathbf{L}$-uniform polynomial-size circuits of depth $O(\log^k n)$ with fanin 2 where all gates are chosen from the universal set $\{\wedge, \vee, \neg\}$. $\mathbf{AC}^k$ is the same, except for unbounded fanin.

Lastly, let intDET denote the problem of computing the determinant of an $n \times n$ n-bit integer matrix, as was introduced by Cook in [11]. The complexity class $\mathbf{DET}$ is the collection of all problems that are $\mathbf{NC}^1$-reducible to intDET. Cook showed that many natural linear algebraic problems are in fact $\mathbf{DET}$-complete, that is intDET, intMATINV (the problem of computing the inverse of an integer matrix), intITMATPROD (the problem of computing the product of n integer matrices) and intMATPOW (the problem of computing the first n powers of an integer matrix).

Fefferman and Remscrim defined poly-conditioned promise versions of these problems in [9] and showed that all of them are complete for $\mathbf{BQL}$.

See figure 1.1 for the known inclusions of all the introduced complexity classes. Proof sketches for them can be found in the appendix.

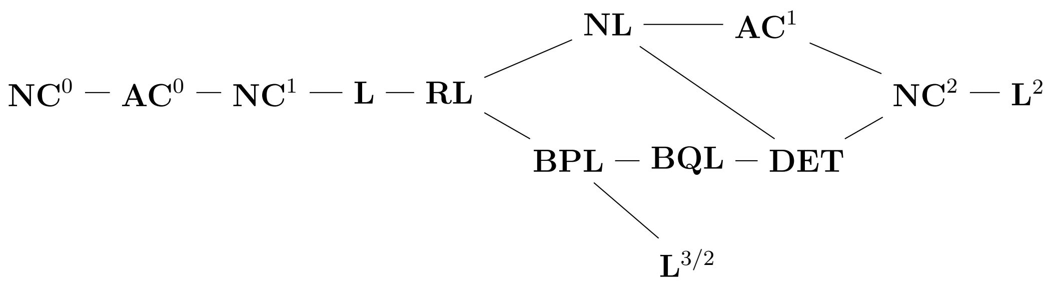


Figure 1.1: Inclusion diagram for space-bounded and related complexity classes.

1.2 Random Walks on Graphs

In this section we consider random walks on directed and undirected graphs. This has two reasons. First, they are the natural classical approach to space-efficiently solve connectivity. Essentially, the only required space of a random walk algorithm is that of a timer to make sure that the algorithm halts at some point. Second, we find a close corelation of when a random walk succeeds to decide connectivity in **RL** and when the spectral properties of graph related matrices are well-behaved and also allow for a direct **BQL**-procedure. We start with a short introduction to the topic.

A random walk on a (directed or undirected) graph $G = (V, E)$ starts from some initial vertex (sampled from some distribution p_0 over V represented by a row vector) and at every timestep jumps uniformly at random to one of its (out-)neighbouring vertices. This gives rise to a finite Markov Chain $(X_t)_{t\in\mathbb{N}}$ on the state space V with transition probability matrix P such that

$$P(i,j) = \mathbb{P}[X_{t+1} = j | X_t = i] = \begin{cases} \frac{1}{d(i)} & \text{if } (i,j) \in E, \\ 0 & \text{otherwise.} \end{cases}$$

Here $d(i)$ denotes the (out-)degree of vertex $i \in V$. The distribution after t steps of the random walk is $p_t = p_0 P^t$.

The Markov Chain is said to be *irreducible* if the underlying graph is *strongly connected*, that is every vertex is reachable from every other vertex. If this is the case, then there is a unique *stationary distribution* π satisfying $\pi P = \pi$. Furthermore, if the Markov Chain is also *aperiodic*, that is there is some $k \in \mathbb{N}^*$ such that $P^k(i,j) > 0$ for all $(i,j) \in V^2$, then every initial distribution converges to the stationary one, i.e.

$$p_0 P^t \xrightarrow{t\to\infty} \pi.$$

The time it takes to get close to the stationary distribution is quantified by the *mixing time*. Formally, this is

$$\mathrm{MT}(\varepsilon) := \min\{t \in \mathbb{N} : ||p_0 P^t - \pi||_{\mathrm{TV}} \leq \varepsilon \ \forall p_0\}$$

where $||\cdot||_{\mathrm{TV}}$ denotes the *total variation distance*. For two distributions μ and ν over V the total variation distance is given by

$$||\mu - \nu||_{\mathrm{TV}} := \max_{A \subseteq V} |\mu(A) - \nu(A)| = \frac{1}{2}||\mu - \nu||_1.$$

We make two remarks about the mixing time. First, in the minimum of the definition it suffices to consider only initial distributions concentrated at a single node, i.e. if for some $t \geq 0$ we find $||e_i P^t - \pi||_{\mathrm{TV}} \leq \varepsilon$ for all $i \in V$, then already $\mathrm{MT}(\varepsilon) \leq t$. Here e_i denotes the row vector that is 1 only at index i and 0 everywhere else. Second, choosing a different ε, as long as $\varepsilon < 1/2$, changes the value of the mixing time only by a constant factor. It is thus a convention to fix $\varepsilon = \frac{1}{4}$ whenever referring to ‘the’ mixing time.

1.2.1 The undirected case

In this subsection we show that USTCON $\in$ **RL** as in [1]. For this, we first observe that for undirected connected graphs the stationary distrbution is given by

$$\pi(i) = \frac{d(i)}{2|E|}.$$

We define the *hitting time* $\mathrm{HT}(i,j)$ as the expected number of steps until a random walk starting from i reaches j,

$$\mathrm{HT}(i,j) := \mathbb{E}[\min\{t \geq 0 : X_t = j\} | X_0 = i].$$

The *return time* from i to itself is given by

$$\mathrm{HT}^+(i,i) := \mathbb{E}[\min\{t > 0 : X_t = i\} | X_0 = i].$$

Further, we define the *cover time* from i, $\mathrm{CT}(i)$, as the expected number of steps until a random walk starting from i has visited all vertices,

$$\mathrm{CT}(i) := \mathbb{E}[\min\{t \geq 0 : \cup_{k=0}^{t} X_k = V\} | X_0 = i].$$

The cover time of the graph is then given by the cover time from the worst-possible starting node, i.e. $\mathrm{CT}(G) := \max_{i \in V} \mathrm{CT}(i)$. Clearly, the cover time is an upper bound for the hitting time, $\mathrm{HT}(i,j) \leq \mathrm{CT}(G)$ for all $(i,j) \in V^2$.

We find that the cover time of any undirected connected graph is polynomial.

Lemma 5. *Every undirected connected graph $G = (V, E)$ with $n = |V|$ vertices has cover time $\mathrm{CT}(G) = O(n^3)$. More precisely, it is bounded by*

$$2 \cdot (|V| - 1) \cdot 2|E| \leq 2 \cdot n(n-1)^2.$$

Proof. The return time from any node i to itself is given by $\mathrm{HT}^+(u,u) = \frac{2|E|}{d(u)}$ which is the reciprocal of its stationary probability. Further, the expected return time can also be calculated via the hitting time from every neighbour j to i:

$$\frac{2|E|}{d(i)} = \mathrm{HT}^+(i,i) = 1 + \frac{1}{d(i)} \sum_{\{i,j\} \in E} \mathrm{HT}(j,i).$$

It follows that for any edge $\{i,j\} \in E$ the hitting time from i to j is bounded by $\mathrm{HT}(i,j) < 2|E|$.

Now let T be a spanning tree of G. Note that any traversal of T need not visit an edge more than twice and is thus made up of no more than $2n - 2$ nodes. Let $(v_0, ..., v_{2n-2} = v_0)$ be a traversal of T. We find

$$\mathrm{CT}(G) \leq \mathrm{HT}(v_0, v_1) + ... + \mathrm{HT}(v_{2n-3}, v_{2n-2}) < (2n-2) \cdot 2|E|. \qquad \square$$

This asymptotic bound is tight as can be seen by the following example.

Example 6. The lollipop graph of $2n-1$ nodes has cover time $\Theta(n^3)$. The graph is made up of a path of length n that is connected at one end to a complete component of n nodes, compare figure 6. The hitting times of the graph can be calculated by solving a linear system of recurrences because for each pair of nodes $(i,j) \in V^2$ we have

$$\mathrm{HT}(i,j) = 1 + \frac{1}{d(i)} \sum_{u \in N(i)} \mathrm{HT}(u,j)$$

where $N(i) = \{u \in V : \{i,u\} \in E\}$ is the neighborhood of vertex i.

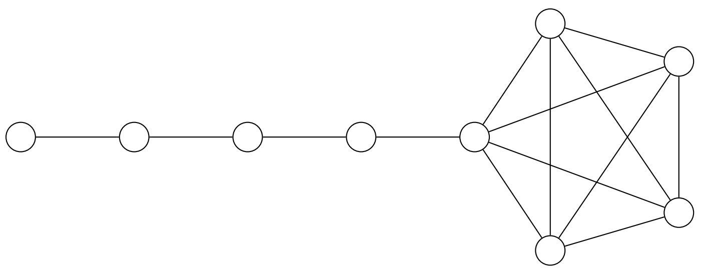


Figure 1.2: The lollipop graph for $n = 5$.

We do not solve this system here but instead give an intuitive explanation for the cover time being cubic. If we start a random walk at any node from the complete component except for the one connected to the path, then the probability to reach the single node that is part of the path is $1/(n-1)$, so we need roughly n trials until we get access to it. Once we are located at that node, the probability to jump into the path and not back to the complete component is $1/n$, making another n trials necessary. And finally, the expected number of attempts to reach the other end of the path before going back to the end the walk started in is n again. This follows from the well-known problem called Gambler's Ruin [12]. So, all in all we need roughly n^3 steps of the random walk until we hit the leftmost node of the path.

This polynomial bound on the cover time suffices to show

Corollary 7. *USTCONN* $\in$ **RL**.

Proof. Given an undirected graph G with nodes $v_1, ..., v_n$, we start a random-walk from v_1 and keep a counter for every step we take. After $2n^3$ steps we stop and return 1 if we reached t at some point, otherwise we return 0. The required space for the counter is $\log(2n^3) = O(\log n)$. If v_1 and v_n are connected, then the hitting time is bounded by $\mathrm{HT}(v_1, v_n) \leq \mathrm{CT}(G) < 2n^3$ and we know that the probability to reach v_n before $2n^3$ steps is greater than $1/2$ by Markov bound. $\square$

As mentioned in the introduction, this result has been derandomized by Reingold who showed that USTCON is already contained in **L**.

We now define the *(combinatorial) Laplacian* of G as $\mathcal{L} := D - A$ where $D = \mathrm{diag}(d(v_1), ..., d(v_n))$ is the diagonal (out-)degree matrix and A is the adjacency matrix of some undirected graph G. Analyzing its eigenvalues will give a direct **BQL**-procedure to decide USTCON. For this, we first define the *vertex-edge* *incidence matrix* $B \in \{0,1\}^{|V|\times|E|}$ via

$$B(v,e) := \begin{cases} 1 & \text{if vertex } v \text{ is incident to edge } e, \\ 0 & \text{otherwise.} \end{cases}$$

It is easily verified that $\mathcal{L} = B^T B$ so that for all (column) vectors $x \in \mathbb{R}^n$, $x^T \mathcal{L} x = \sum_{(i,j)\in E}(x_i - x_j)^2$. Thus, the combinatorial Laplacian is positive semi-definite. Furthermore, we compute that the all ones vector $\mathbf{1}$ is part of its kernel $\mathcal{L}\mathbf{1} = 0$. In fact, the dimension of the kernel is equal to the number of connected components. This follows in particular from the following lower bound on the second smallest eigenvalue of $\mathcal{L}$ for a connected graph taken from [13].

Lemma 8. *Let G be an undirected graph G that is connected and has diameter* $\operatorname{diam}(G) := \max_{u,v\in V} d(u,v)$. *Then the second smallest eigenvalue of the combinatorial Laplacian $\mathcal{L} = D - A$ is lower bounded by*

$$\lambda_{n-1} \geq \frac{1}{n \cdot \operatorname{diam}(G)}.$$

Proof. Let x be the corresponding eigenvector to the eigenvalue λ_{n-1} and choose $u \in V$ such that $|x_u| = \max_{i\in V}|x_i|$. Since the Laplacian $\mathcal{L}$ is symmetric, we find that x is orthogonal to the all ones vector $\mathbf{1}$ from the kernel of $\mathcal{L}$, i.e. $\sum_{i\in V} x_i = 0$. From this we know that there must be some node v such that $x_u \cdot x_v < 0$. Now let P be a path from u to v of length $T \leq \operatorname{diam}(G)$. We obtain

$$\begin{aligned} \lambda_{n-1} = \frac{x^T \mathcal{L} x}{||x||^2} &= \frac{\sum_{\{i,j\}\in E}(x_i - x_j)^2}{\sum_{i\in V} x_i^2} \\ &\geq \frac{\sum_{(i,j)\in P}(x_i - x_j)^2}{n x_u^2} \\ &\geq \frac{(x_u - x_v)^2/T}{n x_u^2} \\ &\geq \frac{1}{nT} \geq \frac{1}{n \cdot \operatorname{diam}(G)} \end{aligned}$$

where the second inequality follows from Cauchy-Schwarz applied to the T-dimensional all ones vector and the vector containing as entries the differences $(x_i - x_j)$ for $(i,j) \in P$. □

Ta-Shma showed in [6, theorem 5.1] that if the eigenvalues of a hermitian matrix are polynomially bounded, then we can approximate them with inverse polynomial accuracy and estimate their multiplicities in **BQL**. The last lemma assures that the smallest non-zero eigenvalue of the combinatorial Laplacian $\mathcal{L}$ of any undirected graph has inverse polynomial distance from 0. Hence, the accuracy is sufficient and we can estimate the dimension of the kernel of $\mathcal{L}$ to count the number of connected components. In particular, this allows to decide USTCON because counting the components before and after adding an edge between v_1 and v_n allows to decide whether they belong to the same component.

1.2.2 The directed case

Now let $G = (V, E)$ with $n = |V|$ vertices be a directed graph with unique stationary distribution π and probability transition matrix P. Following Reingold et al. [14] we define a normalized inner product on $\mathbb{R}^n$

$$\langle x, y\rangle_\pi := \sum_{i\in V} \frac{x(i)\cdot y(i)}{\pi(i)}.$$

This gives rise to a norm $||x||_\pi := \sqrt{\langle x, x\rangle_\pi}$ and allows us to make the following

Definition 9. The *spectral expansion* of G is defined as

$$\lambda_\pi(G) := \max_{\langle x,\pi\rangle_\pi = 0} \frac{||xP||_\pi}{||x||_\pi}.$$

The spectral expansion coincides with the second largest eigenvalue $\lambda_2(P)$ (in absolute value) in the case that the underlying graph is undirected. In the directed case, P need not be diagonalizable and $\lambda_\pi(G)$ equals the square root of the second largest eigenvalue (in absolute value) of $\tilde{P}P$ where $\tilde{P}(i,j) = \pi(i)P(i,j)/\pi(j)$. The spectral expansion is closely related to the mixing time. If it is small, then the Markov Chain mixes quickly:

Lemma 10. *Let P be irreducible and aperiodic, and let p_0 be any initial distribution over V. Then*

$$||p_0P^t - \pi||_\pi \le \lambda_\pi(G)^t ||p_0 - \pi||_\pi.$$

A well-known method to analyze the mixing time of a Markov Chain is by considering its bottlenecks. This is formalized in the definition below.

Definition 11. The *ergodic flow* from a set $A \subseteq V$ to $B \subseteq V$ is defined as $Q(A, B) := \sum_{x\in A, y\in B} \pi(x)P(x, y)$. The *conductance* of a set $A \subseteq V$ is defined as

$$\Phi(A) := \frac{Q(A, A^C)}{\pi(A)} = \frac{\sum_{x \in A, y \notin A} \pi(x) P(x, y)}{\sum_{x \in A} \pi(x)}.$$

Further, the conductance of G is $\Phi := \min_{\pi(A) \leq \frac{1}{2}} \Phi(A)$.

The conductance gives the following lower bound on the mixing time taken from [12].

Lemma 12. *Let P be irreducible and aperiodic. Then the mixing time is lower bounded by* $\mathrm{MT}(\frac{1}{4}) \geq \frac{1}{4\Phi}$.

Proof. For all $A \subseteq V$ we have

$$\begin{aligned} \mathbb{P}_\pi[X_0 \in A, X_t \in A^C] &\leq \sum_{k=1}^{t} \mathbb{P}_\pi[X_{k-1} \in A, X_k \in A^C] \\ &= t \cdot \mathbb{P}_\pi[X_0 \in A, X_1 \in A^C] \\ &= t \cdot Q(A, A^C). \end{aligned}$$

Dividing by $\pi(A)$ gives

$$\mathbb{P}_\pi[X_t \in A^C | X_0 \in A] \leq t\Phi(A).$$

Hence, there is at least on $x \in A$ such that

$$P^t(x, A) \geq 1 - t\Phi(A).$$

From this we find

$$||p_t - \pi||_{\mathrm{TV}} \geq |p_t(A) - \pi(A)| \geq 1 - t\Phi(A) - \pi(A)$$

which in the case of $\pi(A) \leq \frac{1}{2}$ is smaller or equal to $\frac{1}{4}$ only if $t \geq \frac{1}{4\Phi(A)}$. Maximizing over all A with $\pi(A) \leq \frac{1}{2}$ thus gives us the desired bound. □

The conductance also gives a very powerful upper bound on the mixing time via the following lemma. We refer to [14] for a proof .

Lemma 13. *Let P be irreducible and strongly aperiodic, that is $P(i,i) \geq 1/2$ for all $i \in V$. Then*

$$\lambda_\pi(G) \leq 1 - \frac{\Phi^2}{2}.$$

Two difficulties arise for deciding connectivity on directed graphs that do not appear for undirected graphs.

1. The stationary distribution of an irreducible component can have negligible weight on some nodes. This means that even if we are able to sample from this stationary distribution π (or a distribution that is close to it), we might need an unfeasible amount of samples to learn whether or not a node $v \in \text{supp}(\pi)$.

2. The mixing time need not be polynomial so that we cannot sample from the stationary distribution in reasonable time.

For an example of the first point see figure 1.3. Let P denote the probability transition matrix of the given graph G. It is easy to verify that the stationary distribution of G is $\pi = [\frac{1}{2}, \frac{1}{4}, ..., \frac{1}{2^{n-1}}, \frac{1}{2^{n-1}}]$. A random walk that starts at node 1 and runs for a polynomial number of steps is thus not expected to find that 1 and n are connected. On the other hand, interestingly, the mixing time is constant. One easily verifies that after 2 steps of the random walk from any initial vertex i, we have $||e_iP^2 - \pi||_{\text{TV}} \leq \frac{1}{4}$.

For an example of the second point consider figure 1.4 of a digraph G' with probability transition matrix P'. Let π still denote the stationary distribution of P from figure 1.3 and let $\pi^{(\text{rev})}$ be equal to π but in reverse order, i.e. $\pi^{(\text{rev})} = [\frac{1}{2^{n-1}}, \frac{1}{2^{n-1}}, ..., \frac{1}{4}, \frac{1}{2}]$. The stationary distribution of P' is then given by $\pi' = \frac{1}{2}[\pi, \pi^{(\text{rev})}]$. We find for the conductance of $[n]$

$$\Phi([n], [2n]\backslash[n]) = \frac{\pi'(n)P'(n, n+1)}{\pi'([n])} = \frac{1/2^{n+1}}{1/2} = \frac{1}{2^n}.$$

Hence, by lemma 12, the mixing time is exponential. Furthermore, we find that the second largest singular value of P' is exponentially close to 1. To see this, consider the vector $x' := \frac{1}{2}[\pi, -\pi^{(\text{rev})}]$ that is orthogonal to π' and has norm $||x'||_2 \geq 1/4$. We find that

$$xP = \frac{1}{2}[\pi_{\{1,...,n-1\}}, 0, 0, -\pi^{(\text{rev})}_{\{2,...,n\}}]$$

so that

$$s_2(P)^2 = \max_{\langle x, \pi' \rangle = 0} \frac{||xP||_2^2}{||x||_2^2} \leq \frac{||x'||_2^2 - 2 \cdot \left(\frac{1}{2^{n-1}}\right)^2}{||x'||_2^2} = 1 - 2^{-\Omega(n)}.$$

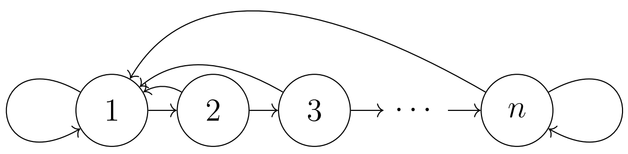


Figure 1.3: Digraph G with inverse exponential stationary probabilities but constant mixing time.

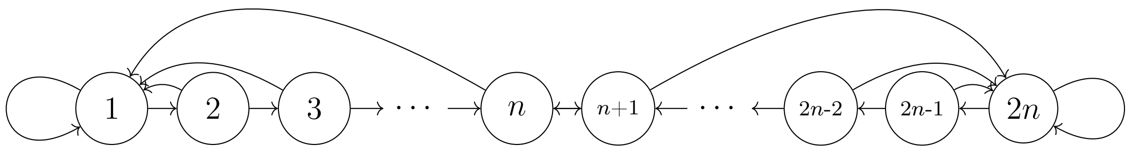


Figure 1.4: Digraph G' with exponential mixing time.

1.3 Unambiguity and Fewness

The notions of unambiguity and fewness are concerned with counting the number of paths in a graph. This really only make sense for graphs that are acyclic (DAGs).

Luckily, we can assume any graph to be of that type by a simple reduction.

Lemma 14. STCON *is* $\mathbf{AC}^0$*-reducible to* STCON *restricted to DAGs.*

Proof. Let G be a digraph with nodes $v_1, ..., v_n$ that potentially contains a cycle. Further, let $T \leq n-1$ be some bound on the diameter of G. We construct a layered graph G' corresponding to a clocked walk on G of time T. G' contains $(T+1)$-many layers with n nodes each. The nodes of the t-th layer are labelled $v_1^{(t)}, ..., v_n^{(t)}$. We now add two types of edges for every $t \in \{0, ..., T-1\}$. First, for all edges (v_i, v_j) in G, we add an edge $\left(v_i^{(t)}, v_j^{(t+1)}\right)$ allowing the walk to take that edge at time t. Second, we add outgoing edges from $v_n^{(t)}$ to $v_n^{(t+1)}$ allowing the walk to wait on the last node. See figure 1.5 for an example of this reduction.

Clearly, G' is acyclic because edges go only up in layers and never down. And there is a path from v_1 to v_n in G if and only if there is one from $v_1^{(0)}$ to $v_n^{(T)}$ in G'. The reduction can be implemented in $\mathbf{AC}^0$. □

There are three notions of unambiguity and fewness that have been studied in the literature. We capture them in the following definition.

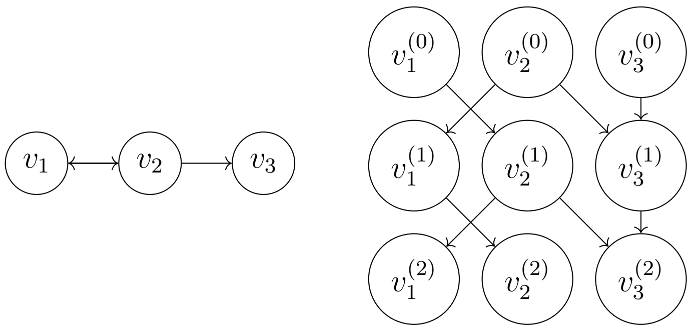


Figure 1.5: On the left a digraph G containing a cycle and on the right the constructed layered DAG G'.

Definition 15. A DAG G with nodes $V = \{1, ..., n\}$ is called

1. *unambiguous* if there is at most one path from 1 to n,
2. *reach-unambiguous* if there is at most one path from 1 to every node that is reachable from 1, and
3. *strongly unambiguous*, or a *mangrove*, if there is at most one path between any two nodes.

See figure 1.6 for examples. Relaxing the restriction on the number of paths to at most polynomially many gives rise to three fewness notions: G is called

1. *few-unambiguous* if there are at most polynomially many paths from 1 to n,
2. *reach-few-unambiguous* (for short *reach-few*) if there are at most polynomially many paths from 1 to every node that is reachable from 1, and
3. *strongly-few-unambiguous* (for short *strongly-few*) if there are at most polynomially many paths between any two nodes.

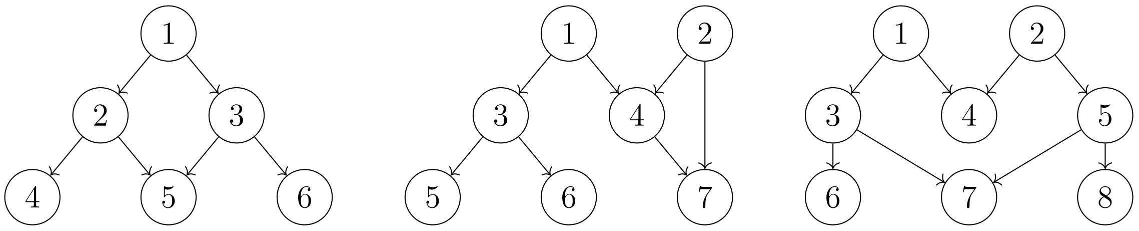


Figure 1.6: On the left an unambiguous, in the middle a reach-unambiguous and on the right a strongly-unambiguous graph. From [15].

The previous definition naturally gives rise to six complexity classes which are semantically defined as the classes of problems accepted by **NL**-machines for which

$$\begin{tikzcd}
\mathbf{L} \arrow[r, dash] & \mathbf{StUL} \arrow[r, dash] & \mathbf{StFewL} \arrow[r, dash] \arrow[dr, dash] & \begin{array}{c}\mathbf{ReachFewL}\\ = \mathbf{ReachUL}\end{array} \arrow[r, dash] \arrow[dr, dash] & \mathbf{UL} \arrow[r, dash] & \mathbf{FewL} \arrow[r, dash] & \mathbf{NL} \\
 & & & \mathbf{BQL} & \mathrm{DSPACE}(\frac{\log^2 n}{\log\log n}) & &
\end{tikzcd}$$

Figure 1.7: Inclusion diagram for unambiguity and fewness complexity classes.

the configuration graph satisfies the respective unambiguity or fewness conditions. We list them in inclusion order

- *strongly-unambiguous logspace*, **StUL**,
- *strongly-few logspace*, **StFewL**,
- *reach-unambiguous logspace*, **ReachUL**,
- *reach-few logspace*, **ReachFewL**,
- *unambiguous logspace*, **UL**,
- *few logspace*, **FewL**.

Except for the inclusions **StFewL** $\subseteq$ **ReachUL** and **ReachFewL** $\subseteq$ **UL** all of them follow directly from the definition. As mentioned in the introduction, Allender and Lange showed in [4] that there is a DSPACE($\log^2 n/\log\log n$) algorithm solving STCON on mangroves. In fact, they mention themselves that their algorithm also works for reach-unambiguous graphs. Furthermore, Garvin et al. showed in [5] that **ReachUL** = **ReachFewL**. Both of their results thus imply **ReachFewL** = **ReachUL** $\subseteq$ DSPACE($\log^2 n/\log\log n$). Last but not least, in chapter 3 we show that **StFewL** $\subseteq$ **BQL**. Putting all together we obtain the inclusion diagram depicted in figure 1.7.

Chapter 2

HHL-algorithm

In this chapter we give some background on Ta-Shma's result that approximately inverting well-contioned matrices is in **BQL**. For a rigorous treatment, we refer to his paper [6]. As is common in the quantum setting, we use the bra-ket notation. Loosely speaking, a ket $|x\rangle$ is a normalized column vector and a bra $\langle x|$ is a normalized row vector. Ta-Shma's work builds on the HHL-algorithm due to Harrow, Hassidim and Lloyd [8] that we sketch in the following. The algorithm solves a quantum version of the κ-conditioned linear system problem (κ-QLSP) for hermitian matrices.

κ-QLSP FOR HERMITIAN MATRICES

Input:	A hermitian matrix $H \in \mathbb{C}^{n\times n}$ with singular values satisfying $1 \geq s_1(H) \geq ... \geq s_n(H) \geq 1/\kappa$ and an n-dimensional state $|b\rangle$.
Output:	An approximation of a state $|x\rangle$ that is proportional to $H^{-1}|b\rangle$.

The algorithm uses two subroutines: One for Hamiltonian Simulation and one for Quantum Phase Estimation (QPE). We explain them in the following sections.

2.1 Hamiltonian Simulation

Hamiltonian Simulation is the problem of outputting a circuit that approximates e^{iH} given a hermitian matrix H.

HAMILTONIAN SIMULATION

Input:	A hermitian matrix $H \in \mathbb{C}^{n\times n}$ with $\|\|H\|\| \leq 1$.
Output:	A circuit U that ε-approximates e^{iH}, i.e. such that $\|\|U - e^{iH}\|\| \leq \varepsilon$.

Note that hermitian matrices $H \in \mathbb{C}^{n\times n}$ are diagonalizable and all its eigenvalues are real. Hence, we can write any such matrix in its eigenbasis $H = \sum_{j=1}^{n} \lambda_j |h_j\rangle \langle h_j|$ and we find that $e^{iH} = \sum_{j=1}^{n} e^{i\lambda_j} |h_j\rangle \langle h_j|$ is unitary. Ta-Shma showed how to compute the desired circuit in DSPACE$(\log n + \log \varepsilon^{-1})$.

2.2 Quantum Phase Estimation

Eigenvalues of unitary operators have unit modulus and can therefore be characterized by their phase. If $|\psi\rangle$ is an eigenvector of a unitary U, then there is some $\theta \in [0,1]$ such that $U|\psi\rangle = e^{2\pi i\theta}|\psi\rangle$. Quantum Phase Estimation is the problem of determining the phase θ given access to U and $|\psi\rangle$.

QPE	
Input:	A unitary $U \in \mathbb{C}^{n\times n}$ and a corresponding eigenvector $\|\psi\rangle$.
Output:	An approximation of the phase $\theta \in [0,1]$ of the eigenvector $\|\psi\rangle$.

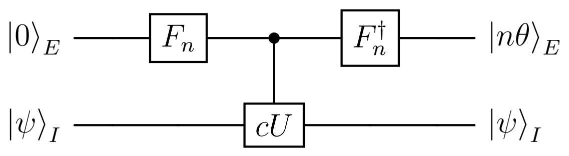
The circuit solving it uses the Quantum Fourier Transform (QFT) $F_n : \mathbb{C}^n \to \mathbb{C}^n$. This is a fundamental building block of many quantum algorithms. It is the unitary transformation that maps any vector from the computational basis to its so called Fourier mode

$$F_n |k\rangle := \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} \omega_n^{kj} |j\rangle$$

where $\omega_n = e^{2\pi i/n}$ denotes the n-th root of unity.

See e.g. [16] for a proof that the QFT can be implemented by a circuit made up of $\Theta(\log^2 n)$ simple gates. It is straightforward to see that this circuit is in fact **L**-uniform.

Now suppose we have access to a unitary U and one of its eigenvectors $|\psi\rangle$ with unknown phase θ. For simplicity, we assume that $n\theta$ is an integer. The following circuit allows to determine the phase θ. It works on the product space of two registers: A phase estimation register $|\cdot\rangle_E$ and an input register $|\cdot\rangle_I$.



Here the cU-gate is an application of U on the input register conditioned on the value of the estimation register. It maps any state $|j\rangle_E |\psi\rangle_I$ to $U^j |j\rangle_E |\psi\rangle_I$.

Let us analyze step by step what the circuit does to the initial state

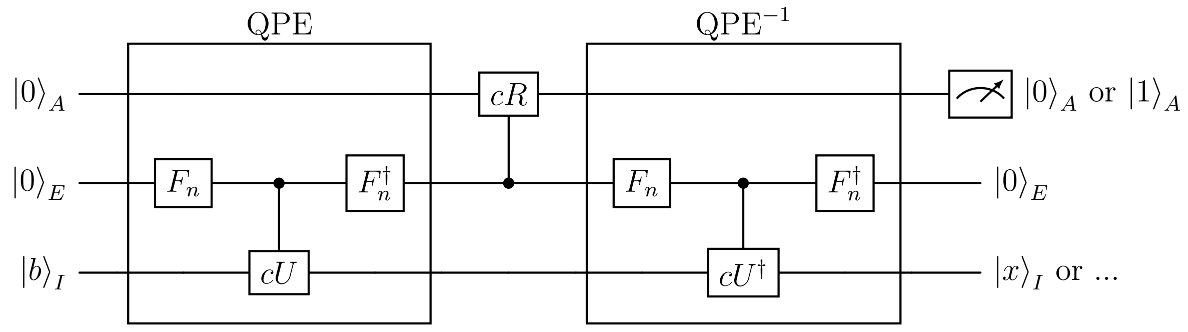
$$\begin{aligned}
|0\rangle_E |\psi\rangle_I &\overset{F_N}{\longmapsto} \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} |j\rangle_E |\psi\rangle_I \\
&\overset{cU}{\longmapsto} \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} U^j |j\rangle_E |\psi\rangle_I = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} e^{2\pi i\theta j} |j\rangle_E |\psi\rangle_I = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} \omega_n^{n\theta j} |j\rangle_E |\psi\rangle_I \\
&\overset{F_n^\dagger}{\longmapsto} |n\theta\rangle_E |\psi\rangle_I .
\end{aligned}$$

So at the end of the computation we find the phase (scaled by n) in the first register.

We note at this point that if we are given a hermitian matrix $H = \sum_{j=1}^n \lambda_j |h_j\rangle \langle h_j|$, then the phases of the unitary $e^{iH} = \sum_{j=1}^n e^{i\lambda_j} |h_j\rangle \langle h_j|$ correspond to the eigenvalues λ_j of H. Hence, the QPE circuit applied to e^{iH} allows us to create a state corresponding to the eigenvalues of H. This can be used to simply approximate the spectrum of H or, as we do in the following, to manipulate the eigenvalues to implement matrix operations.

2.3 The HHL-circuit

We now have all the tools we need to describe the circuit solving the κ-QLSP. Suppose we have access to a hermitian $n \times n$ matrix $H = \sum_{j=1}^n \lambda_j |h_j\rangle \langle h_j|$ that satisfies $1 \geq s_1(H) \geq ... \geq s_n(H) = \frac{1}{\kappa}$ and a state $|b\rangle$ and we want to solve the linear system $Hx = |b\rangle$. For simplicity, let us assume that we can space-efficiently compute a circuit that exactly implements $U = e^{iH}$ instead of just an approximation. Then the following circuit solves the κ-QLSP. It works on three registers: An ancillary qubit $|\cdot\rangle_A$, an eigenvalue estimation register $|\cdot\rangle_E$ of size $\lceil \log \kappa^{-1} \rceil$ and an input register $|\cdot\rangle_I$ of size $\lceil \log n \rceil$.



Here the cR-gate denotes a conditioned rotation mapping

$$|0\rangle_A |n\lambda\rangle_E \mapsto \left(\frac{1}{\kappa\lambda}|0\rangle_A + \sqrt{1-\left(\frac{1}{\kappa\lambda}\right)^2}|1\rangle_A\right)|n\lambda\rangle_E .$$

Rewriting $|b\rangle = \sum_{j=1}^n \beta_j |h_j\rangle$ in the eigenbasis of H, we again analyze step by step how the initial state evolves

$$\begin{aligned}
|0\rangle_A |0\rangle_E |b\rangle_I &= \sum_{j=1}^n \beta_j |0\rangle_A |0\rangle_E |h_j\rangle_I \\
&\xmapsto{\text{QPE}} \sum_{j=1}^n \beta_j |0\rangle_A |n\lambda_j\rangle_E |h_j\rangle_I \\
&\xmapsto{cR} \sum_{j=1}^n \beta_j \left(\frac{1}{\kappa\lambda_j}|0\rangle_A + \sqrt{1-\left(\frac{1}{\kappa\lambda_j}\right)^2}|1\rangle_A\right)|n\lambda_j\rangle_E |h_j\rangle_I \\
&\xmapsto{\text{QPE}^{-1}} \sum_{j=1}^n \beta_j \left(\frac{1}{\kappa\lambda_j}|0\rangle_A + \sqrt{1-\left(\frac{1}{\kappa\lambda_j}\right)^2}|1\rangle_A\right)|0\rangle_E |h_j\rangle_I .
\end{aligned}$$

If the measurement outcome is $|0\rangle_A$ at the end of the computation, the input-register collapses to a state proportional to $\sum_{j=1}^n \frac{\beta_j}{\lambda_j}|h_j\rangle_I$, i.e. proportional to the solution vector x. Thus, we can repeatedly run the circuit until we measure $|0\rangle_A$ to obtain the desired state. In fact, estimating the probability of outcome $|0\rangle_A$ even allows us to approximate the norm $||x||$.

Ta-Shma further shows that using quantum state tomography, we can approximately learn the i-th column of H^{-1} by solving the κ-QLSP for input states $|b\rangle = |i\rangle$ multiple times.

2.4 The general non-hermitian case

So far, we only considered hermitian matrices. The HHL-algorithm also works for arbitrary regular matrices. This is because we can turn any $A \in \mathbb{C}^{n\times n}$ into a hermitian matrix $H = H(A)$ whose inverse is closely related to that of A,

$$H := \begin{bmatrix} 0 & A \\ A^\dagger & 0 \end{bmatrix}, \qquad H^{-1} = \begin{bmatrix} 0 & (A^\dagger)^{-1} \\ A^{-1} & 0 \end{bmatrix}.$$

Thus, from the inverse of H we can read of the inverse of A. Furthermore, we find that the singular values of A directly correspond to the eigenvalues of H. In particular, H is *poly-conditioned*, that is its singular values satisfy $\text{poly}(n) \geq s_1(H) \geq ... \geq s_n(H) \geq 1/\text{poly}(n)$, if and only if A is.

Lemma 16. *The eigenvalues of H are $\pm s_1(A), ..., \pm s_n(A)$.*

Proof. The eigenvalues of H correspond to the solutions of the equation $\lambda I_{2n} - H(A) = 0$. We find that the determinant of the left side is equal to

$$\det\left(\begin{bmatrix} \lambda I_n & -A \\ -A^\dagger & \lambda I_n \end{bmatrix}\right) \cdot \underbrace{\det\left(\begin{bmatrix} I_n & \lambda^{-1}A \\ 0 & I_n \end{bmatrix}\right)}_{=1} = \det\left(\begin{bmatrix} \lambda I_n & 0 \\ -A^\dagger & \lambda I_n - \lambda^{-1}A^\dagger A \end{bmatrix}\right)$$

$$= \det(\lambda I_n) \cdot \det\left(\lambda^{-1}(\lambda^2 I_n - A^\dagger A)\right) = \det\left(\lambda^2 I_n - A^\dagger A\right)$$

which is equal to 0 if and only if $\lambda = \pm s_i(A)$ for some $i \in [n]$. □

Chapter 3

Connectivity in Quantum-Logspace

In this section, we prove our main result that STCON restricted to strongly-few graphs can be decided in **BQL**.

For this, let us cite the main result of Ta-Shma [6].

Theorem 17. *Fix* $\varepsilon(n), \zeta(n) > 0$. *There exists a* $\mathrm{BQSPACE}(O(\log n + \log \varepsilon^{-1} + \log \zeta^{-1}))$ *algorithm that given an* $n \times n$ *matrix* A *with* $1 \geq s_1(A) \geq ... \geq s_n(A) \geq \zeta$, *outputs with probability* $1 - \varepsilon$ *a matrix approximating* A^{-1} *with* ε *accuracy in the* l_1 *norm.*

We can actually also handle the case where the largest singular value is greater than 1.

Corollary 18. *Fix* $\varepsilon(n), \zeta(n) > 0$ *and* $Z(n) \geq 1$. *There exists a* $\mathrm{BQSPACE}(O(\log n + \log \varepsilon^{-1} + \log \zeta^{-1} + \log Z))$ *algorithm that given an* $n \times n$ *matrix* A *with* $Z \geq s_1(A) \geq ... \geq s_n(A) \geq \zeta$, *outputs with probability* $1 - \varepsilon$ *a matrix approximating* A^{-1} *with* ε *accuracy in the* l_1 *norm.*

Proof. By scaling the matrix A with factor $1/Z$ we obtain for its singular values:

$$1 \geq s_1\left(\frac{1}{Z}A\right) \geq ... \geq s_n\left(\frac{1}{Z}A\right) \geq \frac{\zeta}{Z}.$$

Then using theorem 17, we can approximate the inverse $\left(\frac{1}{Z}A\right)^{-1} = ZA^{-1}$ up to $\varepsilon \cdot Z$ accuracy. Rescaling the result with $1/Z$ then gives an approximation of A^{-1} with ε accuracy. $\square$

It is often difficult to directly analyze singular values. We observe that we can avoid doing so if we know the l_∞ norm of the matrix and of its inverse.

Lemma 19. *If $||A||_\infty$ and $||A^{-1}||_\infty$ of an $n \times n$ matrix are well-defined and polynomially bounded, then the matrix is poly-conditioned and we can approximate the entries of A^{-1} up to $1/\mathrm{poly}(n)$ accuracy in* **BQL**.

Proof. Recall that the spectral norm and the l_∞-norm are equivalent. In fact, we find that one is within a polynomial factor of the other:

$$\frac{1}{\sqrt{n}}||A||_\infty \leq ||A||_2 \leq \sqrt{n}||A||_\infty.$$

Hence, if $||A||_\infty$ and $||A^{-1}||_\infty$ are polynomially bounded, then so are $s_1(A) = ||A||_2$ and $1/s_n(A) = ||A^{-1}||_2$. Thus, we find that in this case the functions $\varepsilon^{-1}, \zeta^{-1}$ and Z from corollary 18 are all polynomially bounded and Ta-Shma's algorithm runs in **BQL**. □

We are going to apply this result in the following two sections.

3.1 The counting Laplacian

Definition 20. For a digraph G with adjacency matrix A, the counting Laplacian is $\mathcal{L}^\# := I - A$.

Lemma 21. *If G is a DAG with $n = |V|$ vertices, then the counting Laplacian $\mathcal{L}^\#$ is invertible and we find*

$$(\mathcal{L}^\#)^{-1} = (I - A)^{-1} = I + A + A^2 + \cdots + A^{n-1}.$$

So that $(\mathcal{L}^\#)^{-1}(i,j)$ equals the number of paths from i to j.

Proof. Let G be a DAG with n vertices. Then there is no path in G of length n or higher. For A the adjacency matrix of G this translates to $A^n = 0$, i.e. A is nilpotent of order at most n. Hence, the Neumann-series $\sum_{k\in\mathbb{N}} A^k = I + A + A^2 + ... + A^{n-1}$ becomes a sum and converges so that the rest of the statement follows from basic operator theory. □

The inverse of the previous lemma is not true, i.e. if a graph contains a cycle this does not imply that the counting Laplacian $\mathcal{L}^\#$ is singular. However, the entries of the inverse do not seem to contain any helpful information then. We immeadiately obtain

Corollary 22. *The counting Laplacian $\mathcal{L}^\#$ of a DAG G is poly-conditioned if and only if G is strongly-few.*

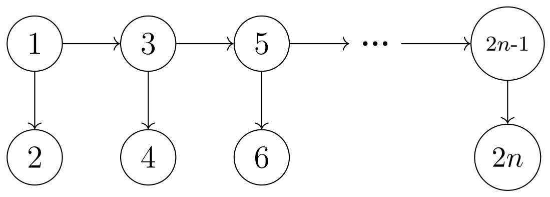


Figure 3.1: Hard instance for a random walk.

Using corollary 19 this gives our main result.

Theorem 23. STCON *restricted to strongly-few graphs can be decided in* **BQL**.

Example 24. Consider the directed tree in figure 3.1. A classical random walk has probability $1/2^{n-1}$ to hit node $2n$ starting from 1. A counter to keep track of the needed attemps to reach $2n$ would thus exceed reasonable space bounds. On the other hand, the number of paths between any two nodes is polynomially bounded, in fact it is either 0 or 1. Hence, we can invert $\mathcal{L}^{\#}$ using Ta-Shmas algorithm in **BQL** and decide if $2n$ is reachable from 1 by checking whether $(\mathcal{L}^{\#})^{-1}_{1,2n} = 1$.

The example shows that the counting Laplacian allows to solve connectivity for any directed tree. This includes the tree instances from Poon in [17] used to show the space lower bound of $\Omega(\frac{\log^2 n}{\log\log n})$ in the restricted NNJAG-model. In this model algorithms start with some number of pebbles at the first node and are only allowed to move them along edges or jump them to each other's location. But in fact connectivity for trees can already be solved in **L** as is proved in the appendix A.12. It is one of the main criticisms of the NNJAG-model that it does not allow efficient tree-traversals. For the instances used to show the tight space lower bound of $\Omega(\log^2 n)$ [18] in the NNJAG-model, our approach with the counting Laplacian fails.

At a first glance, it may seem that undirected edges are problematic for our approach. If we use lemma 14 to construct a layered DAG starting from a graph G with two or more adjacent undirected edges, then the DAG will contain exponentially many paths. This follows because a walk on G has exponentially many possibilities to wait on the node where the undirected edges intersect by walking the edges back and forth. However, since USTCON is in **L** we can avoid this problem. Instead of directly reducing the graph to a DAG, we can first contract all undirected components into single nodes and then invert the counting Laplacian of the resulting graph. This way, we can decide STCON in **BQL** for graphs containing undirected edges as long as the resulting graph after contraction is strongly-few.

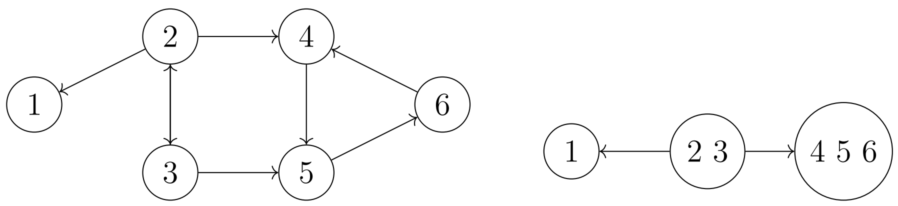


Figure 3.2: On the right the condensation of the graph on the left.

We remark that the contraction of undirected components corresponds to computing the condensation of a graph where all strongly connected components contain only undirected edges.

Definition 25. The *condensation* of a graph G, $SC[G]$, is obtained from G by identifying all vertices of the same strongly connected component as a single node (cf. figure 3.2).

Note that the condensation is always a DAG.

3.2 The random-walk Laplacian

Definition 26. For a graph G with adjacency matrix A the random-walk Laplacian is $\mathcal{L}^{\mathrm{rw}} := I - S$ where $S := D^{+}A$.

Here D^{+} denotes the pseudo-inverse of the out-degree matrix D, i.e. $D^{+}(i,i) = 1/d(i)$ if $d(i) > 0$, $D^{+}(i,i) = 1$ if $d(i) = 0$ and $D(i,j) = 0$ if $i \neq j$. It should be noted that S need not be a stochastic matrix if it contains sinks, i.e. vertices without outgoing edges. In general it is substochastic, meaning for all $i \in [n]$: $\sum_j S(i,j) \leq 1$.

Lemma 27. *Let G be a digraph so that the expected number of steps until a random walk hits a sink are finite, then the random-walk Laplacian $\mathcal{L}^{\mathrm{rw}}$ is invertible and we find*

$$(\mathcal{L}^{\mathrm{rw}})^{-1} = (I - S)^{-1} = \sum_{k \in \mathbb{N}} S^k.$$

So that $(\mathcal{L}^{\mathrm{rw}})^{-1}(i,j)$ equals the expected number of visits to a non-sink j of a random walk from node i.

Proof. Let G be a digraph so that the expected number of steps until a random walk hits a sink are finite. Further, let $(X_t)_{t \in \mathbb{N}}$ be a Markov Chain corresponding to a random walk on G (with probability transition matrix P where we add self-loops

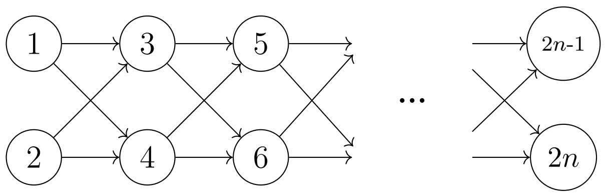


Figure 3.3: DAG with exponentially many paths.

to sinks so that the chain is well-defined). We find that

$$\begin{aligned}\sum_{k=0}^{\infty} S^k(i,j) &= \sum_{k=0}^{\infty} \mathbb{P}[X_k = j | X_0 = i] \\ &= \sum_{k=0}^{\infty} \mathbb{E}\left[\mathbf{1}_{\{X_k=j\}} | X_0 = i\right] \\ &= \mathbb{E}\left[\sum_{k=0}^{\infty} \mathbf{1}_{\{X_k=j\}} \middle| X_0 = i\right] < \infty\end{aligned}$$

converges for all $(i,j) \in V^2$ and is indeed equal to the expected number of visits to node j of a random walk from node i. Hence, $\mathcal{L}^{\text{rw}} = I - S$ is invertible and its inverse is $(I-S)^{-1} = \sum_{k=0}^{\infty} S^k$. □

Similar to the analysis for the counting Laplacian we find:

Corollary 28. *The random-walk Laplacian of a digraph G is poly-conditioned if and only if for all pairs of non-sink vertices $(i,j) \in V^2$ the expected number of visits to j of a random walk from i are polynomially bounded.*

In particular, the random-walk Laplacian for a DAG is always poly-conditioned. However, the inverse polynomial accuracy of Ta-Shma's procedure for matrix inversion becomes a problem here because the expected number of visits can be negligible as in figure 3.1.

Example 29. See figure 3.3 for a DAG where the number of paths is exponential in the number of nodes. The counting Laplacian of the graph is thus not poly-conditioned. But the random walk Laplacian easily decides connectivity for this graph because the expected number of visits to any node are either 0 or greater or equal to 1/2. Inverse polynomial accuracy for approximating the entries is thus sufficient.

3.3 Open Questions

We observe two open questions for further research.

1. Can we generalize our approach with the counting Laplacian to decide STCON on reach-few graphs? A thorough spectral analysis of the counting Laplacian might show that the subspaces corresponding to negligible singular values of subgraphs with exponentially many paths can be disregarded if they are not reachable from the frist node.

2. What is the relation of the singular values of different graph related matrices and the spectral expansion? We see that the instances for which the random-walk Laplacian is poly-conditioned are inherently different from the ones for which the counting Laplacian is. A comparison of the singular values of different matrices and the spectral expansion might give closer insight into the classes of digraphs for which connectivity can be decided in **BQL** and that may be hard classicaly.

Bibliography

1. Aleliunas, R., Karp, R. M., Lipton, R. J., Lovasz, L. & Rackoff, C. *Random walks, universal traversal sequences, and the complexity of maze problems* in *20th Annual Symposium on Foundations of Computer Science* (1979), 218–223.
2. Reingold, O. Undirected connectivity in log-space. *J. ACM* **55** (2008).
3. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences* **4,** 177–192 (1970).
4. Allender, E. & Lange, k.-j. RUSPACE($\log n$) $\subseteq$ DSPACE($\log^2 n/\log\log n$). *Theory of Computing Systems* **31** (1998).
5. Garvin, B., Stolee, D., Tewari, R. & Vinodchandran, N. ReachFewL = ReachUL. *Electronic Colloquium on Computational Complexity (ECCC)* **18,** 60 (2011).
6. Ta-Shma, A. *Inverting Well Conditioned Matrices in Quantum Logspace* in *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing* (2013), 881–890.
7. Csanky, L. Fast Parallel Matrix Inversion Algorithms. *SIAM Journal on Computing* **5,** 618–623 (1976).
8. Harrow, A. W., Hassidim, A. & Lloyd, S. Quantum Algorithm for Linear Systems of Equations. *Physical Review Letters* **103** (2009).
9. Fefferman, B. & Remscrim, Z. *Eliminating Intermediate Measurements in Space-Bounded Quantum Computation* in *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing* (Association for Computing Machinery, Virtual, Italy, 2021), 1343–1356.
10. Melkebeek, D. v. & Watson, T. Time-Space Efficient Simulations of Quantum Computations. *Theory of Computing* **8,** 1–51 (2012).

11. Cook, S. A. A taxonomy of problems with fast parallel algorithms. *Information and Control* **64.** International Conference on Foundations of Computation Theory, 2–22 (1985).

12. Levin, D. A., Peres, Y. & Wilmer, E. L. *Markov chains and mixing times* (American Mathematical Society, 2006).

13. Chung, F. R. K. *Spectral Graph Theory* (American Mathematical Society, 1997).

14. Reingold, O., Trevisan, L. & Vadhan, S. *Pseudorandom Walks on Regular Digraphs and the RL vs. L Problem* in *Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing* (2006), 457–466.

15. Lange, k.-j. *An unambiguous class possessing a complete set* in (1997), 339–350.

16. Nielsen, M. A. & Chuang, I. L. *Quantum Computation and Quantum Information* (Cambridge University Press, 2000).

17. Poon, C. *On the Complexity of the st-Connectivity Problem* PhD thesis (University of Toronto, 1997).

18. Edmonds, J., Poon, C. K. & Achlioptas, D. Tight Lower Bounds for st-Connectivity on the NNJAG Model. *SIAM Journal on Computing* **28,** 2257–2284 (1999).

19. Furst, M., Saxe, J. B. & Sipser, M. *Parity, circuits, and the polynomial-time hierarchy* in *22nd Annual Symposium on Foundations of Computer Science* (1981), 260–270.

20. Saks, M. & Zhou, S. BPHSPACE(S)$\subseteq$DSPACE($S^{3/2}$). *Journal of Computer and System Sciences* **58,** 376–403 (1999).

21. Lassaigne, R. & Rougemont, M. *Logic and Complexity* (2004).

22. Cook, S. A. & McKenzie, P. Problems complete for deterministic logarithmic space. *Journal of Algorithms* **8,** 385–394 (1987).

Appendix

A.1 Inclusions of Complexity Classes

Lemma A.1. $\mathbf{NC}^0 \subsetneq \mathbf{AC}^0$.

Proof. The subset relation $\mathbf{NC}^k \subseteq \mathbf{AC}^k$ is trivial for all $k \in \mathbb{N}$ since $\mathbf{NC}^k$-circuits are just $\mathbf{AC}^k$-circuits with the additional restriction of bounded fanin 2.
The inequality for $k = 0$ holds since every output bit of an $\mathbf{NC}^0$-circuit-family of constant depth d can depend on at most 2^d input bits, i.e. constantly many, due to the fanin 2 restriction. Thus, for example the language $\{x \in \{0,1\}^* | \ x_i = 1 \ \forall i\}$ to decide whether all input bits are 1 is not in $\mathbf{NC}^0$. However, it is obviously in $\mathbf{AC}^0$ using a single AND-gate with unbounded fanin. □

Lemma A.2. $\mathbf{AC}^0 \subsetneq \mathbf{NC}^1$.

Proof. The inclusion $\mathbf{AC}^k \subseteq \mathbf{NC}^{k+1}$ is true for all $k \in \mathbb{N}$. Given an $\mathbf{AC}^k$-circuit we can replace every occurence of AND- or OR-gates using $O(n^c)$ fanin by a binary tree of $O(\log(n))$-many AND- or OR-gates. Even if every gate is replaced in this fashion, we obtain at most an $O(\log(n))$-factor on the depth. The resulting circuit is thus in $\mathbf{NC}^{k+1}$.
The inequality was proven in [19] by Furst, Saxe and Sipser. They showed that the parity function $\oplus$ that maps every $x \in \{0,1\}^n$ to $\oplus(x_1, ..., x_n) = \Sigma_i x_i \mod 2$ is not in $\mathbf{AC}^0$. However, it is simple to see that the parity function can be computed by an $\mathbf{NC}^1$-circuit as a binary tree of many XOR-gates. □

Lemma A.3. $\mathbf{NC}^1 \subseteq \mathbf{L}$.

Proof. We find $\mathbf{NC}^k \subseteq \mathbf{L}^k$ for all $k \in \mathbb{N}^*$. Given a language decided by some $\mathbf{NC}^k$-circuit we can determine the value of any output bit of the circuit during a depth-first search (DFS) exploration from it. During the DFS exploration we always save the current path we are on as a boolean sequence of edge labels, all past evaluations on the path and the node label we are currently at. This is possible in space $O(\log^k n)$ because of the bounded depth. Whenever it is possible to update the evaluation of the currently visited node, because it is a leave or because all child nodes are evaluated, we do so. This procedure allows us to eventually evaluate the output bit. $\square$

Lemma A.4. $\mathbf{L} \subseteq \mathbf{RL} \subseteq \mathbf{NL}, \mathbf{BPL}$ *and* $\mathbf{BPL} \subseteq \mathbf{BQL}$.

Proof. Clear from the definition. $\square$

Lemma A.5. $\mathbf{NL} \subseteq \mathbf{AC}^1$.

Proof. We show that STCON is in $\mathbf{AC}^1$. For this, note that given two boolean $n \times n$ matrices A and B, we can calculate in $\mathbf{AC}^0$ their boolean "product"

$$(AB)(i,j) := \bigvee_{k=1}^{n} \left(A(i,k) \wedge B(k,j)\right).$$

This notion of a product obviously does not coincide with the usual boolean matrix multiplication because we replaced boolean addition by a simple OR-gate.

Now suppose we are given the adjacency matrix A of some input graph G with nodes s and t. Repeadetly squaring, in the sense we just defined, allows us to calculate any power A^k for $k \leq n$ in $\mathbf{AC}^1$. Further, it is easy to see that $A^k(s,t)$ equals 1 if and only if there is a path of length k from s to t. Hence, returning $\bigvee_{k=1}^{n} A^k(s,t)$ correctly decides STCON in $\mathbf{AC}^1$. $\square$

Lemma A.6. $\mathbf{NL} \subseteq \mathbf{DET}$.

Proof. As in [11] we reduce STCON to intDET. Let A be the adjacency matrix of the input graph G with nodes $1, ..., n$. Without loss of generality we assume the diagonal entries of A to be zero, i.e. G has no self-loops. Note that then $A^k(i,j)$ equals the number of paths of length k from i to j. We now choose $\varepsilon := \frac{1}{n} < ||A||^{-1}$ and consider the matrix $M := I - \varepsilon A$. We find that it is invertible and

$$M^{-1} = (I - \varepsilon A)^{-1} = I + \varepsilon A + (\varepsilon A)^2 + (\varepsilon A)^3 + ...$$

such that $M^{-1}(i,j) \neq 0$ if and only if there exists a path from i to j. In particular we have $M^{-1}(1,n) \neq 0$ if and only if there exists a path from 1 to n. Using the adjugate of M we can calculate $M^{-1} = \frac{\operatorname{adj}(M)}{\det(M)}$ and we find $M^{-1}(1,n) \neq 0$ if and only if $\det\big(M_{[n|1]}\big) \neq 0$ if and only if $\det\big((nI - A)_{[n|1]}\big) \neq 0$. Here $M_{[n|1]}$ refers to the matrix that is obtained from M by cutting out row n and column 1. Thus we can reduce STCON to determining the determinant of $(nI - A)_{[n|1]}$. $\square$

Lemma A.7. $\mathbf{BPL} \subseteq \mathbf{L}^{3/2}$.

The result is due to Saks and Zhou [20].

Lemma A.8. $\mathbf{BQL} \subseteq \mathbf{DET}$.

The result is proven by Fefferman and Remscrim in [9].

Lemma A.9. $\mathbf{AC}^1 \subseteq \mathbf{NC}^2$.

Proof. Similar to $\mathbf{AC}^0 \subseteq \mathbf{NC}^1$. □

Lemma A.10. $\mathbf{DET} \subseteq \mathbf{NC}^2$.

The proof is due to Csanky [7]. A very accessible presentation can be found in [21].

Lemma A.11. $\mathbf{NC}^2 \subseteq \mathbf{L}^2$.

Proof. Similar to $\mathbf{NC}^1 \subseteq \mathbf{L}$. □

A.2 Connectivity on directed trees

Lemma A.12. *Let T be a directed tree with nodes s and t. There is a deterministic logspace algorithm deciding if there is a path from s to t.*

Proof. Note that given a rooted undirected tree, we can compute a DFS-exploration in deterministic logspace since we do not need to keep track of the previously visited nodes (cf. [22]). The basic idea of the algorithm is to first ignore all edge orientations and consider the resulting undirected tree rooted at s and check whether s and t are weakly connected using a DFS-exploration. If not, we already know there cannot be a path between them. If yes, we consider every out-neighbour s' of s and the corresponding subtree T' rooted at s' that we obtain by deleting the edge from s to s' in T. Clearly, there is a directed path from s to t in T if and only if there exists some out-neighbour s' such that there is a directed path from s' to t in T'. Further, we know that if such s' exists, it needs be on the path from s to t. Applying the same procedure to T' gives the desired logspace algorithm by induction. □